4:e utgåvan

FULLSTÄNDIGT UPPDATERAD FÖR **RASPBERRY PI 400**

# DEN OFFICIELLA Raspberry Pi nybörjarguiden för

## Så här använder du din nya dator


av Gareth Halfacree

# DEN OFFICIELLA Raspberry Pi nybörjarguiden för

## Så här använder du din nya dator


Publicerad för första gången 2021 av Raspberry Pi Trading Ltd, Maurice Wilkes Building,
St. John's Innovation Park, Cowley Road, Cambridge, CB4 0DS

Förlagschef: Russell Barnes • Redaktör: Phil King
Design: Critical Media • Illustrationer: Sam Alder
VD: Eben Upton

ISBN: 978-1-912047-84-0

# Välkommen till den officiella nybörjarguiden för Raspberry Pi

Vi tror att du kommer att älska Raspberry Pi. Oavsett vilken modell du har – ett vanligt Raspberry Pi-kort eller den nya Raspberry Pi 400 med integrerat tangentbord – kan du använda den här prisvärda datorn för att lära dig kodning, bygga robotar och skapa alla slags märkliga och härliga projekt.

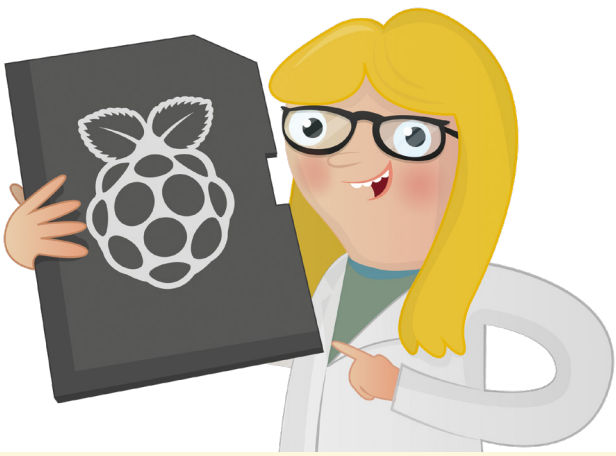
Raspberry Pi kan göra allt du förväntar dig av en dator – allt från att surfa på internet och spela spel till att titta på filmer och lyssna på musik. Men Raspberry Pi är mycket mer än en modern dator.

Med en Raspberry Pi kan du komma in till datorns kärna. Du får installera ett eget operativsystem och kan ansluta kablar och kretsar direkt till GPIO-stiften. Den har utformats för att lära unga människor att programmera på språk såsom Scratch och Python, och alla större programmeringsspråk ingår i det officiella operativsystemet.

Världen behöver programmerare mer än någonsin, och Raspberry Pi har tänt en kärlek till datavetenskap och teknik i en ny generation.

Människor i alla åldrar använder Raspberry Pi för att skapa spännande projekt: allt från retrospelkonsoler till internetanslutna väderstationer.

Så om du vill skapa spel, bygga robotar eller programmera en mängd fantastiska projekt, så finns den här boken här för att hjälpa dig att komma igång.



# Om författaren

Gareth Halfacree är en frilansande teknikjournalist, författare och tidigare systemadministratör inom utbildningssektorn. Hans passion för programvara och maskinvara med öppen källkod gjorde att han tidigt började använda Raspberry Pi-plattformen, och han har skrivit flera publikationer om dess kapacitet och flexibilitet. Han finns på Twitter som **@ghalfacree** och på sin hemsida **freelance.halfacree.co.uk**.

# Innehåll



## BILAGOR

Kapitel 1

# Lär känna Raspberry Pi

**Bekanta dig med din nya dator i kreditkortsstorlek genom att följa med på en guidad rundtur i Raspberry Pi. Upptäck de många komponenterna och vad de gör**

Raspberry Pi är en anmärkningsvärd enhet: en fullt fungerande dator i ett litet och billigt paket. Oavsett om du letar efter en enhet som du kan använda för att surfa på nätet eller spela spel, är intresserad av att lära dig att skriva egna program eller vill kunna skapa egna kretsar och fysiska enheter kommer Raspberry Pi, och dess fantastiska community, att stödja dig varje steg på vägen.

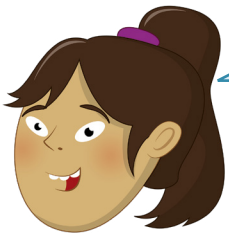
Raspberry Pi är en så kallad *enkortsdator*, vilket betyder precis vad det låter som: en dator som är byggd på ett enda *kretskort*. Liksom de flesta enkortsdatorer är Raspberry Pi liten – ungefär samma storlek som ett kreditkort – men det betyder inte att den inte är kraftfull: en Raspberry Pi kan göra allt som en större och mer energislukande dator kan göra, men inte nödvändigtvis lika snabbt.

Raspberry Pi-familjen föddes ur en önskan att uppmuntra till mer praktisk datorutbildning runt om i världen. Skaparna, som gick ihop för att bilda den ideella Raspberry Pi Foundation, hade ingen aning om att det skulle bli så populärt: de få tusen datorer som byggdes 2012 för att sondera intresset sålde slut omedelbart, och miljontals datorer har skickats över hela världen under åren efter detta. Kretskorten har tagit sig in i hem, klassrum, kontor, datacenter, fabriker och till och med självstyrande båtar och rymdballonger.

Olika modeller av Raspberry Pi har släppts sedan den ursprungliga B-modellen, och var och en har antingen förbättrade specifikationer eller funktioner som är specifika för ett visst användningsfall. Raspberry Pi Zero-familjen är till exempel en liten version av den fullstora Raspberry Pi som blivit av med några funktioner, i synnerhet de många USB-portarna och den trådbundna nätverksporten, till förmån för en betydligt mindre layout och minskade energikrav.

Alla Raspberry Pi-modeller har dock en sak gemensamt: de är *kompatibla*, vilket innebär att programvara som skrevs för en modell kan köras på vilken annan modell som helst. Det är till och med möjligt att ta den senaste versionen av Raspberry Pi:s operativsystem och köra den på en ursprunglig prototyp av B-modellen som byggdes före lanseringen. Det kommer att gå långsammare, det är sant, men det kommer fortfarande att fungera.

I denna bok får du lära dig mer om Raspberry Pi 4 Model B och Raspberry Pi 400, de senaste och mest kraftfulla versionerna av Raspberry Pi. Det som du lär dig kan dock enkelt appliceras på andra modeller i Raspberry Pi-familjen, så det gör inget om du använder en annan version.



### RASPBERRY PI 400

Om du har en Raspberry Pi 400 är kretskortet inbyggt i tangentbordshöljet. Läs vidare för att lära dig mer om alla komponenter som får Raspberry Pi att fungera, eller hoppa till sidan 20 för en rundtur i enheten.

## En guidad rundtur i Raspberry Pi

Till skillnad från en traditionell dator, som döljer sina inre funktioner i ett hölje, har en vanlig Raspberry Pi alla sina komponenter, portar och funktioner synliga – även om du kan köpa ett hölje för att ge extra skydd, om du föredrar det. Detta gör den till ett utmärkt verktyg för att lära sig om vad de olika delarna av en dator gör, och det gör det också lätt att lära sig var saker ska sitta när det är dags att ansluta de olika tillbehören, så kallad *kringutrustning*, som du måste ha för att komma igång.

- A USB Type-C för ingående ström
- B DSI-bildskärmsport
- C Trådlöst/Bluetooth
- D Micro-HDMI 0
- E Micro-HDMI 1
- F System-på-chipp
- G GPIO
- H RAM
- I CSI-kameraport
- J 3,5 mm AV-uttag
- K PoE
- L USB 2.0
- M USB 3.0
- N Ethernet-port

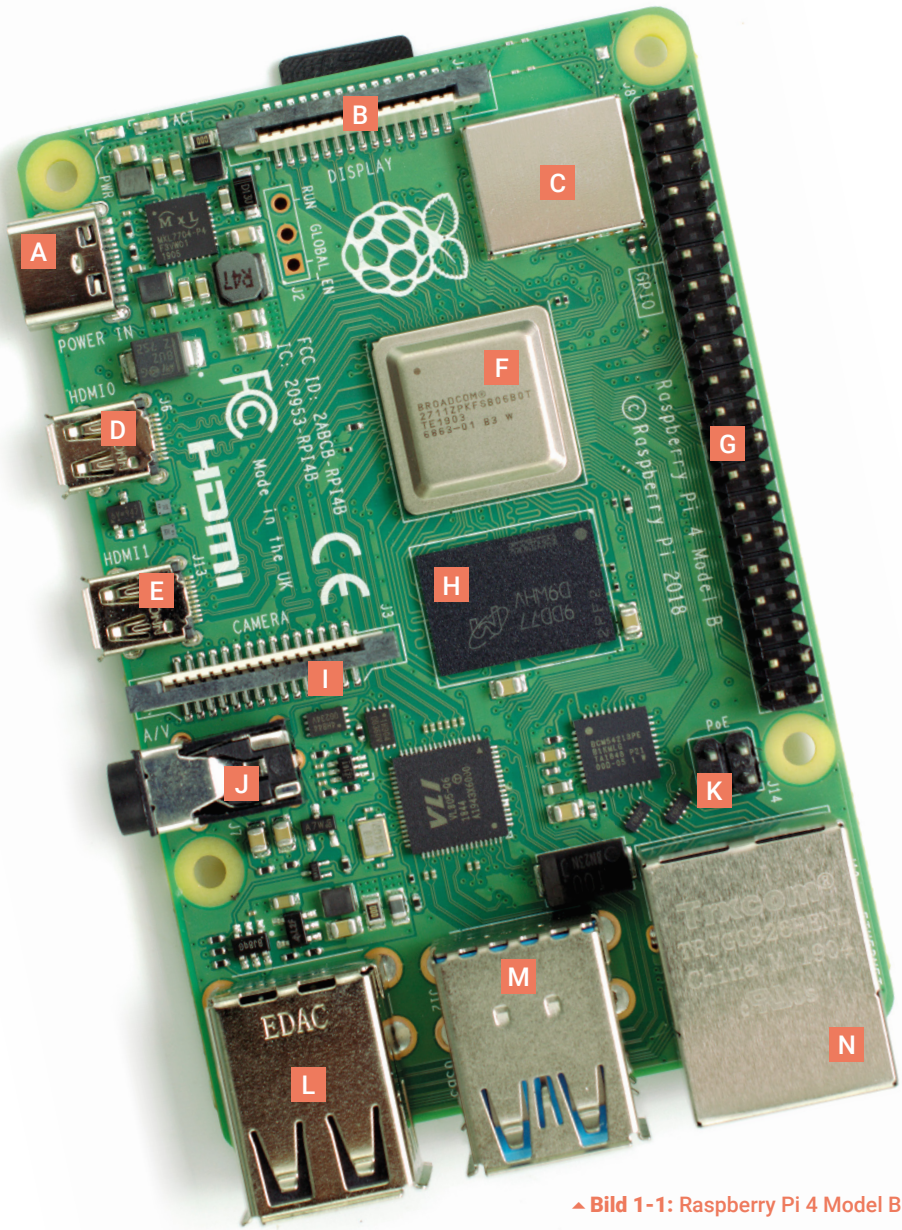


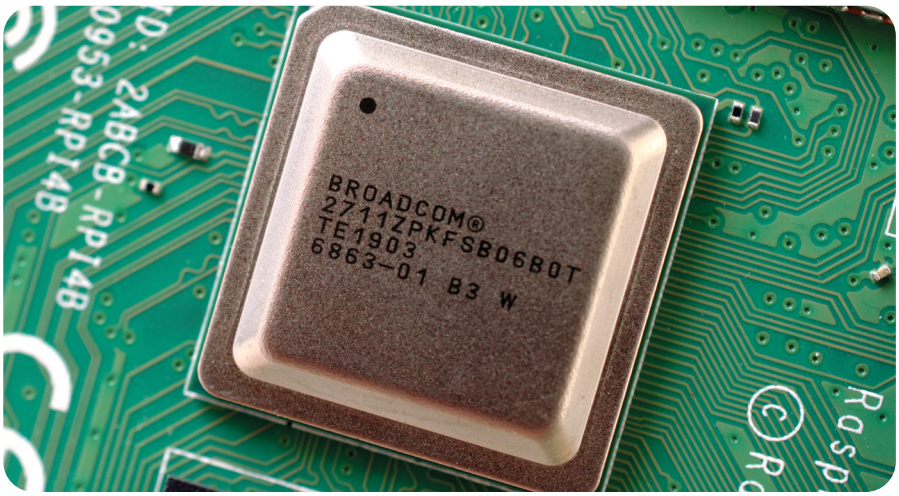

**Bild 1-1:** Raspberry Pi 4 Model B

**Bild 1-1** visar en Raspberry Pi 4 Model B sedd ovanifrån. När du använder en Raspberry Pi med den här boken bör du försöka hålla den vänd på samma sätt som på bilden; om den vänds kan det bli förvirrande när det gäller att använda saker som GPIO (detaljerad beskrivning i **Kapitel 6, Fysisk programmering med Scratch och Python**).

Även om det kan se ut som om det är mycket intryckt på det lilla kortet, är en Raspberry Pi väldigt enkel att förstå, och vi börjar med *komponenterna*, de inre processerna som får den att fungera.

## Raspberry Pi:s komponenter

Precis som alla datorer består Raspberry Pi av olika komponenter, där var och en har en roll att spela för att datorn ska fungera. Den första och förmodligen viktigaste av komponenterna sitter precis ovanför mittpunkten på den övre sidan av kortet (**Bild 1-2**) och täcks av ett metallock: *system på chipp* (SoC).




▲ **Bild 1-2:**Raspberry Pi:s system på chipp (SoC)

Namnet system på chipp är en bra indikator på vad du skulle hitta om du tog bort metallhöljet: ett kiselchipp, känt som en *integrerad krets*, som innehåller huvuddelen av Raspberry Pi-systemet. Det inkluderar den *centrala processorenheten* (CPU), ofta betraktad som datorns hjärna, och *grafikbehandlingsenheten* (GPU), som hanterar den visuella sidan av saker och ting.

En hjärna är inte mycket att ha utan minne, så precis bredvid SoC hittar du exakt det: ett annat chipp, som ser ut som ett liten, svart plastkvadrat (**Bild 1-3**, på nästa sida). Detta är Raspberry Pi:s *random access memory (RAM)*. När du arbetar på Raspberry Pi är det RAM-minnet som har hand om det du gör. Först när du sparar ditt arbete kommer det att skrivas till microSD-kortet. Tillsammans bildar dessa komponenter Raspberry Pi:s flyktiga och icke-flyktiga minnen: det flyktiga RAM-minnet förlorar sitt innehåll när Raspberry Pi stängs av, medan det icke-flyktiga microSD-kortet behåller sitt innehåll.

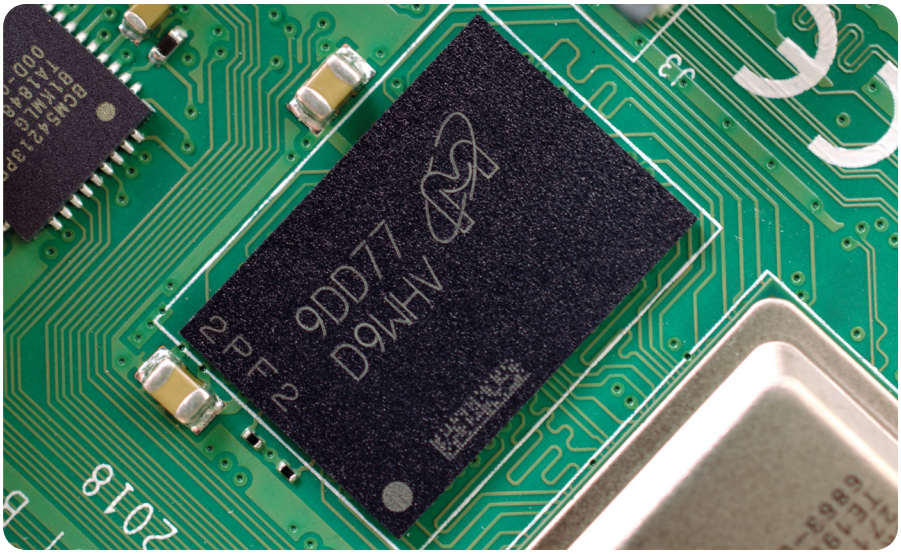



**Bild 1-3:** Raspberry Pi:s random access memory (RAM)

Överst till höger på kortet hittar du ett annat metallock (**Bild 1-4**) som täcker *radion*, den komponent som ger Raspberry Pi möjlighet att kommunicera trådlöst med enheter. Själva radion fungerar i själva verket som två huvudkomponenter: en *Wi-Fi-radio* för anslutning till datanätverk och en *Bluetooth-radio* för anslutning till kringutrustning som möss och för att skicka data till eller ta emot data från närliggande smarta enheter som sensorer eller smarttelefoner.

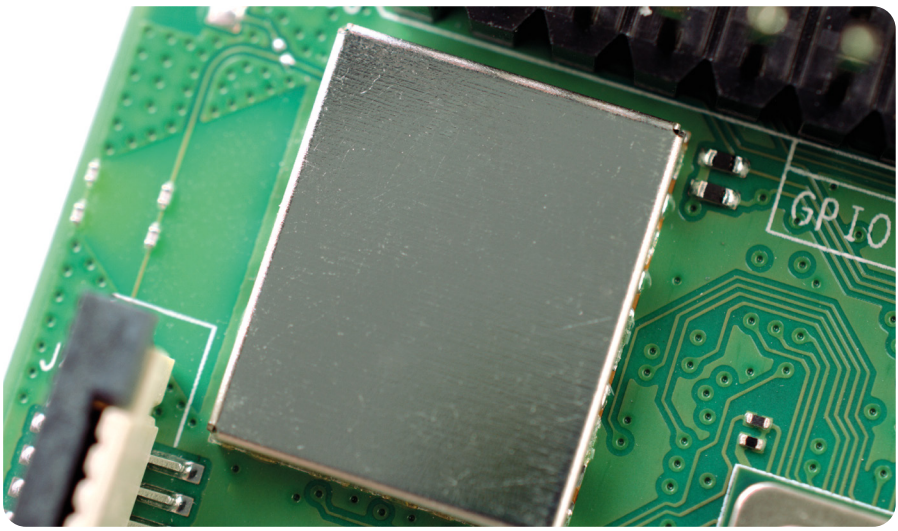
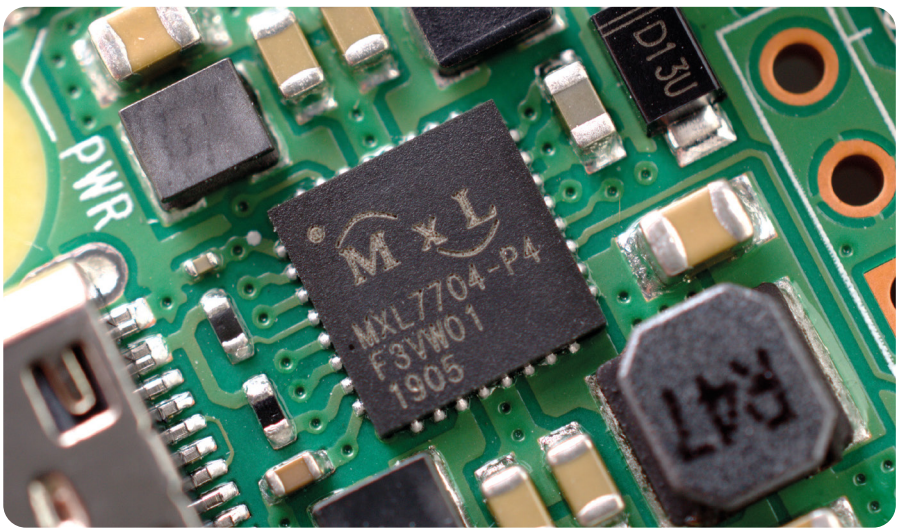



**Bild 1-4:** Raspberry Pi:s radiomodul

Ett annat svart, plasttäckt chipp sitter vid kortets nedre kant, strax bakom den mellersta uppsättningen USB-portar. Detta är *USB-kontrollenheten*, och den har ansvar för att driva de fyra USB-portarna. Bredvid det sitter ett ännu mindre chipp, *nätverkskontrollenheten*, som hanterar Raspberry Pi:s Ethernet-nätverksport. Ett sista svart chipp, som är mindre än de övriga, sitter lite ovanför USB Type-C-strömkontakten längst upp till vänster om kortet (**Bild 1-5**). Detta är känt som en PMIC (*power management integrated circuit*) och hanterar omvandlingen av strömmen som kommer in från micro-USB-porten till den ström som Raspberry Pi behöver för att fungera.




▲ **Bild 1-5:** Raspberry Pi:s integrerade strömhanteringskrets (PMIC)

Oroa dig inte om det här känns som mycket att ta in: du behöver inte veta vad varenda komponent är eller var du hittar den på kretskortet för att kunna använda Raspberry Pi.

## Raspberry Pi:s portar

Raspberry Pi har en uppsättning portar som börjar med fyra USB-portar (*Universal Serial Bus*) (**Bild 1-6**) i mitten och till höger på den nedre kanten. Med dessa portar kan du ansluta all USB-kompatibel kringutrustning, från tangentbord och möss till digitalkameror och flashenheter, till Raspberry Pi. Tekniskt sett finns det två typer av USB-portar: de med svarta delar inuti är USB 2.0-portar, baserade på version två av Universal Serial Bus-standarden; de med blå delar är snabbare USB 3.0-portar, som baseras på den nyare tredje versionen.

▲ **Bild 1-6:** Raspberry Pi:s USB-portar

Till höger om USB-portarna finns en *Ethernet-port*, även känd som en *nätverksport* (**Bild 1-7**). Du kan använda den här porten för att ansluta Raspberry Pi till ett trådbundet datornätverk med en kabel som har en så kallad RJ45-kontakt i änden. Om du tittar noga på Ethernet-porten ser du två ljusdioder (LED) längst ner. Dessa är statuslysdioder och meddelar att anslutningen fungerar.

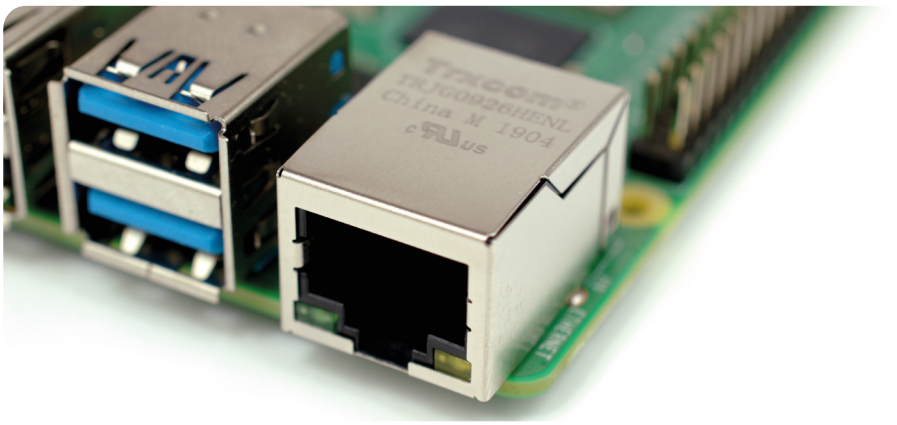


**Bild 1-7:** Raspberry Pi:s Ethernet-port

Precis ovanför USB-portarna, till vänster på Raspberry Pi, finns ett *3,5 mm audiovisuellt uttag (AV)* (**Bild 1-8**). Detta kallas även för *hörlursuttag* och det kan användas för just det ändamålet – även om du får bättre ljud om du ansluter till förstärkta högtalare än om du använder hörlurar. Det har dock en dold, extra funktion: förutom ljud har 3,5 mm AV-uttaget en videosignal som kan anslutas till tv-apparater, projektorer och andra skärmar som har stöd för en *kompositvideosignal* med en speciell kabel som kallas för en *TRRS-adapter (tip-ring-ring-sleeve)*.

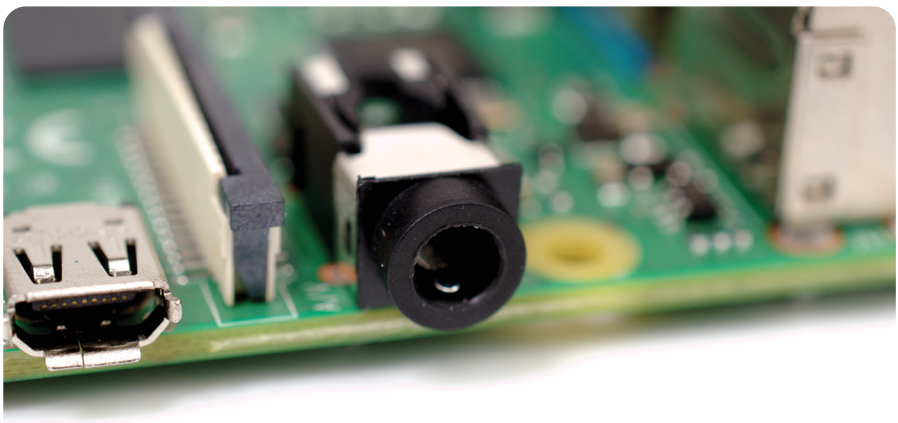
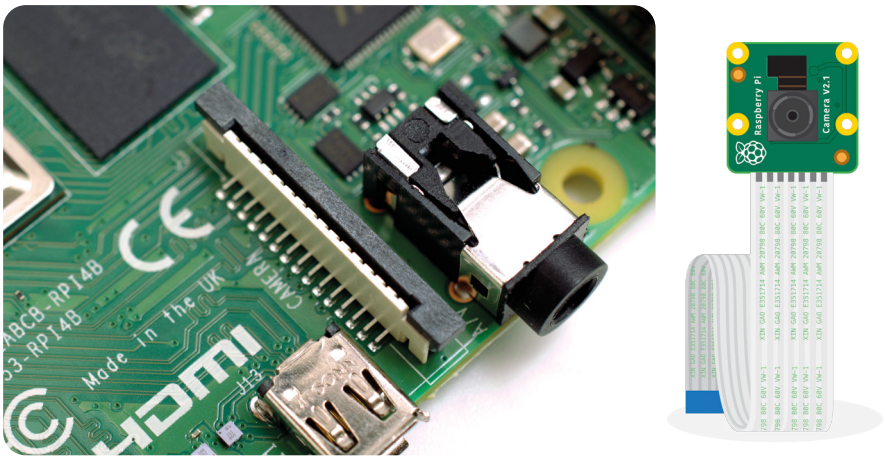


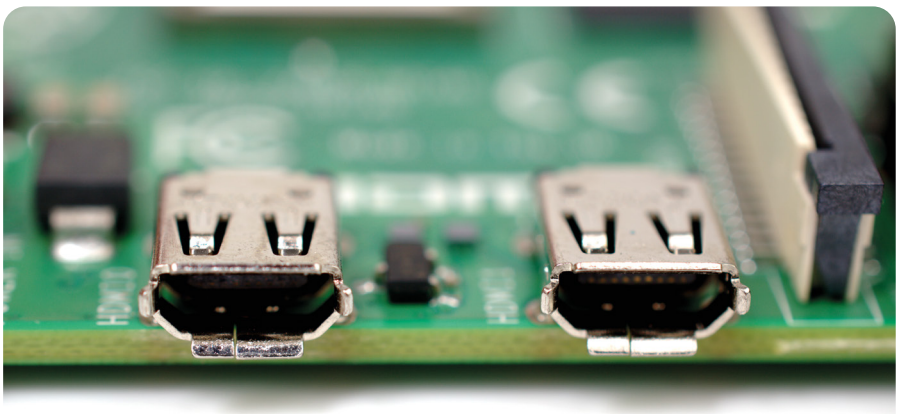
**Bild 1-8:** Raspberry Pi:s 3,5 mm AV-uttag

Direkt ovanför 3,5 mm AV-uttaget finns en kontakt som ser lite märklig ut med en plastflik som går att dra upp. Detta är *kamerakontakten*, som även kallas *Camera Serial Interface (CSI)* (**Bild 1-9**). Denna gör det möjligt att använda den specialdesignade Raspberry Pi Camera Module (som du får lära dig mer om i **Kapitel 8, Raspberry Pi Camera Module**).




▲ **Bild 1-9:** Raspberry Pi:s kamerakontakt

Ovanför den, fortfarande på den vänstra kanten av kretskortet, finns *micro-HDMI-portarna (High Definition Multimedia Interface)*, som är en mindre version av kontakterna som du hittar på en spelkonsol, digitalbox eller tv (**Bild 1-10**). Delen ”multimedia” i namnet talar om att den bär både ljud- och videosignaler, medan high-definition säger att du kan förvänta dig utmärkt kvalitet. Du använder dessa för att ansluta Raspberry Pi till en eller två skärmenheter: en datorskärm, tv eller projektor.



▲ **Bild 1-10:** Raspberry Pi:s micro-HDMI-portar

Ovanför HDMI-portarna finns en *USB Type-C-strömport* (**Bild 1-11**) som du använder för att ansluta Raspberry Pi till en strömkälla. USB Type-C-porten är en vanlig syn på smarttelefoner, surfplattor och andra bärbara enheter. Även om du kan använda en vanlig mobilladdare för att driva Raspberry Pi, bör du använda den officiella Raspberry Pi USB Type-C-strömförsörjningen för bästa resultat.

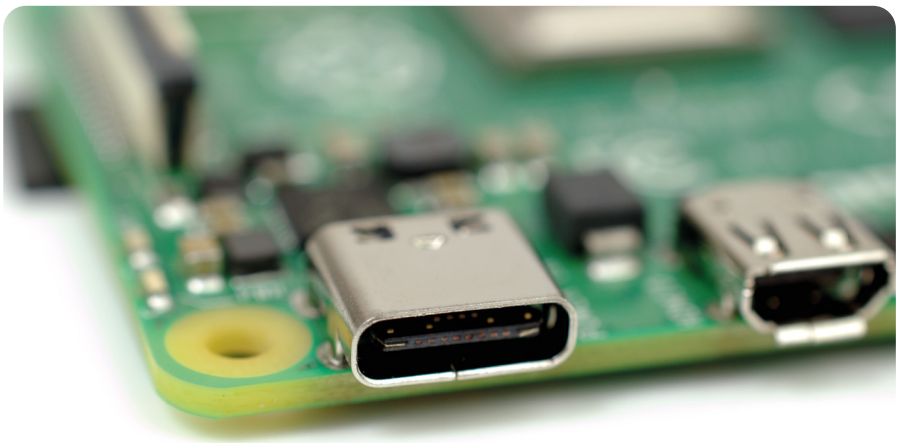


**Bild 1-11:** Raspberry Pi:s USB Type-C-strömport

Vid den övre kanten av kretskortet finns en annan konstig kontakt (**Bild 1-12**), som vid första anblicken ser identisk ut med kamerakontakten. Detta är emellertid dess raka motsats: en *bildskärmskontakt* eller *Display Serial Interface (DSI)* som är designad för användning med en Raspberry Pi Touch Display (**Bild 1-13**, på nästa sida).

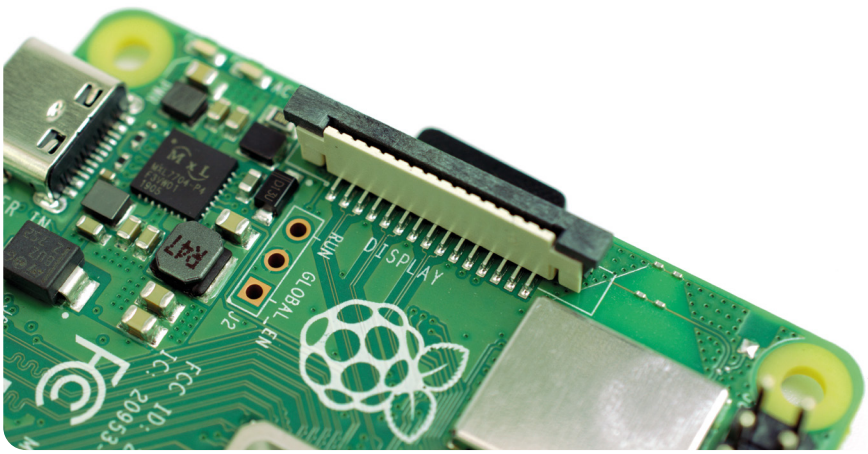



**Bild 1-12:** Raspberry Pi:s bildskärmskontakt (DSI)

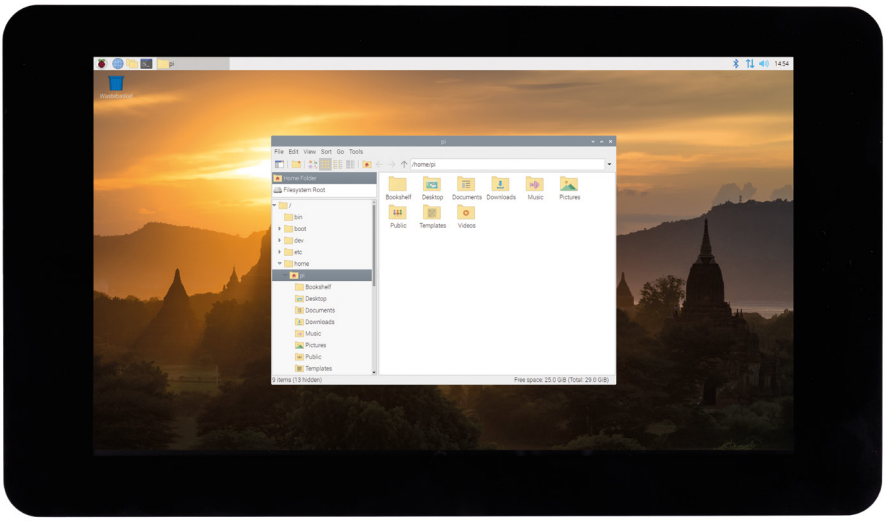


**Bild 1-13:** Raspberry Pi:s pekskärm

Vid kortets högra kant hittar du 40 metallstift, uppdelade på två rader med 20 stift (**Bild 1-14**). Detta är *GPIO (in-/utgång för allmänna syften)*, en funktion i Raspberry Pi som används för att prata med ytterligare maskinvara, från lysdioder och knappar till temperatursensorer, joysticks och pulsfrekvensövervakare. Du lär dig mer om GPIO i **Kapitel 6, Fysisk programmering med Scratch och Python**. Precis nedanför och till vänster om denna finns en annan mindre kontakt med fyra stift. Den används för att ansluta Ström över Ethernet (PoE) HAT, ett tillval som gör det möjligt för Raspberry Pi att få ström från en nätverksanslutning snarare än från USB Type-C-porten.

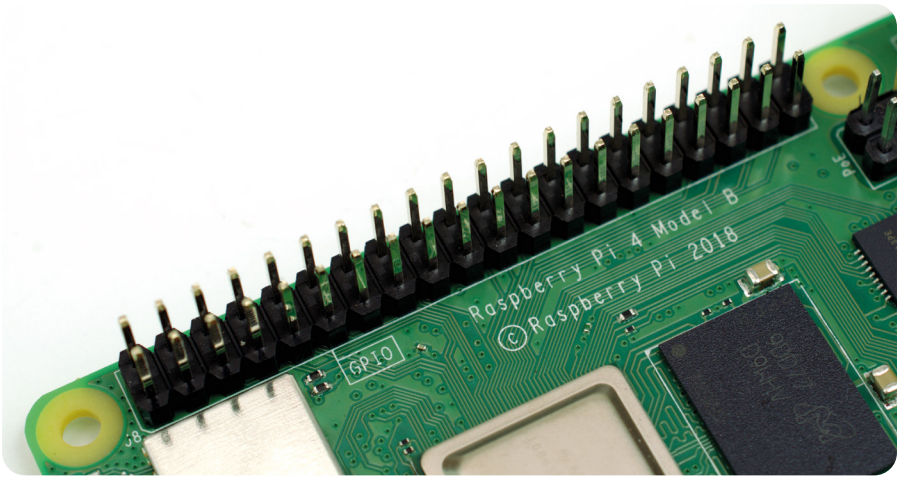



**Bild 1-14:** Raspberry Pi:s GPIO

Det finns en sista port på Raspberry Pi, men du kan inte se den på ovansidan. Vänd på kortet så hittar du en *microSD-kortkontakt* på motsatta sidan av där bildskärmskontakten sitter (**Bild 1-15**). Detta är Raspberry Pi:s lagringsenhet: microSD-kortet som sätts in här innehåller alla filer du sparar, all programvara du installerar och operativsystemet som gör att Raspberry Pi kan fungera.

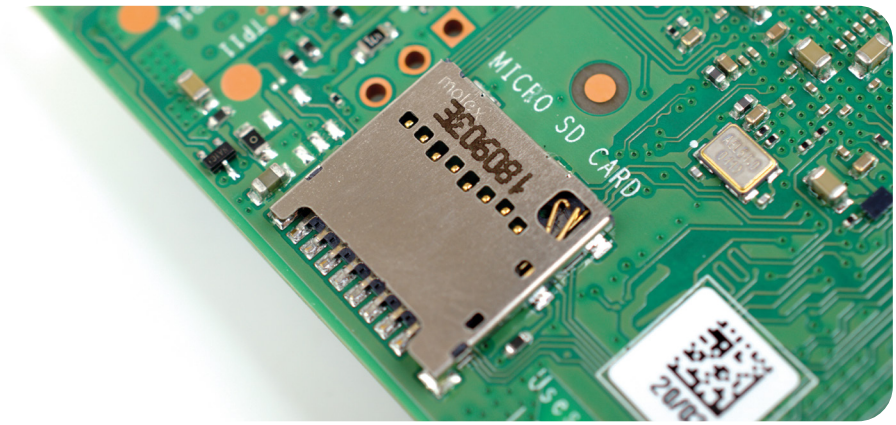



**Bild 1-15:** Raspberry Pi:s microSD-kortkontakt


▲ **Bild 1-16:** Raspberry Pi 400 har ett integrerat tangentbord

## Raspberry Pi 400

Raspberry Pi 400 har samma komponenter som Raspberry Pi 4, men de är placerade inuti ett tangentbordshölje. Förutom att skydda komponenterna tar tangentbordshöljet mindre plats på skrivbordet och hjälper till att hålla ordning på kablarna.

Raspberry Pi 400 består av samma kärnkomponenter som Raspberry Pi 4, inklusive system-på-chipp och minne. Du kan inte se dem, men de är ändå där. Du kan se de utvändiga delarna som börjar med tangentbordet (**Bild 1-16**). Mot det högra hörnet finns tre lysdioder (LED-lampor): den första tänds när du trycker på Num Lock-tangenten, som gör att vissa tangenter fungerar som en sifferknappsats på ett tangentbord i full storlek; den andra tänds när du trycker på Caps Lock, vilket gör bokstavstangenterna till stora bokstäver istället för små; och den sista tänds när Raspberry Pi 400 startas.

På baksidan av Raspberry Pi 400 (**Bild 1-17**) finns portarna. Porten längst till vänster sett bakifrån är GPIO, in-/utgång för allmänna syften. Detta är samma kontakt som beskrivs på sidan 17, men vänd upp och ner: det första stiftet, stift 1, sitter högst upp till höger, medan det sista stiftet, stift 40, sitter längst ner till vänster. Du kan få reda på mer om GPIO i **Kapitel 6, Fysisk programmering med Scratch och Python**.

**Bild 1-17:** Portarna finns på baksidan av Raspberry Pi 400

Bredvid GPIO sitter microSD-kortplatsen. Detta lagrar microSD-kortet som fungerar som lagringsenhet för Raspberry Pi 400 till operativsystemet, applikationer och annan data. MicroSD-kortet levereras förinstallerat i Raspberry Pi 400. Du kan ta bort det genom att trycka försiktigt på kortet tills det klickar och fjädrar ut och sedan dra ut det. När du sätter in kortet igen ska du se till att de blanka metallkontakterna är vända nedåt. Kortet ska glida in med ett försiktigt klick.

De två portarna bredvid är micro-HDMI-portarna för anslutning till en bildskärm, tv eller annan visningsenhet. Precis som Raspberry Pi 4 har Raspberry Pi 400 stöd för upp till två skärmar. Bredvid dessa sitter USB Type-C-strömporten för anslutning till Raspberry Pi:s strömförsörjning eller kompatibel USB-strömförsörjning.

De två blå portarna är USB 3.0-portar, som ger en höghastighetsanslutning till enheter, inklusive halvledarenheter (SSD), minnesstickor, skrivare med mera. Den vita porten till höger om dessa är en USB 2.0-port med lägre hastighet. Du kan ansluta den medföljande Raspberry Pi-musen till denna.

Den sista porten är en Gigabit Ethernet-nätverksport som gör att du kan ansluta Raspberry Pi 400 till ditt nätverk med en nätverkskabel, som ett alternativ till att använda det inbyggda trådlösa Wi-Fi-nätverket. Du kan läsa mer om att ansluta Raspberry Pi 400 till ett nätverk i **Kapitel 2, Kom igång med din Raspberry Pi**.

## Kapitel 2

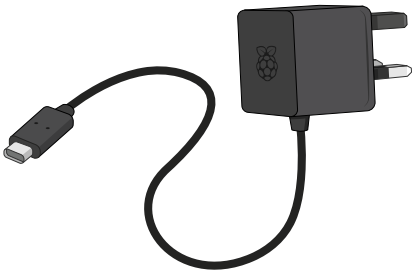
# Kom igång med Raspberry Pi

### Upptäck de viktiga sakerna du behöver till Raspberry Pi och hur du ansluter dem för att få den att fungera

Raspberry Pi har utformats för att vara så snabb och enkel att installera och använda som möjligt, men – som alla datorer – är den beroende av olika externa komponenter, så kallad *kringutrustning*. Det är lätt hänt att man tar en titt på Raspberry Pi:s nakna kretskort – som ser väldigt annorlunda ut än de inneslutna, stängda datorer du kanske är van vid – och blir orolig för att saker och ting är på väg att bli komplicerade. Men så är inte fallet. Du kan komma igång med Raspberry Pi på under tio minuter genom att helt enkelt följa stegen i den här guiden.

Om du har fått den här boken i ett Raspberry Pi Desktop Kit eller med en Raspberry Pi 400 har du redan nästan allt du behöver för att komma igång: allt du behöver själv är en datorskärm eller tv med en HDMI-anslutning – samma typ av kontakt som används av digitalboxar, Blu-ray-spelare och spelkonsoler – så att du kan se vad Raspberry Pi gör.

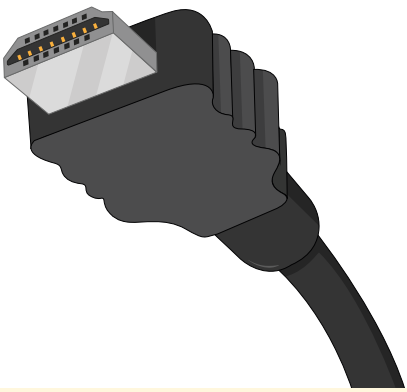
Om du köpte Raspberry Pi utan tillbehör behöver du även:



- **USB-strömförsörjning** – En 5 V-strömförsörjning med strömstyrka på 3 ampere (3 A) och med en USB Type-C-kontakt. Den officiella Raspberry Pi-strömförsörjningen är det rekommenderade valet, eftersom det klarar Raspberry Pi:s snabbt växlande energibehov.

- **MicroSD-kort med NOOBS** – MicroSD-kortet fungerar som Raspberry Pi:s permanenta lagringsenhet. Alla filer du skapar och all programvara du installerar, tillsammans med själva operativsystemet, lagras på kortet. Ett 8 GB-kort hjälper dig att komma igång, men ett kort på 16 GB ger mer utrymme att växa. Att använda ett kort med NOOBS (New Out-of-Box Software) förinstallerat sparar tid. Se annars **Bilaga A** för anvisningar om hur du installerar ett operativsystem (OS) på ett tomt kort.

- **USB-tangentbord och mus** – Med tangentbordet och musen kan du styra Raspberry Pi. Nästan alla trådbundna eller trådlösa tangentbord och möss med en USB-kontakt kommer att fungera med Raspberry Pi, även om vissa tangentbord i ”gaming”-stil med färgglada lampor kan dra för mycket ström för att kunna användas på ett tillförlitligt sätt.



- **Micro-HDMI-kabel** – Denna överför ljud och bilder från Raspberry Pi till tv:n eller bildskärmen. Den ena änden av kabeln har en micro-HDMI-kontakt för Raspberry Pi, den andra en fullstor HDMI-kontakt för skärmen. Eller så kan du använda en micro-HDMI till HDMI-adapter och en standard-HDMI-kabel i full storlek. Om du använder en bildskärm utan HDMI-uttag kan du köpa micro-HDMI till DVI-D-, DisplayPort- eller VGA-adaptrar. Om du ska ansluta till en äldre tv som har ett kompositvideo- eller SCART-uttag ska du använda en 3,5 mm TRRS-kabel för ljud/video (tip-ring-ring-sleeve).

Raspberry Pi är säker att använda utan hölje, förutsatt att du inte placerar den på en metallyta som kan leda elektricitet och orsaka kortslutning. Tillvalshöljet kan dock ge ytterligare skydd. Desktop Kit innehåller det officiella Raspberry Pi-höljet, medan höljen från tredje part finns tillgängliga hos alla välsorterade återförsäljare.

Om du vill använda Raspberry Pi i ett trådbundet nätverk, istället för ett trådlöst (WiFi) nätverk, behöver du även en nätverkskabel. Denna ska anslutas i ena änden till nätverkets switch eller router. Om du planerar att använda Raspberry Pi:s inbyggda trådlösa radio behöver du inte någon kabel. Du måste dock veta namnet och nyckeln eller lösenfrasen till ditt trådlösa nätverk.

### KONFIGURATION AV RASPBERRY PI 400

Följande anvisningar är till för att konfigurera Raspberry Pi 4 eller en annan medlem i Raspberry Pi-familjen med naket kretskort. Anvisningar för konfiguration av Raspberry Pi 400 finns på sidan 32.

## Konfigurera maskinvaran

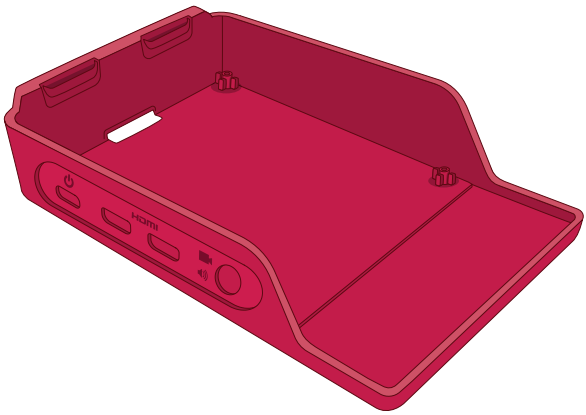
Börja med att packa upp Raspberry Pi ur lådan. Raspberry Pi är en robust maskinvara, men det betyder inte att den är oförstörbar: försök att vänja dig vid att hålla kretskortet i kanterna, istället för på dess platta sidor, och var extra försiktig med de upphöjda metallstiften. Om dessa stift böjs kommer det i bästa fall att göra det svårt att använda tilläggskort och annan extra maskinvara, och i värsta fall kan det orsaka en kortslutning som skadar Raspberry Pi.

Om du inte redan har gjort det, ta en titt i **Kapitel 1, Lär känna Raspberry Pi**, för att få information om exakt var de olika portarna finns och vad de gör.

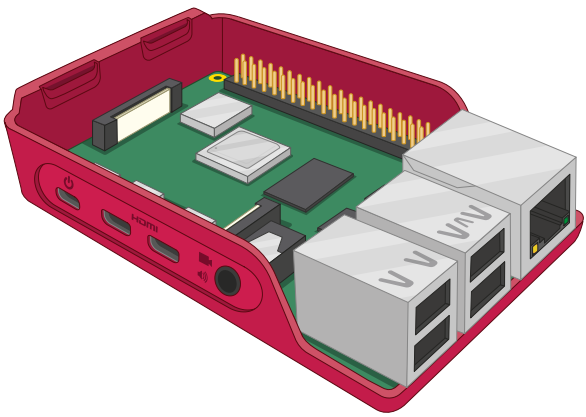
## Montera höljet

Om du installerar Raspberry Pi i ett höljde ska det vara ditt första steg. Om du använder det officiella Raspberry Pi-höljet ska du börja med att dela det i de två enskilda delarna: den röda basen och det vita locket.

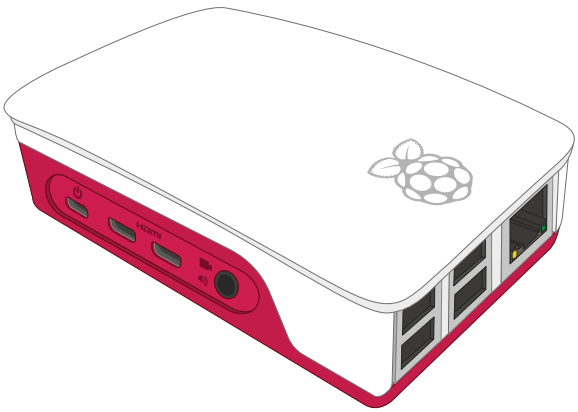
**1** **Ta upp basen och håll den så att den upphöjda änden är till vänster och den lägre änden till höger.**



**2** **Håll Raspberry Pi (utan microSD-kort isatt) i USB- och Ethernet-portarna, lätt vinklad, och för in kontakterna (USB Type-C, 2 × micro-HDMI och 3,5 mm) i hålen på sidan av basen och sänk sedan ner den andra sidan så att den står plant.**

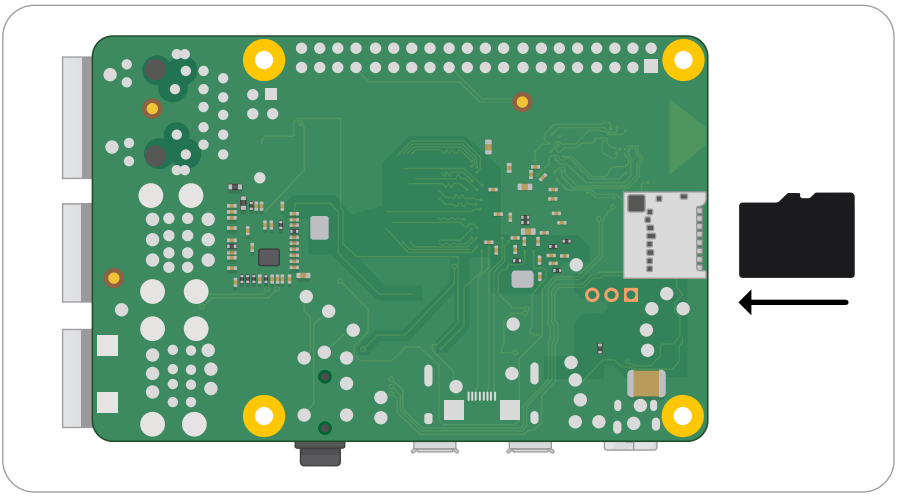


**3** **Ta det vita locket och placera de två clipsen på vänster sida i matchande hål till vänster på basen, ovanför microSD-kortplatsen. När de är på plats trycker du ner höger sida (ovanför USB-portarna) tills du hör ett klick.**

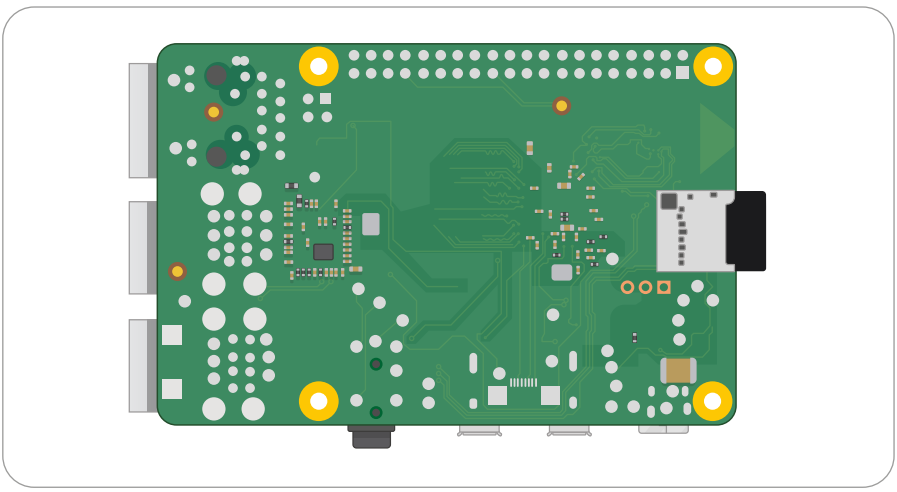


## Ansluta microSD-kortet

För att installera microSD-kortet, som är Raspberry Pi:s *lagringsenhet*, vänder du Raspberry Pi (i höljet om du använder ett sådant) och för in kortet i microSD-kortplatsen med etiketten vänd bort från Raspberry Pi. Det går bara att få in kortet på ett sätt och det bör glida på plats utan särskilt hårt tryck.



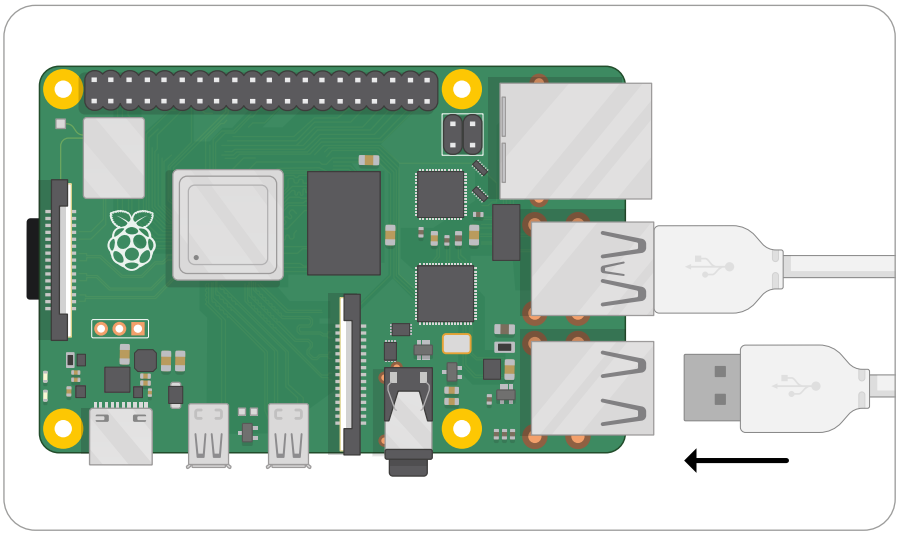
MicroSD-kortet glider in i kontakten och stannar sedan utan att klicka.



Om du vill ta bort det igen i framtiden ska du bara ta tag i kortets ände och försiktigt dra ut det. Om du använder en äldre modell av Raspberry Pi måste du först ge kortet ett försiktigt tryck för att låsa upp det. Detta är inte nödvändigt med Raspberry Pi 3 eller 4.

## Ansluta tangentbord och mus

Anslut tangentbordets USB-kabel till någon av de fyra USB-portarna (2.0 eller 3.0) på Raspberry Pi. Om du använder det officiella Raspberry Pi-tangentbordet finns det en USB-port på baksidan för musen. Om inte, anslut bara musens USB-kabel till en annan USB-port på Raspberry Pi.



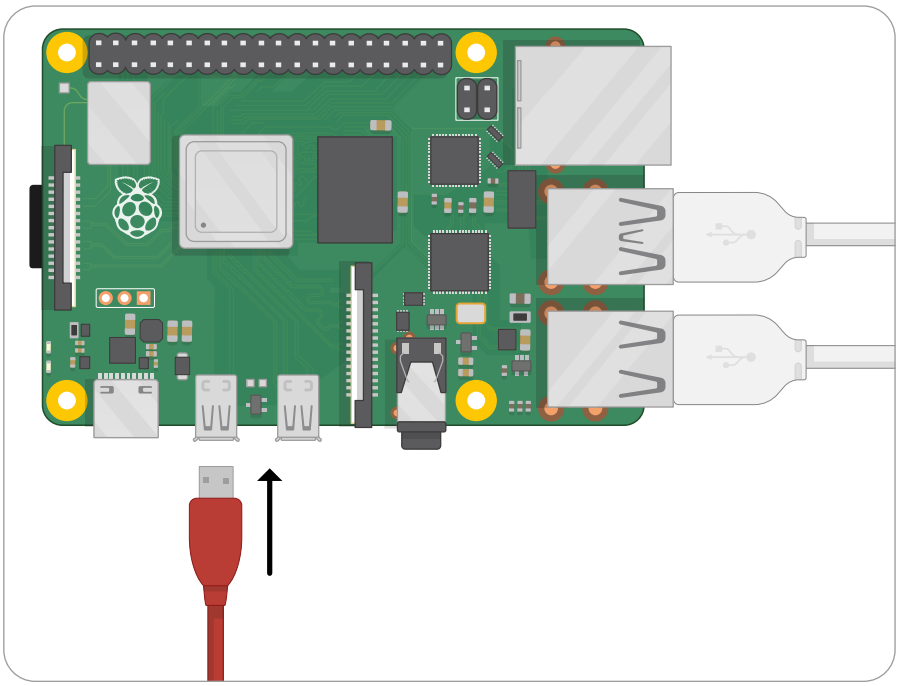
USB-kontakterna till tangentbordet och musen ska glida in utan alltför mycket tryck. Om du behöver tvinga in kontakten är något fel. Kontrollera att USB-kontakten är vänd på rätt håll!

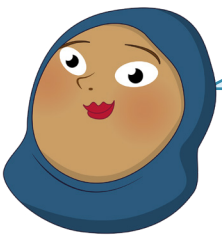
### TANGENTBORD OCH MUS

Tangentbordet och musen fungerar som det huvudsakliga sättet att berätta för Raspberry Pi vad som ska göras. De benämns som *inmatningsenheter*, i motsats till skärmen som är en *utmatningsenhet*.

## Ansluta en skärm

Ta micro-HDMI-kabeln och anslut den mindre änden till micro-HDMI-porten närmast USB Type-C-porten på Raspberry Pi och den andra änden till skärmen. Om skärmen har mer än en HDMI-port ska du leta efter ett portnummer bredvid själva kontakten. Du måste byta ingång på tv:n till detta för att kunna se Raspberry Pi:s skärm. Om du inte ser något portnummer ska du inte oroa dig: du kan helt enkelt växla genom ingångarna i tur och ordning tills du hittar Raspberry Pi.



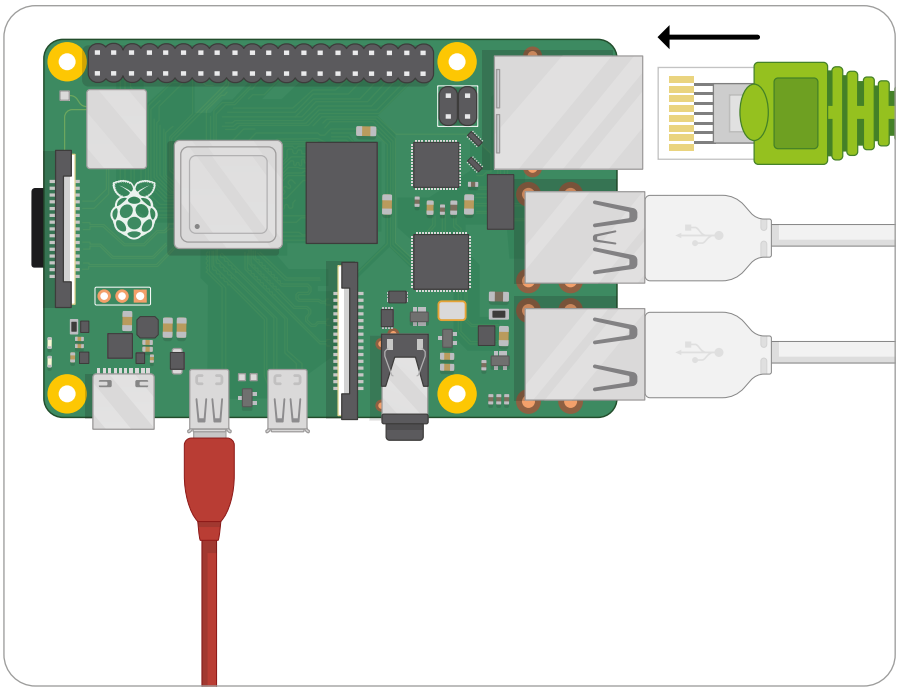


### TV-ANSLUTNING

Att tv:n eller bildskärmen inte har någon HDMI-kontakt innebär inte att du inte kan använda Raspberry Pi. Adapterkablar, som finns hos alla elektronikhandlare, gör att du kan konvertera micro-HDMI-porten på Raspberry Pi till DVI-D, DisplayPort eller VGA som finns på äldre datorskärmar. Dessa ansluts helt enkelt till Raspberry Pi:s micro-HDMI-port, och sedan används en lämplig kabel för att ansluta adapterkabeln till bildskärmen. Om tv:n bara har en kompositvideo- eller SCART-ingång, kan du köpa 3,5 mm TRRS-adapterkabel (tip-ring-ring-sleeve) och komposit-till-SCART-adapter som går att ansluta till 3,5 mm AV-uttaget.

## Ansluta en nätverkskabel (tillval)

För att ansluta Raspberry Pi till ett trådbundet nätverk ska du ta en nätverkskabel – en så kallad Ethernet-kabel – och trycka in den i Raspberry Pi:s Ethernet-port, med plastklämman nedåt, tills du hör ett klick. Om du behöver ta bort kabeln är det bara att trycka plastklämman inåt mot kontakten och försiktigt dra loss kabeln.

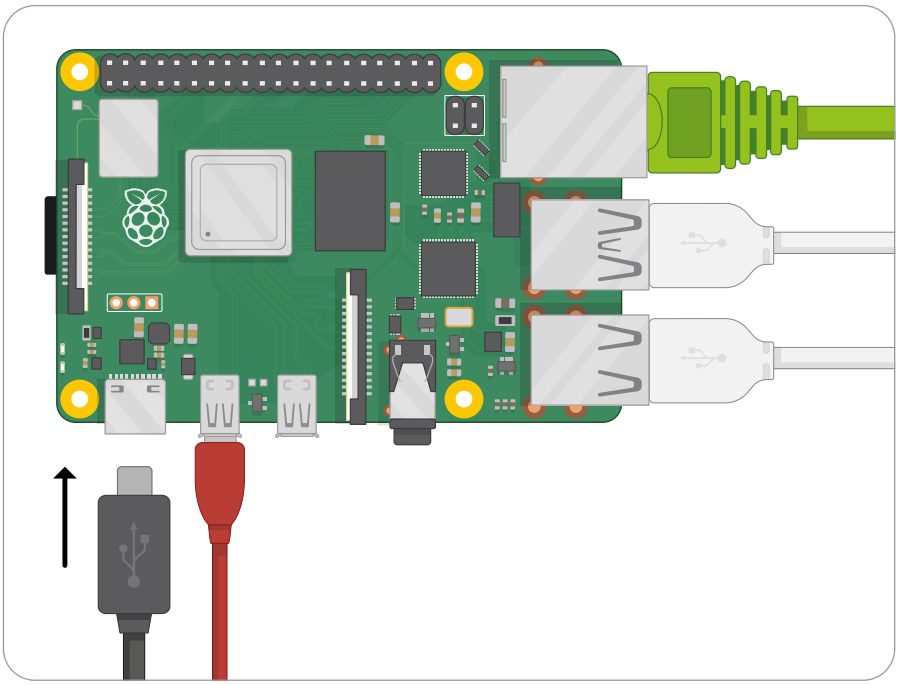


Den andra änden av nätverkskabeln ska anslutas till valfri ledig port på nätverkshubben, switchen eller routern på samma sätt.

## Ansluta en strömförsörjning

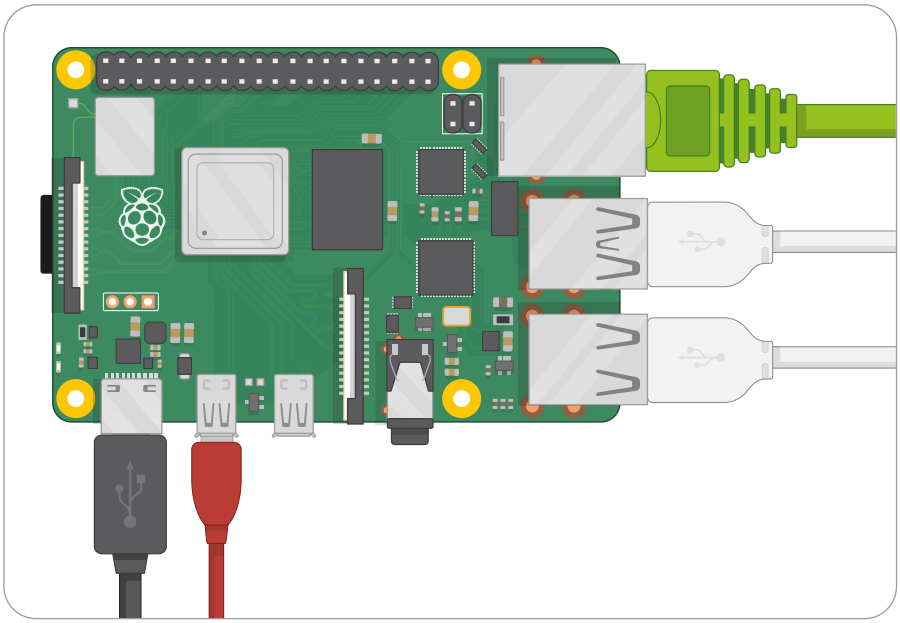
Att ansluta Raspberry Pi till en strömförsörjning är det allra sista steget i maskinvaruinstallationsprocessen, och det bör du bara göra när du är redo att installera programvaran: Raspberry Pi har ingen strömbrytare och aktiveras så snart den ansluts till en strömförsörjning.

Anslut först USB Type-C-änden på strömförsörjningskabeln till USB Type-C-strömkontakten på Raspberry Pi. Den passar på alla håll och bör glida in smidigt. Om strömförsörjningen har en löstagbar kabel, se till att den andra änden är ansluten till strömförsörjningens hölje.



Slutligen ska du ansluta strömförsörjningen till ett eluttag och slå på uttaget. Raspberry Pi startar då omedelbart.

Grattis, du har satt ihop Raspberry Pi!



Du kommer kort att se fyra Raspberry Pi-logotyper längst upp till vänster på en svart skärm och sedan kan du komma att se en blå skärm visas när programvaran ändrar sig själv för att utnyttja microSD-kortet till fullo. Vänta några minuter om du ser en svart skärm. Första gången Raspberry Pi startar upp måste den städa lite i bakgrunden. Efter ett tag ser du Raspberry Pi OS-skrivbordet och installationsguiden, som på **Bild 2-1**. Operativsystemet är nu redo att konfigureras, vilket du lär dig att göra i **Kapitel 3, Använda Raspberry Pi**.

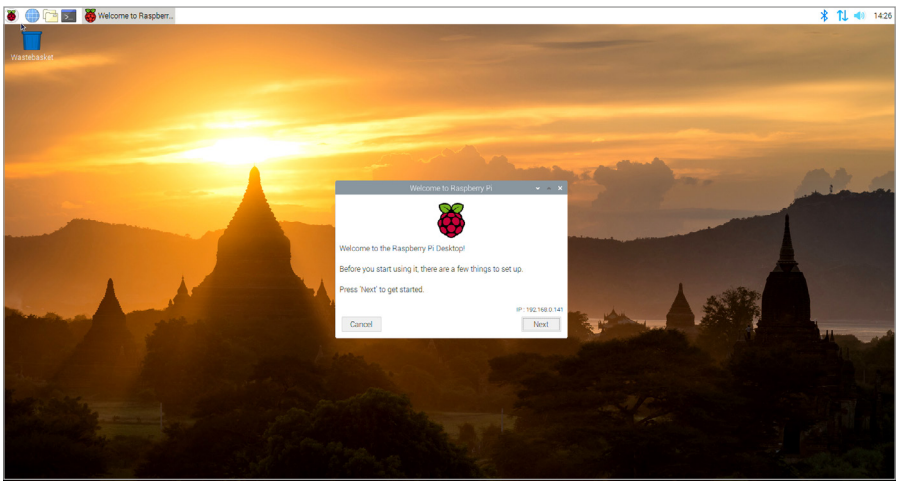


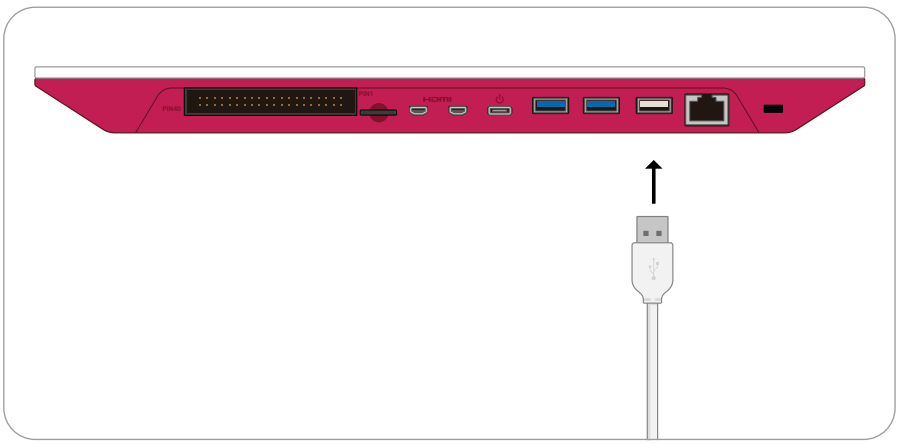

**Bild 2-1:** Raspberry Pi OS-skrivbordet och installationsguiden

# Konfiguration av Raspberry Pi 400

Till skillnad från Raspberry Pi 4 levereras Raspberry Pi 400 med ett inbyggt tangentbord och redan installerat microSD-kort. Du måste fortfarande ansluta några kablar för att komma igång, men det tar bara några minuter.

## Ansluta en mus

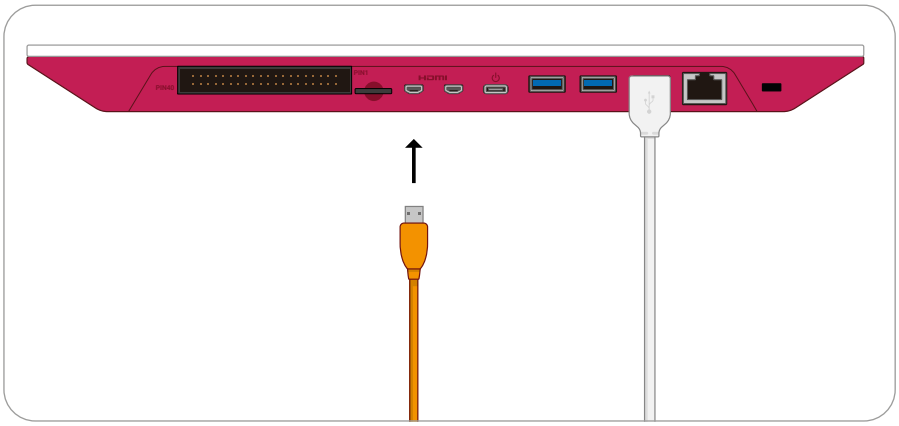
Raspberry Pi 400:s tangentbord är redan anslutet, så du har bara musen kvar att lägga till. Ta USB-kabeln i änden av musen och sätt i den i någon av de tre USB-portarna (2.0 eller 3.0) på baksidan av Raspberry Pi 400. Om du vill spara de två höghastighetsportarna, USB 3.0, till andra tillbehör ska du använda den vita porten.



USB-kontakten ska glida in utan alltför mycket tryck. Om du behöver tvinga in kontakten är något fel. Kontrollera att USB-kontakten är vänd på rätt håll!

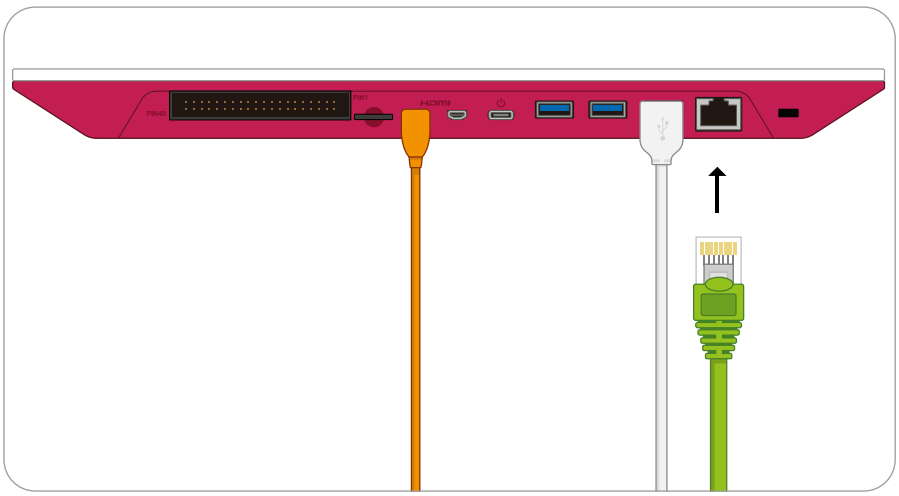
## Ansluta en skärm

Ta micro-HDMI-kabeln och anslut den mindre änden till micro-HDMI-porten närmast microSD-platsen på Raspberry Pi 400 och den andra änden till skärmen. Om skärmen har mer än en HDMI-port ska du leta efter ett portnummer bredvid själva kontakten. Du måste byta ingång på tv:n till detta för att kunna se Raspberry Pi:s skärm. Om du inte ser något portnummer ska du inte oroa dig: du kan helt enkelt växla genom ingångarna i tur och ordning tills du hittar Raspberry Pi.



## Ansluta en nätverkskabel (tillval)

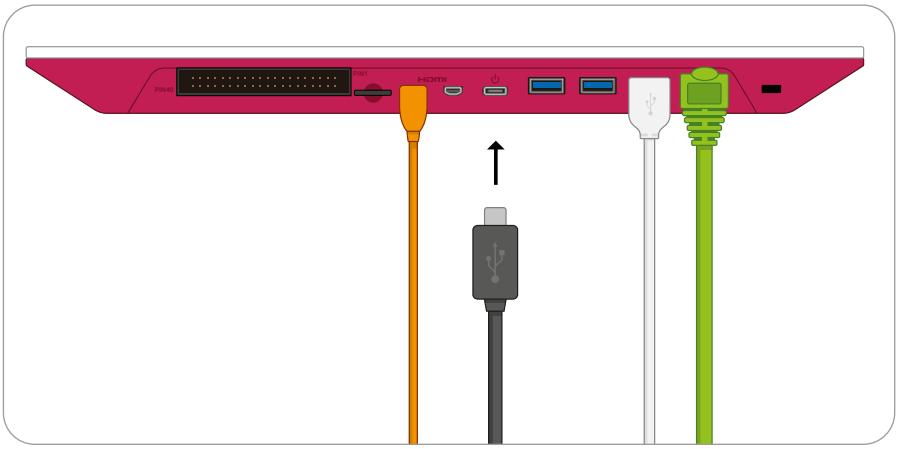
För att ansluta Raspberry Pi 400 till ett trådbundet nätverk ska du ta en nätverkskabel – en så kallad Ethernet-kabel – och trycka in den i Raspberry Pi 400:s Ethernet-port, med plastklämman nedåt, tills du hör ett klick. Om du behöver ta bort kabeln är det bara att trycka plastklämman inåt mot kontakten och försiktigt dra loss kabeln.



Den andra änden av nätverkskabeln ska anslutas till valfri ledig port på nätverkshubben, switchen eller routern på samma sätt.

## Ansluta en strömförsörjning

Att ansluta Raspberry Pi 400 till en strömförsörjning är det allra sista steget i maskinvaruinstallationsprocessen, och det bör du bara göra när du är redo att installera programvaran: Raspberry Pi 400 har ingen strömbrytare och aktiveras så snart den ansluts till en strömförsörjning. Anslut först USB Type-C-änden på strömförsörjningskabeln till USB Type-C-strömkontakten på Raspberry Pi. Den passar på alla håll och bör glida in smidigt. Om strömförsörjningen har en löstagbar kabel, se till att den andra änden är ansluten till strömförsörjningens hölje.



Slutligen ska du ansluta strömförsörjningen till ett eluttag och slå på uttaget. Raspberry Pi 400 startar då omedelbart. Grattis, du har satt ihop Raspberry Pi 400!

Du kommer kort att se fyra Raspberry Pi-logotyper längst upp till vänster på en svart skärm och sedan kan du komma att se en blå skärm visas när programvaran ändrar sig själv för att utnyttja microSD-kortet till fullo. Vänta några minuter om du ser en svart skärm. Första gången Raspberry Pi startar upp måste den städa lite i bakgrunden. Efter ett tag ser du Raspberry Pi OS-skrivbordet och installationsguiden, som på **Bild 2-2**. Operativsystemet är nu redo att konfigureras, vilket du lär dig att göra i **Kapitel 3, Använda Raspberry Pi**.

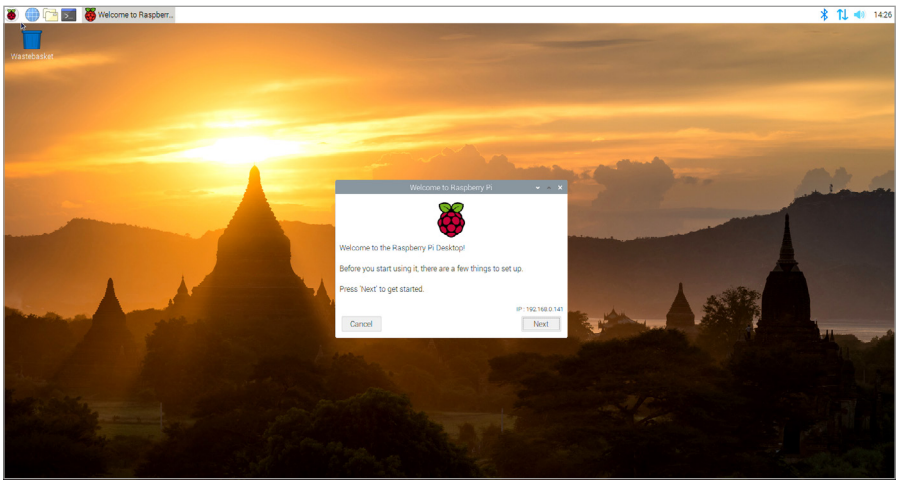
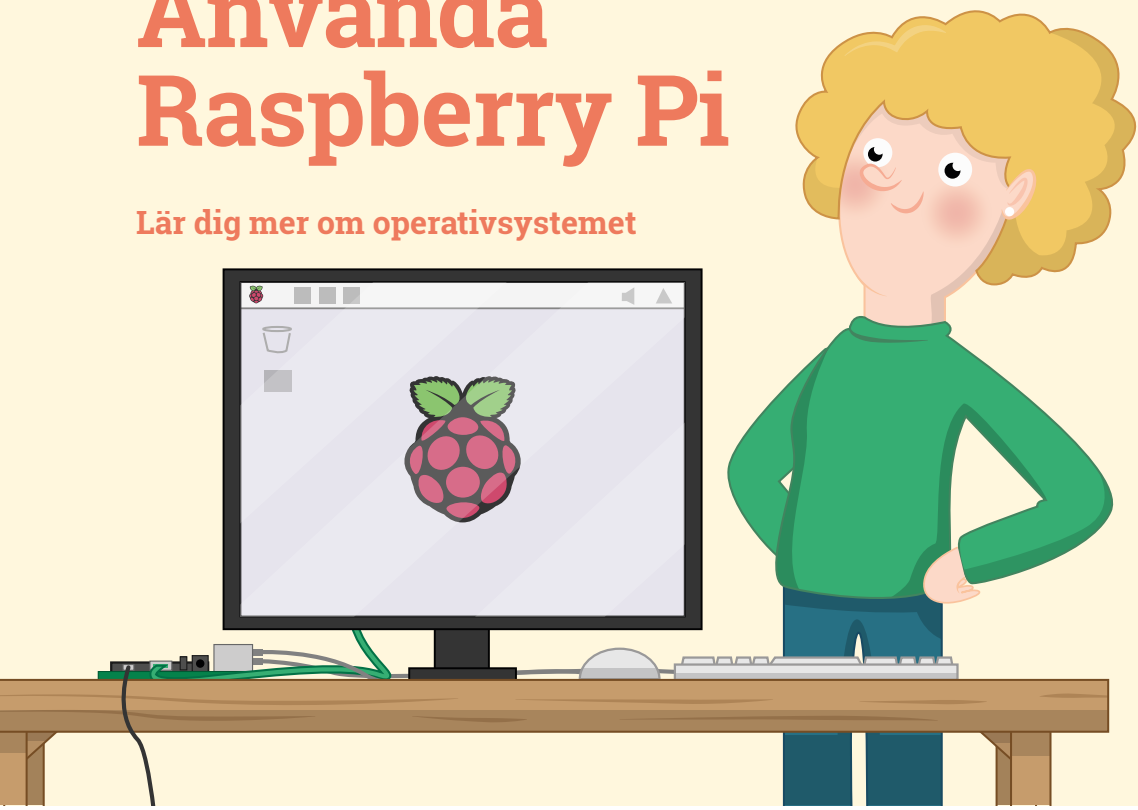



**Bild 2-2:** Raspberry Pi OS-skrivbordet och installationsguiden

Kapitel 3

# Använda Raspberry Pi

## Lär dig mer om operativsystemet

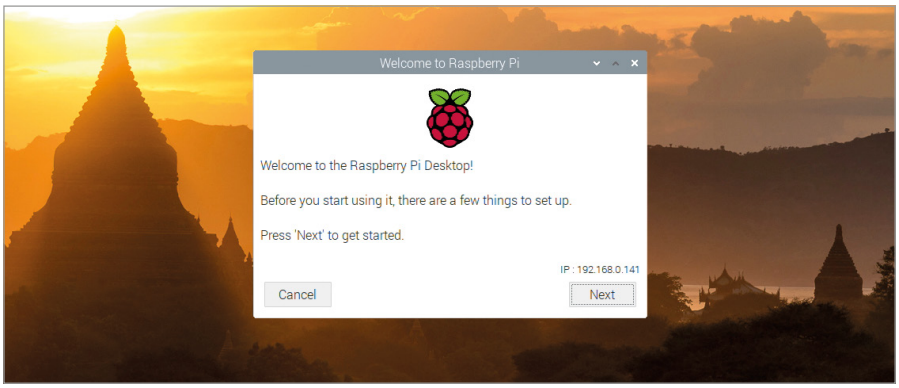


Raspberry Pi kan köra ett brett utbud av programvara, inklusive ett antal olika operativsystem – kärnprogramvaran som får en dator att fungera. Det mest populära av dessa, och det officiella operativsystemet för Raspberry Pi Foundation, är Raspberry Pi OS. Det är baserat på Debian Linux och är skräddarsytt för Raspberry Pi. Det levereras med en rad förinstallerade extrafunktioner och är klart att köra.

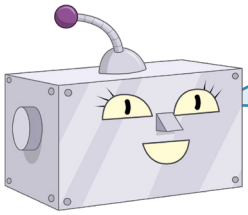
Om du bara har använt Microsoft Windows eller Apple macOS ska du inte oroa dig: Raspberry Pi OS är baserat på samma fönster, ikoner, menyer och pekprinciper (WIMP) och borde snabbt kännas bekant. Följande kapitel hjälper dig att komma igång och introducerar dig för några av programmen som ingår.

## Välkomstguiden

Första gången du kör Raspberry Pi OS tas du emot av välkomstguiden (**bild 3-1**). Det användbara verktyget visar dig hur du ska ändra vissa inställningar i Raspberry Pi OS, vilketd kallas för *konfiguration*, så att de stämmer överens med hur och var du ska använda Raspberry



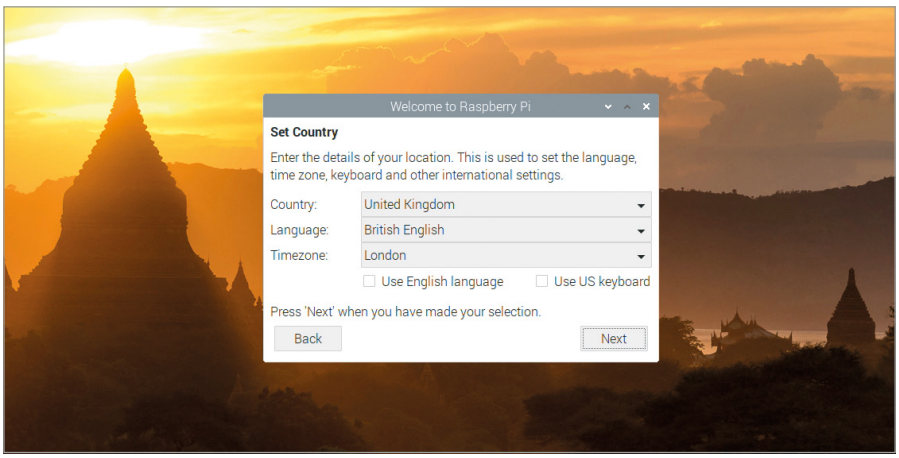

▲ **Bild 3-1:** Välkomstguiden



### STÄNGA GUIDEN

Du kan välja att stänga välkomstguiden genom att klicka på knappen Avbryt, men vissa Raspberry Pi-funktioner, som det trådlösa nätverket, fungerar inte förrän du svarar på den första uppsättningen frågor.

Klicka på knappen Next och välj sedan land, språk och tidszon genom att klicka på varje listruta i tur och ordning och välja ditt svar från listan (**bild 3-2**). Om du använder ett tangentbord med amerikansk layout ska du klicka på kryssrutan för att se till att Raspberry Pi OS använder rätt tangentbordslayout. Om du vill att skrivbordet och programmen ska visas på engelska, oavsett ditt lands språk, klickar du på kryssrutan ”Use English language”. När du är klar klickar du på Next.




▲ **Bild 3-2:** Att välja språk, bland andra alternativ

På nästa skärm blir du ombedd att ändra lösenordet för ”pi”-användaren (från standardinställningen ”raspberry”). Av säkerhetsskäl är det en mycket bra idé att skapa ett nytt lösenord. Skriv in det i rutorna (**bild 3-3**). När du är nöjd klickar du på Next.

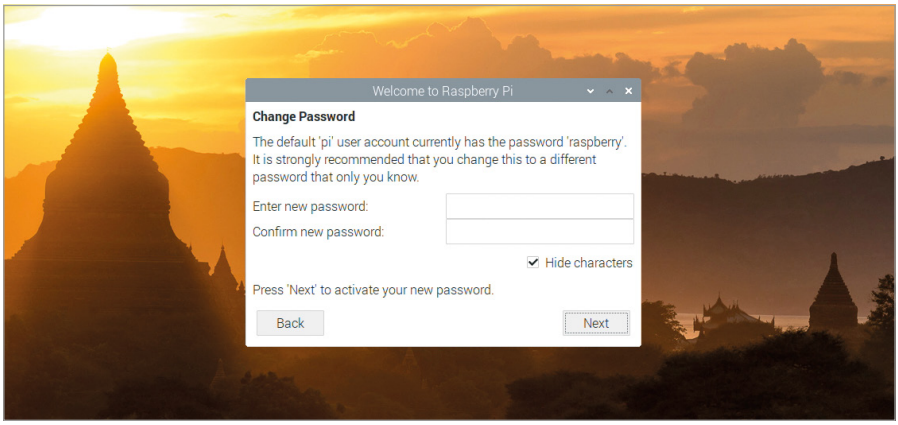



**Bild 3-3:** Ställa in ett nytt lösenord

På nästa skärm blir du tillfrågad om det finns en svart marginal runt skärmens kant (**bild 3-4**). Om Raspberry Pi-skrivbordet fyller hela tv:n eller bildskärmen ska du lämna rutan omarkerad. Om den har en svart marginal runt och är mindre än tv:n eller bildskärmen ska du kryssa i rutan. När du är redo att gå vidare klickar du på Next.

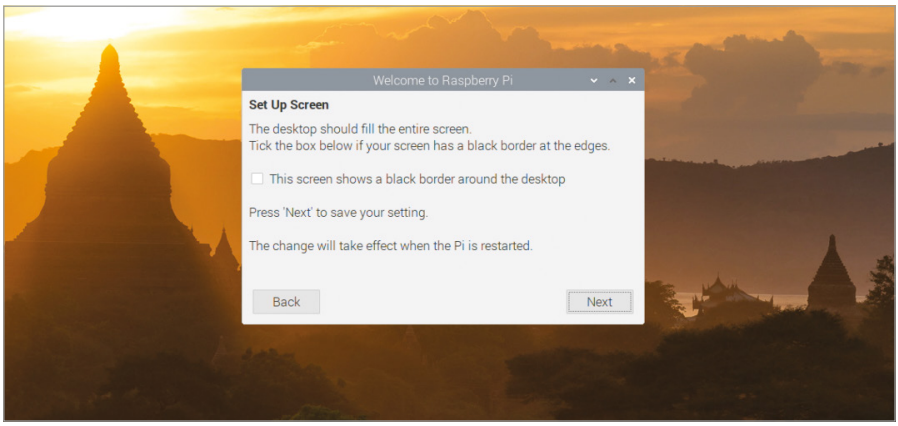
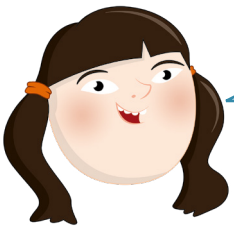



**Bild 3-4:** Kontrollerar att det inte finns någon svart marginal

På följande skärm kan du välja Wi-Fi-nätverket från en lista (**bild 3-5**). Bläddra igenom listan över nätverk med musen eller tangentbordet, hitta ditt nätverksnamn, klicka på det och klicka sedan på Next. Förutsatt att ditt trådlösa nätverk är säkert (det borde det verkligen vara) kommer du att bli ombedd att ange lösenordet till det (kallas även dess fördelade nyckel). Det



### TRÅDLÖST NÄTVERK

Inbyggt trådlöst nätverk är endast tillgängligt i familjerna Raspberry Pi 3, Pi 4 och Pi Zero W. Om du vill använda en annan modell av Raspberry Pi med ett trådlöst nätverk behöver du en USB Wi-Fi-adapter.

står oftast på ett kort som följer med routern eller på själva routerns undersida. Klicka på Next för att ansluta till nätverket. Om du inte vill ansluta till ett trådlöst nätverk klickar du på Skip.

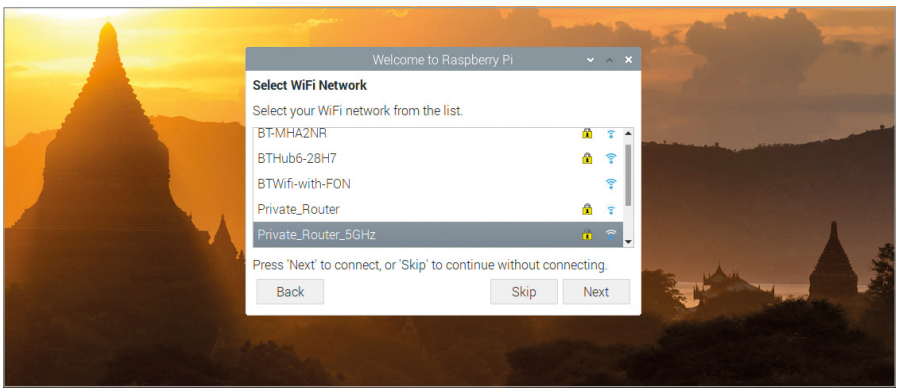



**Bild 3-5:** Välja ett trådlöst nätverk

Nästa skärm gör det möjligt för dig att söka efter och installera uppdateringar för Raspberry Pi OS och övrig programvara på Raspberry Pi (**bild 3-6**). Raspberry Pi OS uppdateras regelbundet för att korrigera buggar, lägga till nya funktioner och förbättra prestandan. Klicka på Next för att installera dessa uppdateringar. Annars klickar du på Skip. Nedladdningen av uppdateringarna kan ta flera minuter, så ha tålamod. När uppdateringarna är installerade visas ett fönster med texten ”System is up to date”. Klicka på knappen OK.

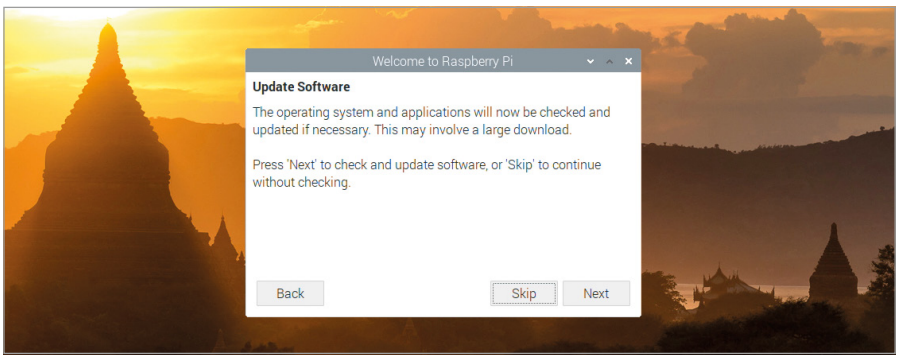
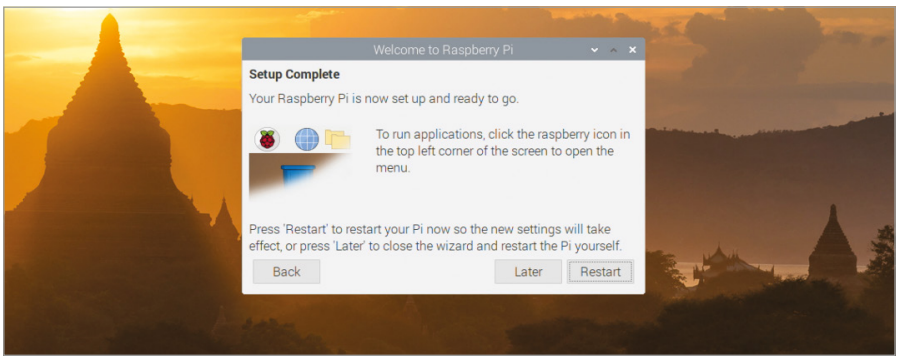



**Bild 3-6:** Letar efter uppdateringar

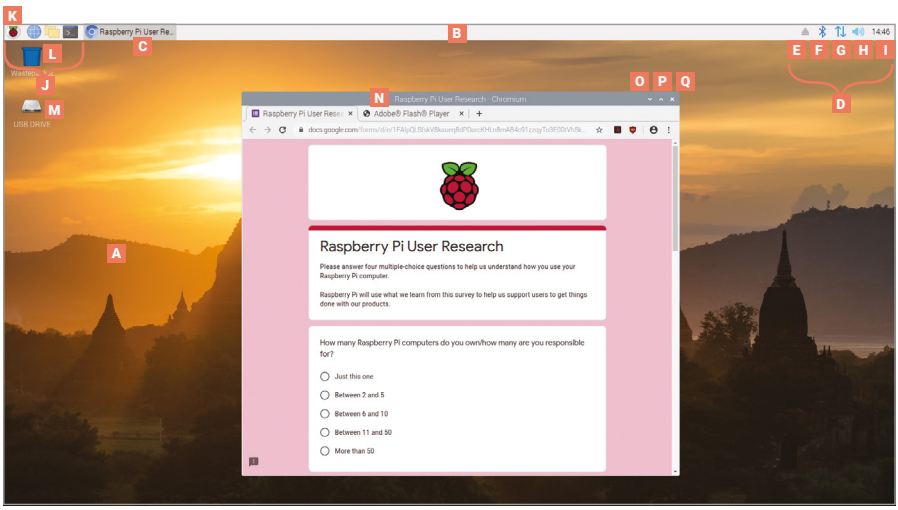
Den sista skärmen i välkomstguiden (**bild 3-7**) har en enkel uppgift att genomföra: vissa ändringar som görs träder i kraft först när du startar om Raspberry Pi (en process som också kallas omstart, eller ”reboot”). Om du uppmanas att göra det klickar du på knappen Restart så startas Raspberry Pi om. Den här gången visas inte välkomstguiden. Det arbetet är klart, och Raspberry Pi är nu redo att användas.




▲ **Bild 3-7:** Starta om Raspberry Pi

# Navigera på skrivbordet

Den version av Raspberry Pi OS som finns installerad på de flesta Raspberry Pi-kretskort är känd som ”Raspberry Pi OS with desktop”, vilket syftar på dess huvudsakliga grafiska användargränssnitt (**bild 3-8**). Huvuddelen av skrivbordet tas upp av en bild som kallas skrivbordsbakgrund (**A** på **bild 3-8**), på vilken programmen du kör visas. Högst upp på skrivbordet finns ett aktivitetsfält (**B**) som gör det möjligt för dig att faktiskt läsa in vart och ett av programmen. Dessa indikeras sedan av uppgifter (**C**) i aktivitetsfältet.




▲ **Bild 3-8:** Raspberry Pi OS-skrivbordet

- **A** Skrivbordsbakgrund
- **B** Aktivitetsfält
- **C** Uppgift
- **D** Systemfält
- **E** Mata ut media
- **F** Bluetooth-ikon
- **G** Nätverksikon
- **H** Volymikon
- **I** Klocka
- **J** Startprogram
- **K** Menyikon (eller hallonikon)
- **L** Papperskorgsikon
- **M** Ikon för borttagbar enhet
- **N** Fönstrets titelfält
- **O** Minimera
- **P** Maximera
- **Q** Stäng

På menyradens högra sida finns *systemfältet* (**D**). Om du har några *flyttbara lagringsutrymmen*, till exempel USB-minnen, anslutna till Raspberry Pi ser du en utmatningssymbol (**E**). Genom att klicka på den kan du mata ut och ta bort dem på ett säkert sätt. Längst till höger finns klockan (**I**). Klicka på den för att få fram en digital kalender (**bild 3-9**).

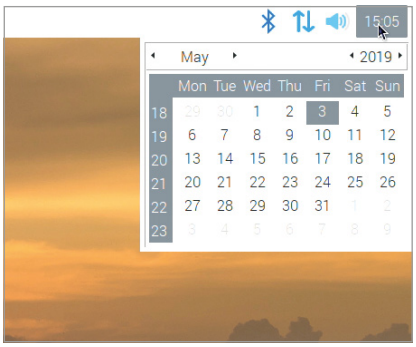



**Bild 3-9:** Den digitala kalendern

Bredvid den finns en högtalarikon (**H**). Klicka på den med vänster musknapp för att justera Raspberry Pi:s ljudvolym eller klicka med höger musknapp för att välja vilken utgång Raspberry Pi ska använda. Bredvid den finns en nätverksikon (**G**). Om du är ansluten till ett trådlöst nätverk ser du signalstyrkan som en serie staplar. Om du är ansluten till ett trådbundet nätverk ser du bara två pilar. Genom att klicka på nätverksikonen får du fram en lista över närliggande trådlösa nätverk (**bild 3-10**). Om du klickar på Bluetooth-ikonen (**F**) bredvid den kan du kan ansluta till en närliggande Bluetooth-enhet.

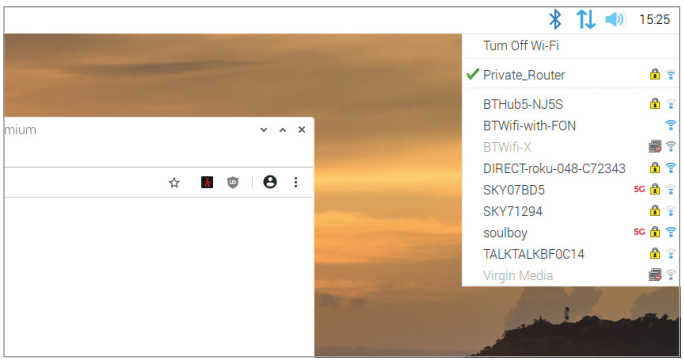
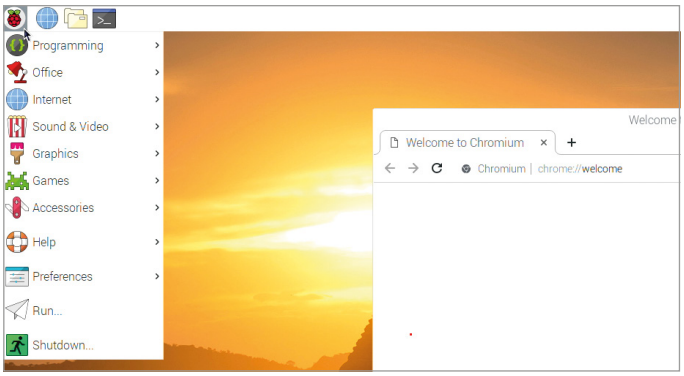



**Bild 3-10:** Listar närliggande trådlösa nätverk

På vänster sida av menyraden finns *startprogrammet* (**J**). Där hittar du programmen som finns installerade tillsammans med Raspberry Pi OS. Några av dessa är synliga som genvägsikoner, andra är dolda i menyn som du kan ta fram genom att klicka på hallonikonen (**K**) längst till vänster (**bild 3-11**, på nästa sida).

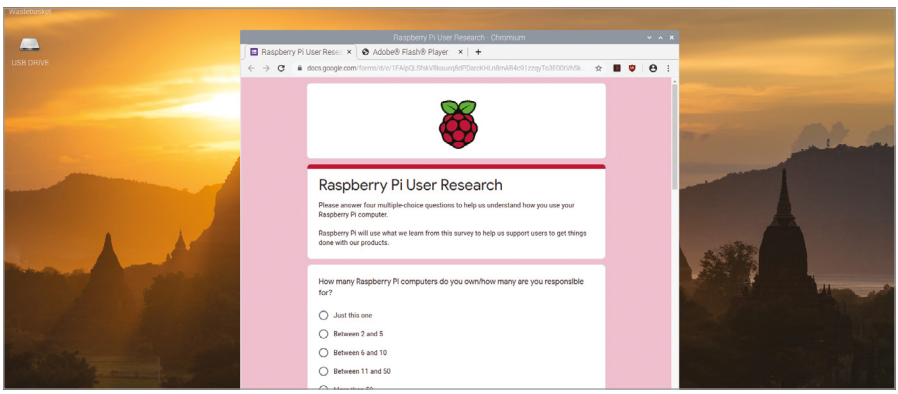



▲ **Bild 3-11:** Raspberry Pi OS-menyn

Programmen i menyn är uppdelade i kategorier vars namn berättar vad du kan förvänta dig: programmeringskategorin innehåller till exempel programvara som är utformad för att hjälpa dig att skriva egna program – vilket förklaras med början i **kapitel 4, Programmering med Scratch** – medan Spel hjälper dig att slå ihjäl några timmar. Alla program beskrivs inte i detalj i guiden. Experimentera gärna med dem för att lära dig mer.

# Webbläsaren Chromium

För att öva på att använda Raspberry Pi kan du börja med att läsa in Chromium-webbläsaren: klicka på hallonikonen längst upp till vänster för att öppna menyn, flytta muspekaren för att välja kategori Internet och klicka på webbläsaren Chromium för att läsa in den (**bild 3-12**).




▲ **Bild 3-12:** Webbläsaren Chromium

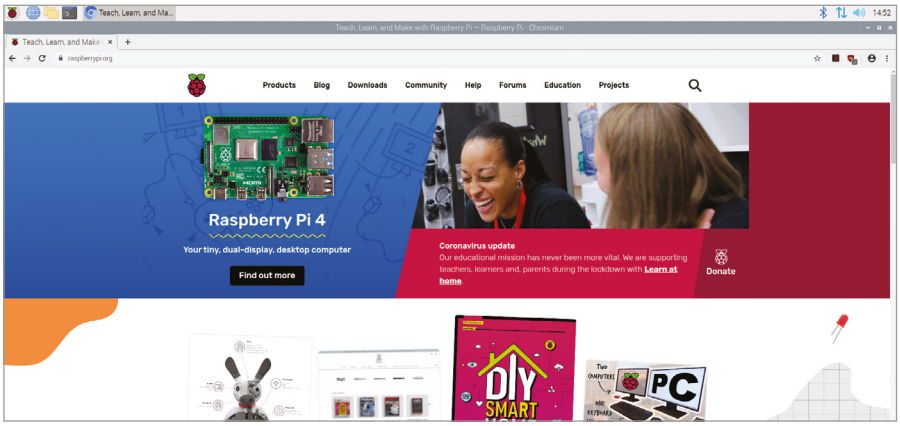
Om du har använt webbläsaren Google Chrome på en annan dator kommer Chromium genast att kännas bekant. Som webbläsare gör Chromium det möjligt för dig att besöka webbplatser, spela videor, spel och till och med kommunicera med människor över hela världen på forum och chattwebbplatser.

Börja använda Chromium genom att maximera fönstret så att det tar upp mer av skärmen: lokalisera de tre ikonerna högst upp till höger i Chromium-fönstrets titelfält (**N**) och klicka på den mellersta uppåtpilen (**P**). Detta är knappen *maximera*, vilken gör att ett fönster fyller skärmen. Till vänster om maximera finns *minimera* (**O**), vilken döljer ett fönster tills du klickar på det i aktivitetsfältet högst upp på skärmen. Krysset till höger om maximera är *stäng* (**Q**), vilken gör precis det du förväntar dig: stänger fönstret.

### STÄNG OCH SPARA

Att stänga ett fönster innan du har sparat ett arbete du har gjort är ingen bra idé. Många program varnar dig så att du kan spara när du klickar på stängningsknappen, men andra gör det inte.

Klicka i adressfältet högst upp i Chromium-fönstret – det stora vita fältet med ett förstoringsglas på vänster sida – och skriv **www.raspberrypi.org**. Tryck sedan på tangenten **ENTER** på tangentbordet. Raspberry Pi-webbplatsen läses in (**bild 3-13**). Du kan också skriva sökningar i adressfältet: försök att söka efter ”Raspberry Pi”, ”Raspberry Pi OS” eller ”Pedagogisk databehandling”.




▲ **Bild 3-13:** Raspberry Pi-webbplatsen läses in i Chromium

Första gången du läser in Chromium kan det ta fram flera *flikar* längst upp i fönstret. Om du vill växla till en annan flik ska du klicka på den. Om du vill stänga en flik utan att stänga själva Chromium klickar du på krysset till höger på fliken som du vill stänga. För att öppna en ny flik, vilket är ett praktiskt sätt att öppna flera webbplatser utan att behöva hantera flera Chromium-fönster, klickar du antingen på flikknappen till höger om den sista fliken i listan eller håller ner tangenten **CTRL** på tangentbordet och trycker på tangenten **T** innan du släpper upp **CTRL**.

När du är klar med Chromium klickar du på stängningsknappen längst upp till höger i fönstret.

# Filhanteraren

Filer du sparar, t.ex. program, videor, bilder, hamnar i *hemkatalogen*. Om du vill se hemkatalogen klickar du på hallonikonen igen för att få upp menyn, flyttar muspekaren för att välja Tillbehör och klicka sedan på Filhanteraren för att läsa in det (**bild 3-14**).

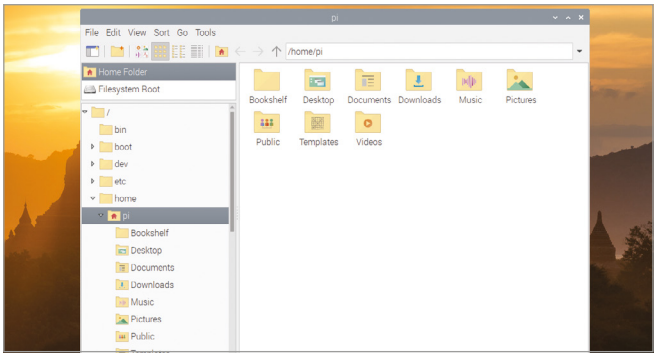



**Bild 3-14:** Filhanteraren

Filhanteraren gör det möjligt för dig att bläddra bland filer och mappar, vilka även kallas *kataloger*, på Raspberry Pi:s microSD-kort, precis som de på eventuella flyttbara lagringsenheter – som USB-flashenheter – som du ansluter till Raspberry Pi:s USB-portar. När du först öppnar den går den automatiskt till hemkatalogen. Här hittar du en serie andra mappar, så kallade *underkataloger*, som är ordnade i kategorier, precis som menyn. De viktigaste underkatalogerna är:

- **Bookshelf:** Den innehåller digitala kopior av böcker, tidskrifter och andra publikationer från Raspberry Pi Press, inklusive en kopia av denna *nybörjarguide*. Du kan läsa dem och ladda fler mer med Bookshelf-applikationen. Du hittar den i hjälpavsnittet på menyn.
- **Desktop:** Den här mappen är den du ser när du först läser in Raspberry Pi OS. Om du sparar en fil här kommer den att visas på skrivbordet, vilket gör den enkel att hitta och ladda.
- **Documents:** Här sparar du de flesta filer du skapar, från noveller till recept.
- **Downloads:** När du laddar ner en fil från internet med webbläsaren Chromium sparas den automatiskt i Downloads.
- **Music:** All musik du skapar eller lägger in på Raspberry Pi kan lagras här.
- **Pictures:** Den här mappen är till särskilt för bilder. I tekniska termer benämns de *bildfiler*.
- **Public:** De flesta av dina filer är privata, men allt du lägger i Public är tillgängligt för andra användare av Raspberry Pi, även om de har sitt eget användarnamn och lösenord.

- **Templates:** Den här mappen innehåller mallar, tomma dokument med en grundläggande layout eller struktur som redan finns på plats, som du har skapat eller som installeras av applikationerna.
- **Videos:**En mapp för videor som är det första ställe där de flesta program för videouppspelning program kommer att leta.

Fönstret Filhanteraren är uppdelat i två rutor: den vänstra rutan visar katalogerna på Raspberry Pi och den högra rutan visar filerna och underkatalogerna i katalogen som är vald i den vänstra rutan. Om du ansluter en flyttbar lagringsenhet till Raspberry Pi:s USB-port dyker ett fönster upp som frågar om du vill öppna den i Filhanteraren (**bild 3-15**). Klicka på OK så kan du se filerna och katalogerna på enheten.

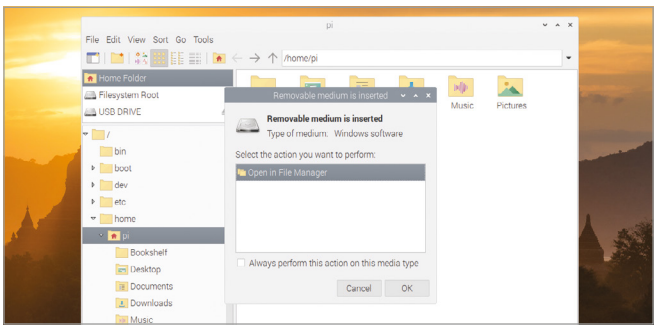
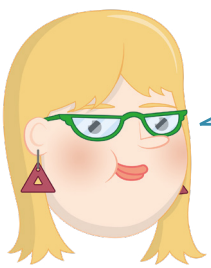



**Bild 3-15:** Sätta i en borttagbar lagringsenhet

Filer kan enkelt kopieras mellan Raspberry Pi:s microSD-kort och en flyttbar enhet: med både hemkatalogen och den flyttbara enheten öppna i separata Filhanterarfönster flyttar du muspekaren till den fil du vill kopiera, klickar och håller vänster musknapp nedtryckt, för muspekaren till det andra fönstret och släpper musknappen (**bild 3-16**, på nästa sida). Detta brukar kallas *dra och släppa*.

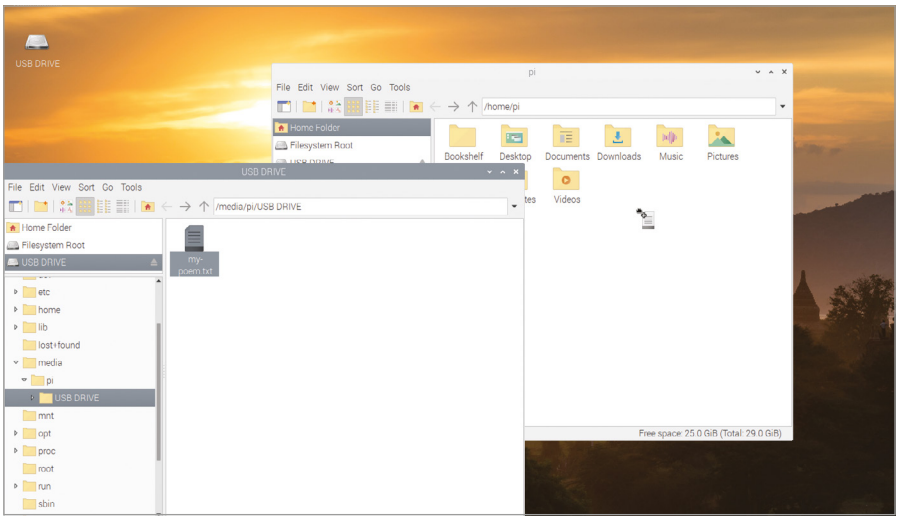
En annan metod är att klicka en gång på filen, klicka på Redigera-menyn, klicka på Kopiera, klicka på det andra fönstret, klicka på Redigera-menyn och klicka på Klistra in.

Alternativet Klipp ut, också tillgängligt i menyn Redigera, gör samma sak förutom att det raderar filen från sin ursprungliga plats efter att kopian har gjorts. Båda alternativen kan också användas genom kortkommandona **CTRL+C** (kopiera) eller **CTRL+X** (klipp ut), och klistra in genom **CTRL+V**.



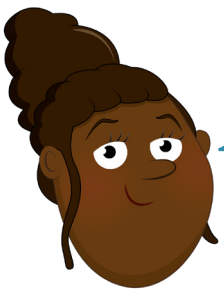
### TANGENTBORDSGENVÄGAR

När du ser en tangentbordsgenväg som **CTRL+C** betyder det att du ska hålla ner den första tangenten på tangentbordet (**CTRL**), trycka på den andra tangenten (**C**) och sedan släpp båda tangenterna.




▲ **Bild 3-16:** Dra och släppa en fil

När du är klar med experimenterandet kan du stänga Filhanteraren genom att klicka på stängningsknappen längst upp till vänster i fönstret. Om du har mer än ett fönster öppet, stäng övriga också. Om du har anslutit en flyttbar lagringsenhet till Raspberry Pi matar du ut den genom att klicka på utmatningsknappen längst upp till höger på skärmen. Du letar upp enheten i listan och klickar på den innan du kopplar från den.



### MATA UT ENHETER

Använd alltid utmatningsknappen innan du kopplar bort en extern lagringsenhet. Om du inte gör det kan filerna på den ta skada och bli oanvändbara.

## Produktivitetspaketet LibreOffice

Om du vill se en annan sak Raspberry Pi kan göra kan du klicka på hallonmenyikonen, flytta muspekaren till Kontor och klicka på LibreOffice Writer. Detta läser in ordbehandlingsdelen av LibreOffice (**bild 3-17**), en populär *produktivitetssvit* – om du har använt Microsoft Office eller Google Docs har du använt en produktivitetssvit.

Observera: LibreOffice kanske inte finns installerat som standard på alla Raspberry Pi OS. Om det inte gör det kan du använda verktyget Recommended Software (se sidan 49) för att installera det.

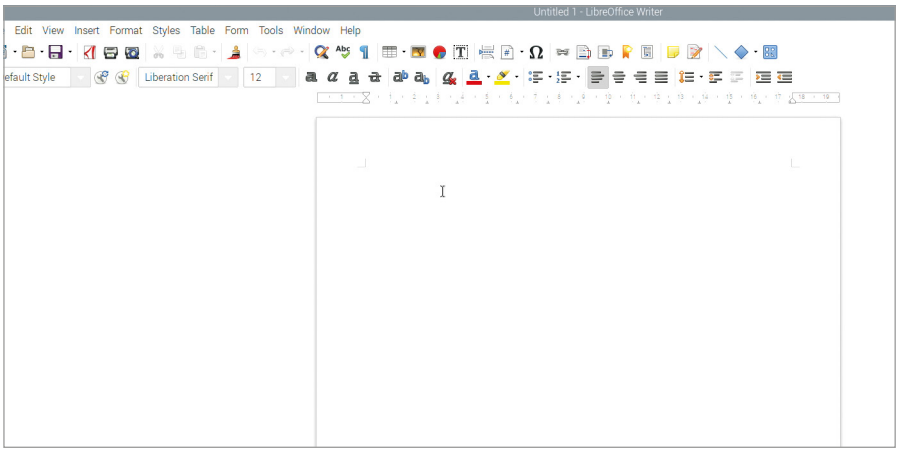



**Bild 3-17:** Programmet LibreOffice Writer

Med en ordbehandlare kan du inte bara skriva dokument, utan även formatera dem på smarta sätt: du kan ändra typsnitt, färg, storlek, lägga till effekter och till och med infoga bilder, diagram, tabeller och annat innehåll. En ordbehandlare gör att du kan kontrollera om det förekommer fel i ditt arbete genom att den markerar stavfel och grammatikproblem i rött respektive grönt när du skriver.

Börja med att skriva ett stycke om vad du hittills har lärt dig om Raspberry Pi och dess programvara. Experimentera med de olika ikonerna högst upp i fönstret för att se vad de gör: se om du kan göra det du skrivit större och ändra färg på det. Om du inte är säker på hur du ska göra, håller du bara muspekaren över varje ikon i tur och ordning för att få se ett tips som berättar vad den ikonen gör. När du är nöjd klickar du på menyn File och alternativet Save för att spara ditt arbete (**bild 3-18**). Ge det ett namn och klicka på knappen Save.

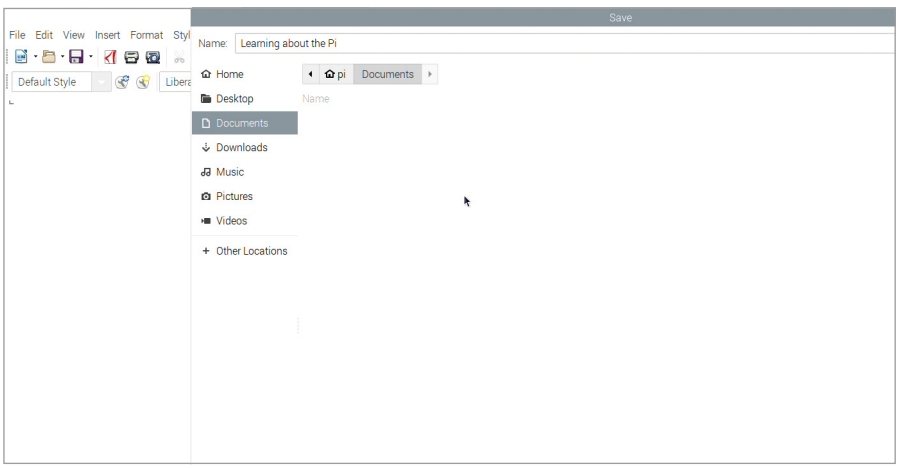



**Bild 3-18:** Spara ett dokument

## SPARA DITT ARBETE

Ha som vana att spara ditt arbete, även om du inte har avslutat det ännu. Det kommer att spara dig mycket besvär om det blir strömavbrott och du blir avbruten halvvägs!

LibreOffice Writer är bara en del av hela produktivitetssviten LibreOffice. De andra delarna, som du hittar i samma menykategori, Kontor, som LibreOffice Writer, är:

- **LibreOffice Base:**En databas: ett verktyg för att lagra information, leta upp den snabbt och analysera den.
- **LibreOffice Calc**: Ett kalkylark: ett verktyg för att hantera siffror och skapa diagram och grafer.
- **LibreOffice Draw**: Ett illustrationsprogram: ett verktyg för att skapa bilder och diagram.
- **LibreOffice Impress**: Ett presentationsprogram för att skapa bilder och visa bildspel.
- **LibreOffice Math**: En formelredigerare: ett verktyg för att skapa korrekt formaterade matematiska formler som sedan kan användas i andra dokument.

LibreOffice är även tillgänglig för andra datorer och operativsystem. Om du gillar att använda den på Raspberry Pi kan du ladda ner den gratis från **libreoffice.org** och installera den på alla Microsoft Windows-, Apple macOS- eller Linux-datorer.

Om du vill veta mer om att använda LibreOffice klickar du på Hjälp-menyn. Annars klickar du på stängningsknappen längst upp till höger i fönstret för att stänga LibreOffice.

## FÅ HJÄLP

De flesta program har en hjälpmeny som innehåller allt från information om vad programmet gör till guider för hur man använder det. Om du någonsin känner dig vilsen eller överväldigad av ett program kan du leta efter hjälpmenyn för att orientera dig på nytt.

## Verktyget Recommended Software

Även om Raspberry Pi OS levereras med ett brett utbud av programvaror är det kompatibelt med ännu mer. Ett urval med det bästa av dessa programvaror finns i verktyget Recommended Software.

Observera att verktyget Recommended Software behöver en internetanslutning. Om Raspberry Pi är uppkopplad klickar du på hallonmenyikonen, flyttar muspekaren till Inställningar och klickar på Recommended Software. Verktyget läses in och börjar sedan ladda ner information om tillgänglig programvara.

Efter några sekunder visas en lista över kompatibla programvarupaket (**bild 3-19**). Dessa, liksom programvaror i hallonmenyn, är ordnade i olika kategorier. Klicka på en kategori i rutan till vänster för att se programvara från den kategorin, eller klicka på All Programs för att se allt.

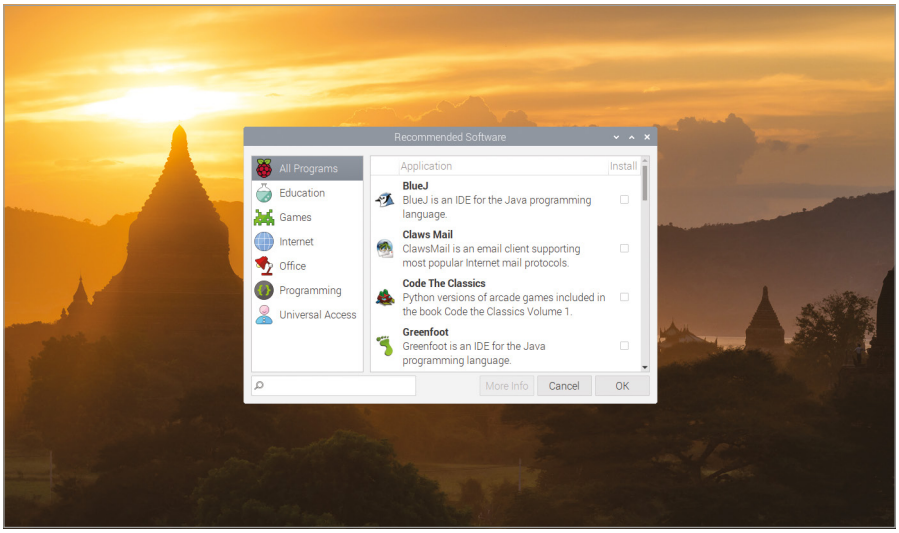



**Bild 3-19:** Verktyget Recommended Software

Om en programvara har en bock bredvid sig är den redan installerad på Raspberry Pi. Om den inte har det kan du klicka på kryssrutan bredvid den för att lägga till en bock och markera den för installation. Du kan markera så många program du vill innan du installerar dem på en och samma gång, men om du använder ett microSD-kort med mindre kapacitet än den som rekommenderas kanske du inte har plats för alla.

Du kan också avinstallera programvara på samma sätt: hitta en programvara som redan har en bock i kryssrutan och klicka sedan på kryssrutan för att ta bort den. Om du har gjort ett misstag eller om du ändrar dig klickar du bara igen för att sätta tillbaka bocken.

När du är nöjd med dina val klickar du på knappen OK för att starta installationen eller avinstallationen (**bild 3-20**, på nästa sida). När du har laddat ner och installerat ny programvara som du har valt visas en dialogruta. Klicka på OK för att stänga verktyget Recommended Software.

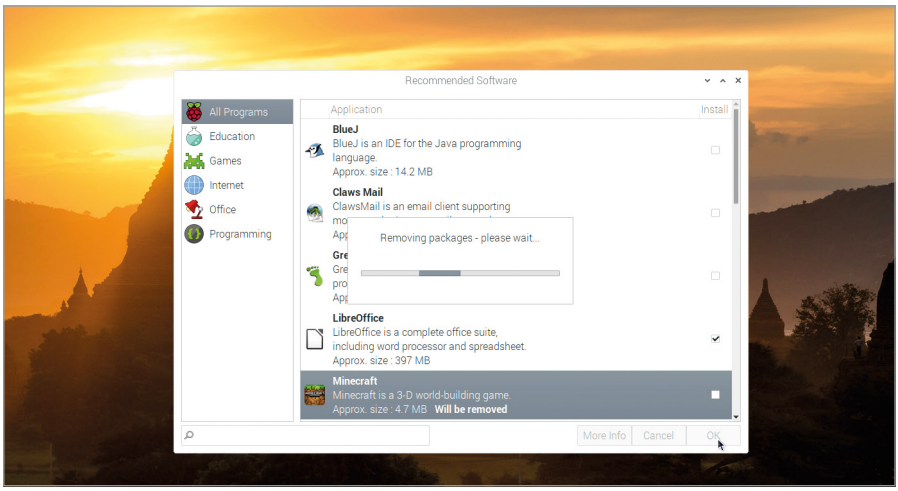



**Bild 3-20:** Avinstallera programvara

Ett annat verktyg som kan användas för att installera eller avinstallera programvara är verktyget Add/Remove Software. Det finns i samma kategori, Inställningar, i Raspberry Pi OS-menyn. Det erbjuder ett bredare urval av programvaror, men dessa har inte kontrollerats av Raspberry Pi Foundation.

## Verktyget Raspberry Pi Configuration

Det sista programmet du får lära dig om i det här kapitlet kallas Raspberry Pi Configuration, och det är ungefär som välkomstguiden du använde i början: det gör det möjligt för dig att ändra olika inställningar i Raspberry Pi OS. Klicka på hallonikonen, flytta muspekaren för att välja kategorin Inställningar och klicka sedan på Raspberry Pi Configuration för att läsa in det (**bild 3-21**).

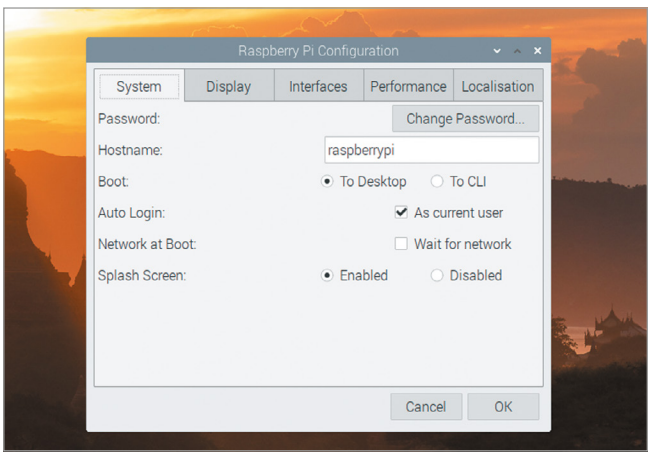



**Bild 3-21:** Verktyget Raspberry Pi Configuration

Verktyget är uppdelat i fem flikar. Den första av dessa är System: här kan du ändra lösenordet för kontot, ange ett värdnamn – namnet Raspberry Pi används i ditt lokala trådlösa eller trådbundna nätverk – och ändra en rad andra inställningar. Majoriteten av dessa borde dock inte behöva ändras. Klicka på fliken Display för att visa nästa kategori. Här kan du ändra skärmvisningsinställningarna om det behövs för att passa tv:n eller bildskärmen.

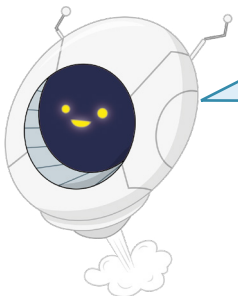
### FLER DETALJER

Denna korta översikt finns här för att du ska vänja dig vid verktyget. Mer detaljerad information om inställningarna finns i **Bilaga E Verktyget Raspberry Pi Configuration**.

Fliken Interface erbjuder en rad inställningar, som samtliga är inaktiverade till att börja med. Dessa inställningar bör bara ändras om du lägger till ny maskinvara, till exempel Raspberry Pi Camera Module, och sedan bara om du har fått anvisningar från maskinvarutillverkaren. Undantagen från denna regel är: SSH, som möjliggör en ”Secure Shell” och gör det möjligt för dig att logga in på Raspberry Pi från en annan dator i nätverket med en SSH-klient; VNC, som möjliggör en ”virtuell nätverksdator” så att du kan se och styra Raspberry Pi OS-skrivbordet från en annan dator i nätverket med en VNC-klient; och Remote GPIO, som gör det möjligt för dig att använda Raspberry Pi:s GPIO-stift – dessa får du lära dig mer om i **kapitel 6, Fysisk databehandling med Scratch och Python** – från en annan dator i nätverket.

Klicka på fliken Performance för att se den fjärde kategorin. Här kan du ställa in mängden minne som används av Raspberry Pi:s grafikbehandlingsenhet (GPU) och, för vissa modeller, öka prestandan för Raspberry Pi genom en process som kallas *överklockning*. Som tidigare är det dock bäst att låta dessa inställningar vara om du inte vet att du behöver ändra dem.

Slutligen kan du klicka på fliken Localisation för att se den sista kategorin. Här kan du ändra din plats, vilken styr saker som språket som används i Raspberry Pi OS och hur siffror visas, ändra tidszonen, ändra tangentbordslayout och ställa in land för Wi-Fi-ändamål. För tillfället ska du dock bara klicka på Avbryt för att stänga verktyget utan att göra några ändringar.



### VARNING!

Olika länder har olika regler om vilka frekvenser en Wi-Fi-radio får använda. Att ställa in Wi-Fi-landet i verktyget Raspberry Pi Configuration till ett annat land än det du faktiskt befinner dig i kommer sannolikt att göra det svårt att ansluta till nätverken och kan till och med vara olagligt enligt radiolicenslagar – så gör inte det!

## Stänga av

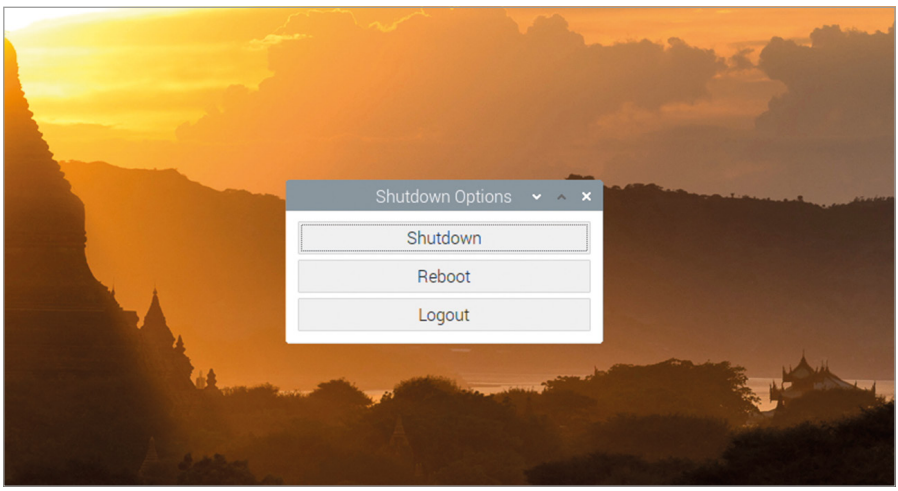
Nu när du har utforskat Raspberry Pi OS-skrivbordet är det dags att lära sig en mycket viktig färdighet: att stänga av Raspberry Pi på ett säkert sätt. Precis som alla andra datorer håller Raspberry Pi de filer du arbetar med i *flyktigt minne*, minne som töms när systemet stängs av. För dokument du skapar räcker det att spara dem i tur och ordning. Detta tar filen från det flyktiga minnet till *icke-flyktigt minne*, microSD-kortet, för att säkerställa att du inte blir av med något.

Dokumenten du arbetar med är dock inte de enda öppna filerna. Raspberry Pi OS i sig håller ett antal filer öppna medan det körs, och att dra ut strömkabeln från Raspberry Pi medan dessa fortfarande är öppna kan leda till att operativsystemet tar skada och behöver installeras om.

För att förhindra att detta händer måste du se till att du ber Raspberry Pi OS att spara alla sina filer och göra sig redo för att stängas av – en process som kallas *shutting down*, eller avstängning av operativsystemet.

Klicka på hallonikonen längst upp till vänster på skrivbordet och klicka sedan på Shutdown. Ett fönster med tre alternativ visas (**bild 3-22**): Shutdown, Reboot och Logout. Avstängning är det alternativ du använder mest: genom att klicka på detta kommer Raspberry Pi OS att stänga alla öppna programvaror och filer och sedan stänga av Raspberry Pi. När skärmen har blivit svart väntar du några sekunder tills det gröna ljuset på Raspberry Pi släcks. Då är det säkert att stänga av strömförsörjningen.

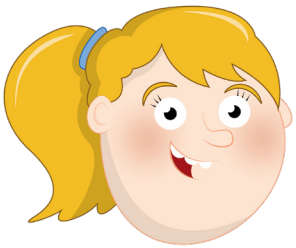
Om du vill sätta på Raspberry Pi igen är det bara att koppla ur och ansluta strömkabeln igen, eller koppla in strömmen i vägguttaget.




▲ **Bild 3-22:** Stänga av Raspberry Pi

Omstart, eller Reboot, sker genom en process som liknar avstängningsprocessen. Allt stängs av, men istället för att strömmen stängs av för Raspberry Pi startas datorn om på nästan exakt samma sätt som om du valt att stänga av och sedan koppla bort och återansluta strömkabeln. Du måste använda Reboot om du gör vissa ändringar som kräver en omstart av operativsystemet, till exempel vid installation av vissa uppdateringar av kärnprogramvaran, eller om någon programvara har fått fel, även kallat *kraschar*, och gjort så att Raspberry Pi OS har hamnat i ett oanvändbart tillstånd.

Slutligen är utloggning, eller Logout, bara riktigt användbart om du har mer än ett användarkonto på Raspberry Pi: den stänger alla program som du har öppnat och tar dig till en inloggningsskärm där du uppmanas att ange ett användarnamn och ett lösenord. Om du av misstag trycker på Logout och vill komma in igen, ska du bara skriva ”pi” som användarnamn och det lösenord som du valde i välkomstguiden i början av detta kapitel.



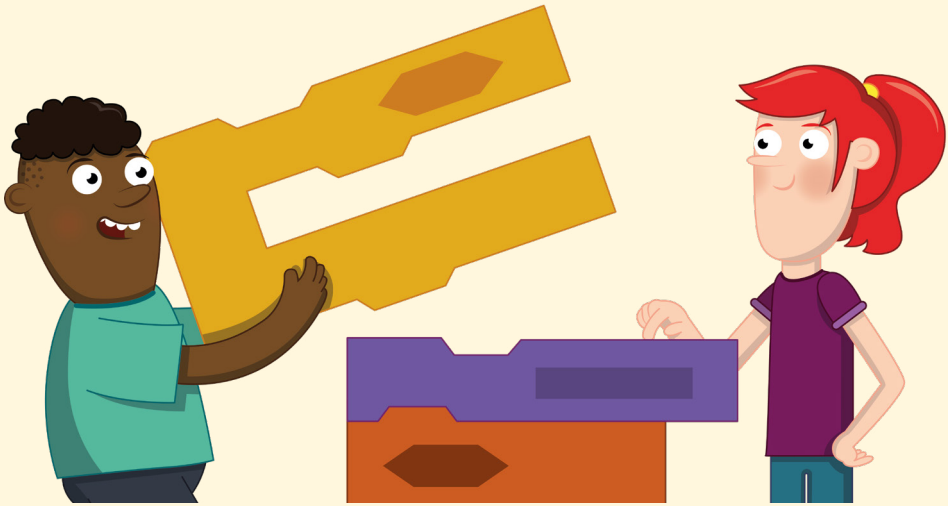
**VARNING!**

Dra aldrig ur strömkabeln från en Raspberry Pi utan att stänga av den först. Om du gör det är det troligt att operativsystemet skadas, och du kan även förlora alla filer du har skapat eller laddat ner.

Kapitel 4

# Programmering med Scratch 3

## Lär dig hur man skriver kod med Scratch, det blockbaserade programmeringsspråket

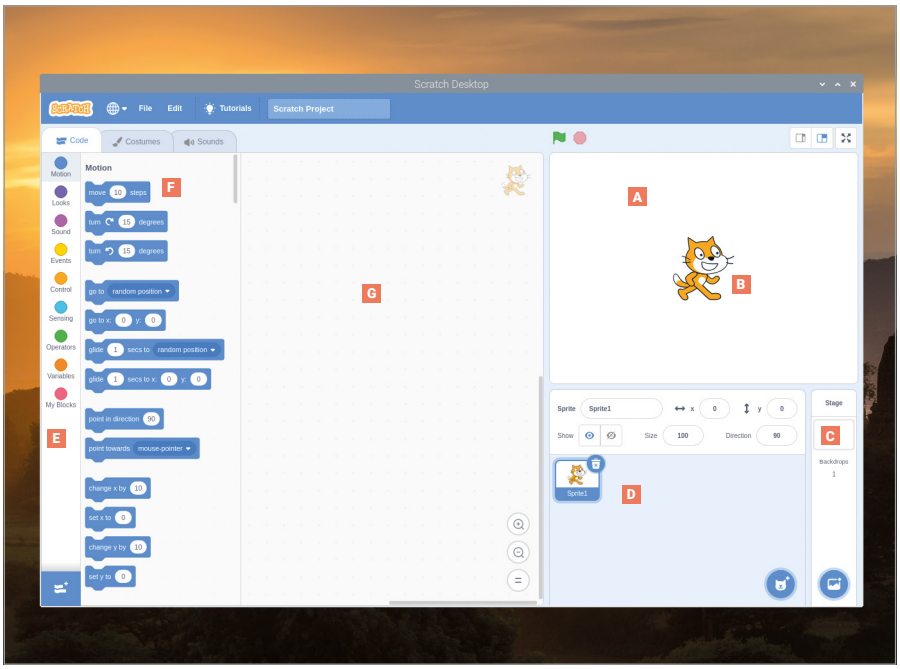


Att använda Raspberry Pi handlar inte bara om att använda programvara som andra har skapat. Det handlar om att skapa din egen programvara, baserad på nästan vad som helst som din fantasi kan framkalla. Oavsett om du har tidigare erfarenhet av att skapa egna program, en process som kallas programmering eller kodning, eller inte, kommer du att upptäcka att Raspberry Pi är en bra plattform för skapande och experimenterande.

För att kunna skriva kod på Raspberry Pi behöver du Scratch, ett visuellt programmeringsspråk som utvecklats av Massachusetts Institute of Technology (MIT). Medan traditionella programmeringsspråk vill att du skriver textbaserade instruktioner som datorn ska utföra, ungefär på samma sätt som du kan skriva ett recept för att baka en tårta, vill Scratch att du bygger programmet steg för steg med hjälp av block – förskrivna bitar av kod som är dolda bakom färgkodade pusselbitar.

Scratch är ett utmärkt första språk för unga och gamla nybörjarkodare, men låt dig inte luras av det vänliga utseendet: det är en kraftfull och fullt funktionell programmeringsmiljö där du kan skapa allt från enkla spel och animationer till komplexa interaktiva robotprojekt.

# Introduktion av gränssnittet i Scratch 3




**A** **Scenområde** – Som skådespelare i en pjäs rör sig sprajtarna runt på scenen styrda av ditt program.

**B** **Sprajt** – Figurerna eller objekten du styr i ett Scratch-program kallas sprajtar och finns på scenen.

**C** **Scenkontroller** – Scenen kan ändras, inklusive att du lägger till dina egna bilder som bakgrunder, med hjälp av scenkontrollerna.

**D** **Lista över sprajtar** – Alla sprajtar som du har skapat eller laddat i Scratch kommer att visas i den här delen av fönstret.

**E** **Blockpalett** – Alla block som är tillgängliga för programmet visas i blockpaletten, som har färgkodade kategorier.

## SCRATCH-VERSIONER

När boken skrevs hade Raspberry Pi OS tre versioner av Scratch: Scratch 1, 2 och 3, vilka alla ingår i avsnittet Programmering i Raspberry Pi OS-menyn. Detta kapitel är skrivet för Scratch 3. Observera att Scratch 3 bara körs på Raspberry Pi 4. Om du vill använda Scratch 2 istället fungerar den versionen inte på Raspberry Pi Zero, Model A, A+, B eller B+.

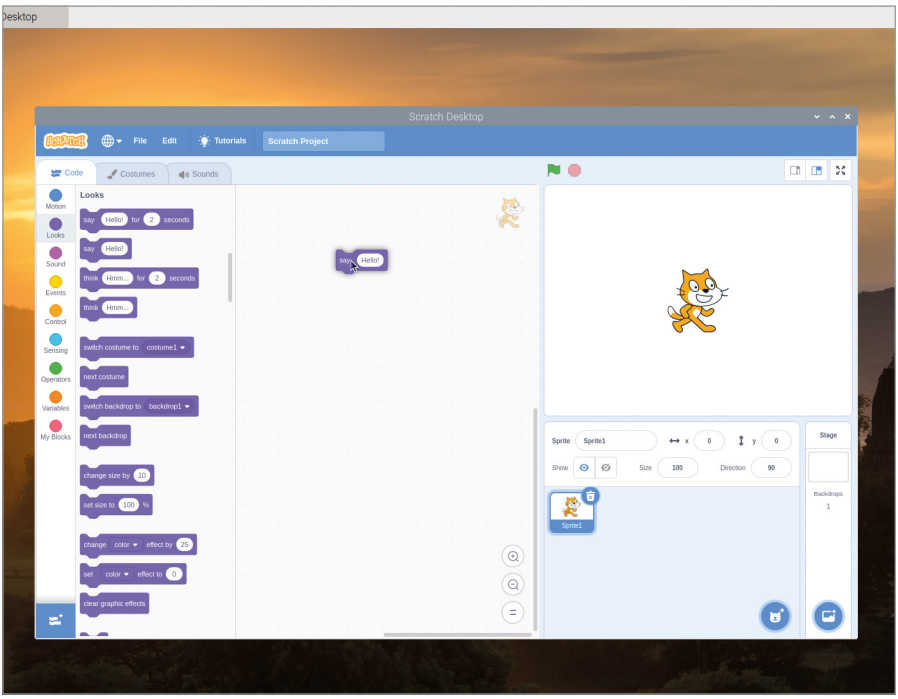
**F** **Block** – Block, som är förskrivna bitar av programkod, gör det möjligt för dig att bygga programmet steg för steg.

**G** **Kodområde** – Kodområdet är den plats där du bygger programmet genom att dra och släppa block från blockpaletten för att bilda skript.

# Ditt första Scratch-program: Hej, världen!

Scratch 3 läses in precis som alla andra program på Raspberry Pi: klicka på hallonikonen för att läsa in Raspberry Pi OS-menyn, flytta markören till avsnittet Programmering och klicka på Scratch 3. Efter några sekunder läses användargränssnittet in för Scratch 3.

I de flesta programmeringsspråk måste du tala om för datorn vad den ska göra genom skriftliga instruktioner, men Scratch är annorlunda. Börja med att klicka på kategorin Utseende i blockpaletten, som finns till vänster i Scratch-fönstret. Detta tar fram blocken i den kategorin, lilafärgade. Hitta säg Hej! -blocket, klicka och håll ner vänster musknapp på det och dra det till kodområdet i mitten av Scratch-fönstret innan du släpper musknappen (**Bild 4-1**).




▲ **Bild 4-1:** Dra och släpp blocket i kodområdet

Titta på formen på blocket du precis släppt: det har ett hål högst upp och en matchande del som sticker ut längst ner. Precis som en pusselbit visar detta dig att blocket förväntar sig att ha något ovanför sig och något under sig. I det här programmet är det som ska sitta ovanför en *utlösare*.

Klicka på kategorin Händelser i blockpaletten, guldfärgad, och klicka sedan på och dra när klickas på -blocket – kallat *hattblock* – till kodområdet. Placera det så att biten som sticker ut från nederdelen ansluts till hålet högst upp på säg Hej! -blocket tills du ser en vit

kontur och släpp sedan musknappen. Du behöver inte vara exakt. Om det är tillräckligt nära sätts blocket på plats precis som en pusselbit. Om det inte gör det ska du klicka på och hålla det igen för att justera positionen tills det hamnar rätt.


Programmet är nu klart. För att få det att fungera, vilket brukar kallas för att *köra* programmet, ska du klicka på den gröna flaggikonen längst upp till vänster i scenområdet. Om allt har gått bra hälsar kattsprajten på scenen dig med ett glatt ”Hej!” (**Bild 4-2**). Du har gjort ditt första program!

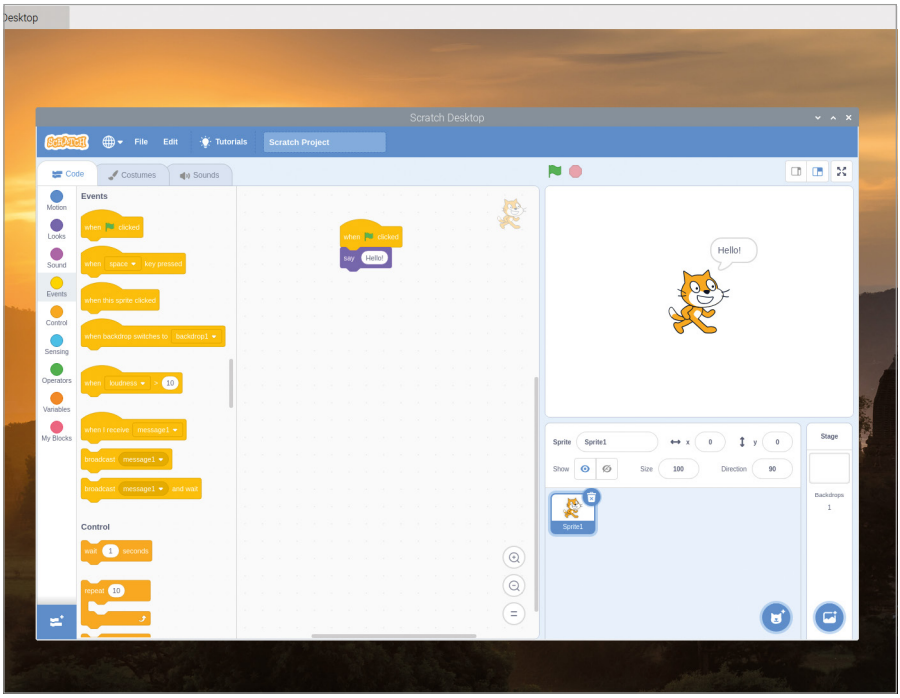
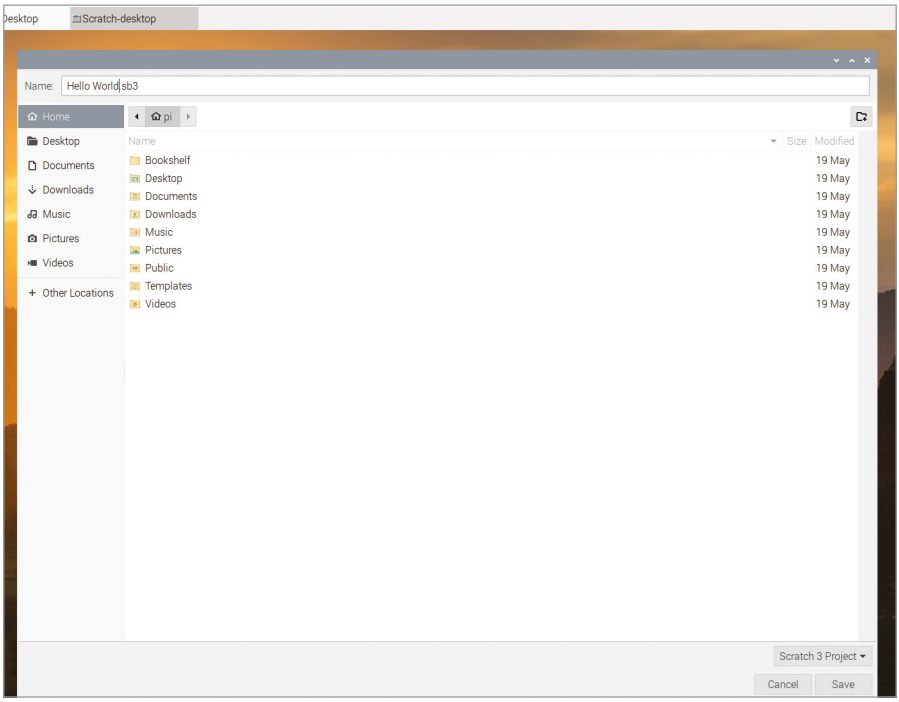
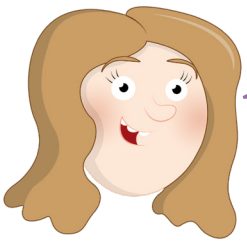



**Bild 4-2:** Klicka på den gröna flaggan ovanför scenen så säger katten ”Hej”

Innan du går vidare ska du namnge och spara programmet. Klicka på menyn Arkiv och sedan på ”Spara till din dator”. Skriv in ett namn och klicka på knappen Save (**Bild 4-3**).




▲ **Bild 4-3:** Spara programmet med ett namn som är lätt att komma ihåg



### VAD KAN DET SÄGA?

Vissa block i Scratch går att ändra. Försök att klicka på ordet ”Hej!” och skriva något annat och klicka sedan på den gröna flaggan igen. Vad händer på scenen?

## Nästa steg: sekvensering

Programmet har två block, men det har bara en riktig instruktion: att säga ”Hej!” varje gång flaggan klickas och programmet körs. För att göra mer måste du ha koll på *sekvensering*. Datorprogram är i sin enkelhet en lista med instruktioner, precis som ett recept. Varje instruktion följer på den senaste i en logisk progression som kallas en *linjär sekvens*.

Börja med att klicka och dra säg Hej! -blocket från kodområdet tillbaka till blockpaletten (**Bild 4-4**). Detta raderar blocket genom att det tas bort från programmet så att bara utlösarblocket återstår när klickas på.

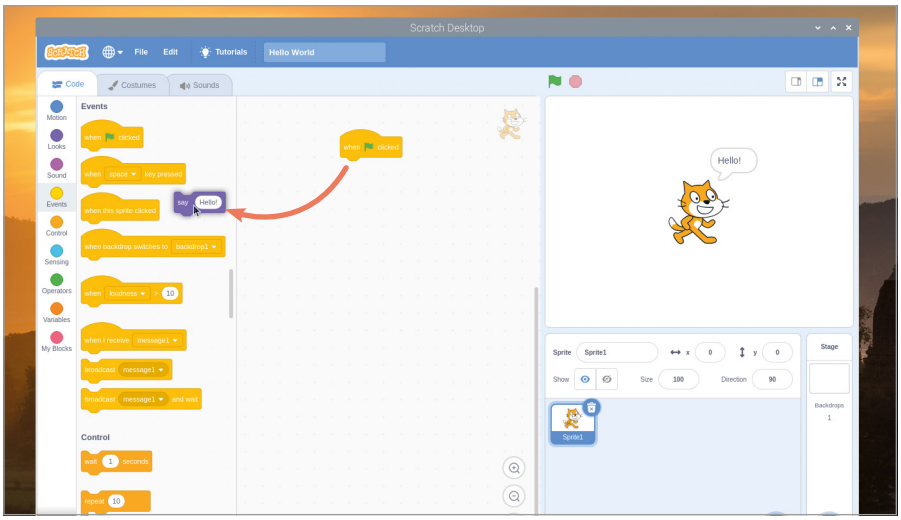
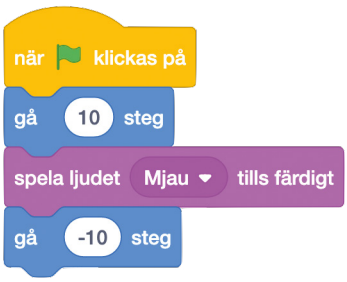



**Bild 4-4:** Om du vill radera ett block drar du bara ut det från kodområdet

Klicka på kategorin Rörelse i blockpaletten och klicka sedan på och dra **gå 10 steg** -blocket så att det låses på plats under utlösarblocket i kodområdet. Som namnet antyder säger detta till sprajten, katten, att flytta ett antal steg i den riktning som den för närvarande står vänd mot.


Lägg till fler instruktioner i programmet för att skapa en sekvens. Klicka på paletten Ljud, färgkodad rosa, och klicka sedan på och dra **spela ljudet Mjau tills färdigt** -blocket så att det låses under **gå 10 steg** -blocket. Fortsätt: klicka igen på kategorin Rörelse och dra ett till **gå 10 steg** -block under ljudblocket, men den här gången klickar du på ”10” för att välja det och skriver ”-10” för att skapa ett **gå -10 steg** -block.



Klicka på den gröna flaggan ovanför scenen för att köra programmet. Du ser katten flytta sig åt höger, göra ett mjau-ljud – se till att du har anslutna högtalare eller hörlurar så att du kan höra den – för att sedan gå tillbaka till början igen. Om du klickar på flaggan igen upprepar katten alltihop.

Grattis: du har skapat en sekvens av instruktioner som Scratch kör igenom, en i taget, uppifrån och ned. Scratch kör bara en instruktion i taget från sekvensen, men den gör det mycket snabbt: försök att ta bort `spela ljudet Mjau tills färdigt`-blocket genom att klicka på det och dra det nedre `gå -10 steg`-blocket för att ta bort det, dra `spela ljudet Mjau tills färdigt`-blocket till blockpaletten och ersätt det sedan med det enklare `starta ljud Mjau`-blocket innan du drar tillbaka `gå -10 steg`-blocket till slutet av programmet.


Klicka på den gröna flaggan för att köra programmet igen. Då verkar kattsprajten inte röra sig. Sprajten rör sig faktiskt, men den rör sig tillbaka så snabbt att den verkar stå stilla. Det beror på att det använda `starta ljud Mjau`-blocket inte väntar på att ljudet ska slutföras innan nästa steg: eftersom Raspberry Pi ”tänker” så snabbt körs nästa instruktion innan du har en chans att se kattsprajten röra sig. Det finns ett annat sätt att åtgärda detta, förutom att använda `spela ljudet Mjau tills färdigt`-blocket: klicka på den ljusorange kategorin Kontroll i blockpaletten och klicka sedan på och dra ett `vänta 1 sekunder`-block mellan `starta ljud Mjau`-blocket och det nedre `gå -10 steg`-blocket.

Klicka på den gröna flaggan för att köra programmet en sista gång. Nu ser du att kattsprajten väntar en sekund efter att ha flyttat sig åt höger innan den flyttar tillbaka åt vänster igen. Detta brukar kallas en *fördröjning* och är nyckeln till att styra hur lång tid instruktionssekvensen ska ta att köra.

**UTMANING:** LÄGG TILL FLER STEG

Försök lägga till fler steg i sekvensen och ändra värdena i befintliga steg. Vad händer när antalet steg i ett dragblock inte matchar antalet steg i ett annat? Vad händer om du försöker spela upp ett ljud medan ett annat ljud fortfarande spelas upp?

## Loopa öglan

Sekvensen som du har skapat hittills körs bara en gång: du klickar på den gröna flaggan, kattsprajten rör sig och jamar och sedan stannar programmet tills du klickar på den gröna flaggan igen. Den behöver dock inte stanna eftersom Scratch innehåller en typ av kontrollblock som kallas en *ögla*.

Klicka på kategorin Kontroll i blockpaletten och leta upp för alltid -blocket. Klicka och dra detta till kodområdet och släpp det under när klickas på -blocket och ovanför det första gå 10 steg -blocket.


Lägg märke till hur det C-formade eviga blocket automatiskt växer och omger de andra blocken i sekvensen. Klicka på den gröna flaggan nu så ser du snabbt vad för alltid -blcoket gör: istället för att programmet körs en gång och avslutas kommer det att köras om och om igen – bokstavligen för evigt. I programmering kallas detta för en *oändlig ögla* – bokstavligen en ögla som aldrig tar slut.

Om du tycker det blir lite jobbigt att höra det eviga jamandet ska du klicka på den röda oktagonen bredvid den gröna flaggan ovanför scenområdet för att stoppa programmet. För att

ändra typ av ögla, klicka på och dra det första [gå 10 steg]-blocket och dra det och blocken under det ur [för alltid]-blocket och släpp dem sedan under [när klickas på]-blocket. Klicka på och dra [för alltid]-blocket till blockpaletten för att radera det, klicka sedan på och dra [repetera 10]-blocket under [när klickas på]-blocket så att det omsluter de andra blocken.


Klicka på den gröna flaggan för att köra det nya programmet. Vid första anblick verkar det göra samma sak som din ursprungliga version, nämligen upprepa sekvensen av instruktioner om och om igen. Men den här gången kommer öglan att avslutas efter tio upprepningar istället för att fortsätta för evigt. Detta kallas en *finit ögla*: du definierar när den ska avslutas. Öglor är kraftfulla verktyg och de flesta program – särskilt spel och avkänningsprogram – använder sig mycket av både oändliga och finita öglor.

### VAD HÄNDER NU?

Vad händer om du ändrar siffran i ögleblocket för att göra den större? Vad händer om den blir mindre? Vad händer om du sätter in siffran 0 i ögleblocket?

## Variabler och villkor

De sista begreppen du behöver förstå innan du börjar koda Scratch-program på allvar är nära besläktade: *variabler* och *villkor*. En variabel är, som namnet antyder, ett värde som kan variera – med andra ord ändras – över tiden och under styrning av programmet. En variabel har två huvudegenskaper: namnet och värdet den lagrar. Det värdet behöver inte heller vara ett tal: det kan vara siffror, text, sant eller falskt eller helt tomt – så kallat *nollvärde*.

Variabler är kraftfulla verktyg. Tänk på de saker du måste spåra i ett spel: en karaktärs hälsa, hastigheten för rörliga objekt, nivån som för närvarande spelas och poängen. Alla dessa spåras som variabler.

Klicka först på Arkiv-menyn och spara ditt befintliga program genom att klicka på ”Spara till din dator”. Om du redan har sparat programmet blir du tillfrågad om du vill skriva över det och ersätta den gamla sparade kopian med din nya uppdaterade version. Klicka sedan på Arkiv och sedan på Nytt för att starta ett nytt, tomt projekt (klicka på OK när du blir tillfrågad om du vill ersätta innehållet i det aktuella projektet). Klicka på den mörkorangea kategorin Variabler i blockpaletten och sedan på knappen ”Skapa en variabel”. Skriv ”öglor” som variabelnamn (**Bild 4-5**) och klicka sedan på OK så att en serie block visas i blockpaletten.

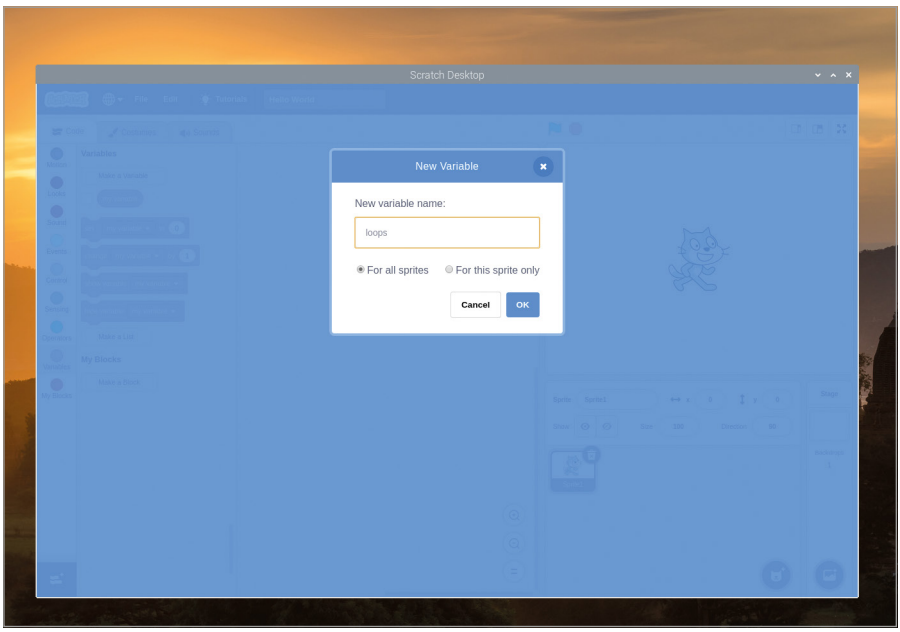
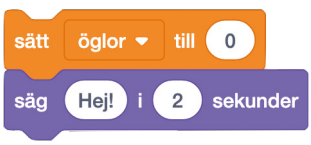


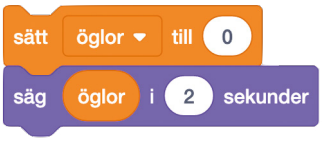

**Bild 4-5:** Namnge den nya variabeln

Klicka på och dra sätt öglor till 0 -blocket till kodområdet. Detta berättar för programmet att det ska *initialisera* variabeln med värdet 0. Klicka sedan på kategorin Utseende i blockpaletten och dra säg Hej! i 2 sekunder -blocket under sätt öglor till 0 -blocket.

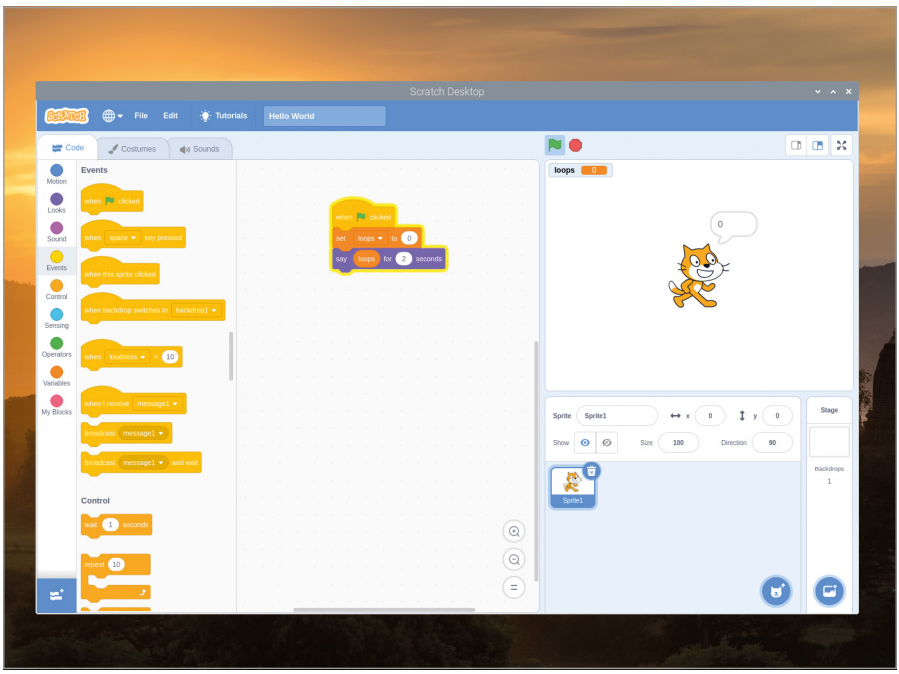



Som du tidigare har sett får säg Hej! -blocken kattsprajten att säga det som står skrivet i dem. I stället för att själv skriva in meddelandet i blocket kan du använda en variabel. Klicka dig tillbaka till kategorin Variabler i blockpaletten och klicka sedan på och dra det rundade öglor-blocket – vilket kallas ett *rapporteringsblock* och finns högst upp i listan med en

kryssruta bredvid – över ordet ”Hej!” i säg Hej! i 2 sekunder -blocket. Detta skapar ett nytt, kombinerat block: säg öglor i 2 sekunder.



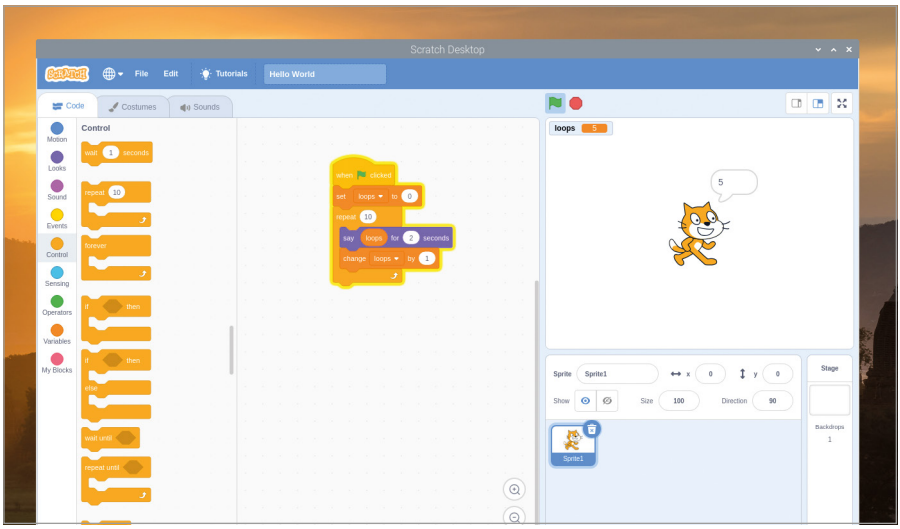

Klicka på kategorin Händelser i blockpaletten och klicka på och dra när klickas på -blocket för att placera det högst upp i blocksekvensen. Klicka på den gröna flaggan ovanför scenområdet så får du se kattsprajten säga ”0” (**Bild 4-6**) – värdet du gav variabeln ”öglor”.



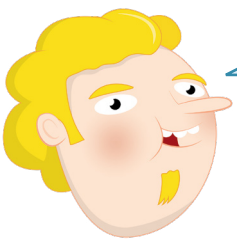

▲ **Bild 4-6:** Den här gången säger katten värdet på variabeln

Variabler är dock inte oföränderliga. Klicka på kategorin Variabler i blockpaletten och klicka sedan på och dra ändra öglor med 1 -blocket till längst ner i blocksekvensen. Klicka sedan på kategorin Kontroll, klicka på och dra ett repetera 10 -block och släpp det så att det börjar direkt under sätt öglor till 0 -blocket och omsluter de återstående blocken i sekvensen.


Klicka på den gröna flaggan igen. Den här gången ser du katten räkna uppåt från 0 till 9. Detta fungerar eftersom programmet nu ändrar, eller *modifierar*, själva variabeln: varje gång öglan körs lägger programmet till ett till värdet i variabeln ”öglor” (**Bild 4-7**).




▲ **Bild 4-7:** Tack vare öglan räknar katten nu uppåt



## RÄKNING FRÅN NOLL

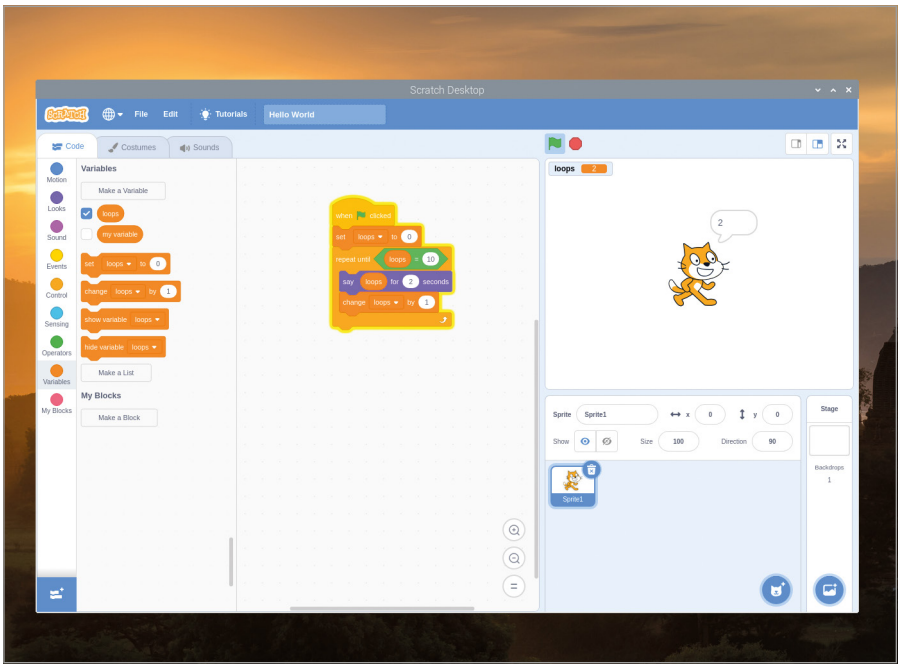
Även om öglan du har skapat går tio gånger räknar kattsprajten bara upp till nio. Det beror på att vi börjar med ett värde på noll för vår variabel. Inklusive noll och nio finns det tio siffror mellan noll och nio, så programmet stannar innan katten säger ”10”. För att ändra detta kan du ställa in variabelns initialvärde till 1 istället för 0.

Du kan göra mer med en variabel än att modifiera den. Klicka på och dra säg öglor i 2 sekunder -blocket för att bryta ut det från repetera 10 -blocket och släpp det under repetera 10 -blocket. Klicka på och dra repetera 10 -blocket till blockpaletten för att radera det, byt sedan ut det mot ett repetera tills -block och se till att blocket är anslutet till nederdelen av sätt öglor till 0 -blocket och omger de båda andra blocken i sekvensen. Klicka på kategorin Operatorer i blockpaletten, färgkodad grön, och klicka sedan på och dra det diamantformade ( ) = ( ) -blocket och släpp det på det matchande diamantformade hålet i repetera tills -blocket.


Operatorer-blocket gör det möjligt för dig att jämföra två värden, inklusive variabler. Klicka på kategorin Variabler, dra rapporteringsblocket öglor till det tomma utrymmet i ( ) = ( ) -Operatorer-blocket, klicka sedan på utrymmet med ”50” i sig och skriv siffran ”10”.


Klicka på den gröna flaggan ovanför scenområdet, så kommer du att märka att programmet fungerar på samma sätt som tidigare: kattsprajten räknar från 0 till 9 (**Bild 4-8**) och sedan

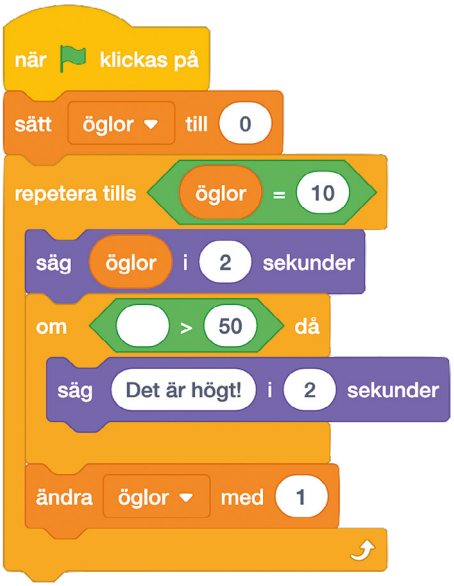
stannar programmet. Det beror på att repetera tills-blocket fungerar på exakt samma sätt som repetera 10-blocket, men istället för att räkna antalet öglor själv, jämför det värdet för variabeln ”öglor” med det värde du skrev till höger om blocket. När variabeln ”öglor” når 10 stannar programmet.



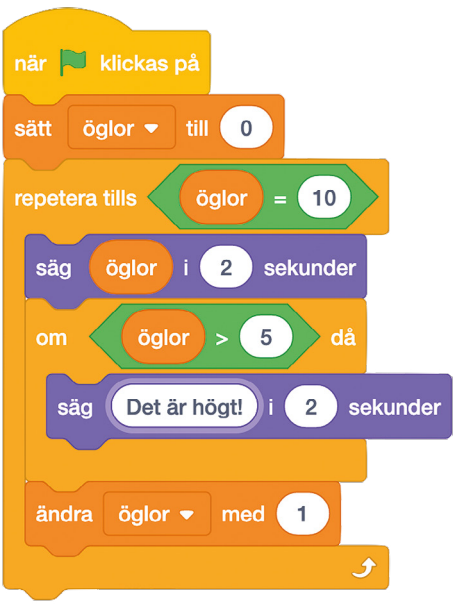

▲ **Bild 4-8:** Använda ett ”repetera tills”-block med en jämförande operator

Detta kallas för en *jämförande operator*: den bokstavligen jämför två värden. Klicka på kategorin Operatorer i blockpaletten. Där finns två andra diamantformade block ovanför och nedanför det med symbolen ”=”. De här är också jämförande operatorer: ”<” jämför två värden och utlöses när värdet till vänster är mindre än det till höger, och ”>” utlöses när värdet till vänster är större än det till höger.

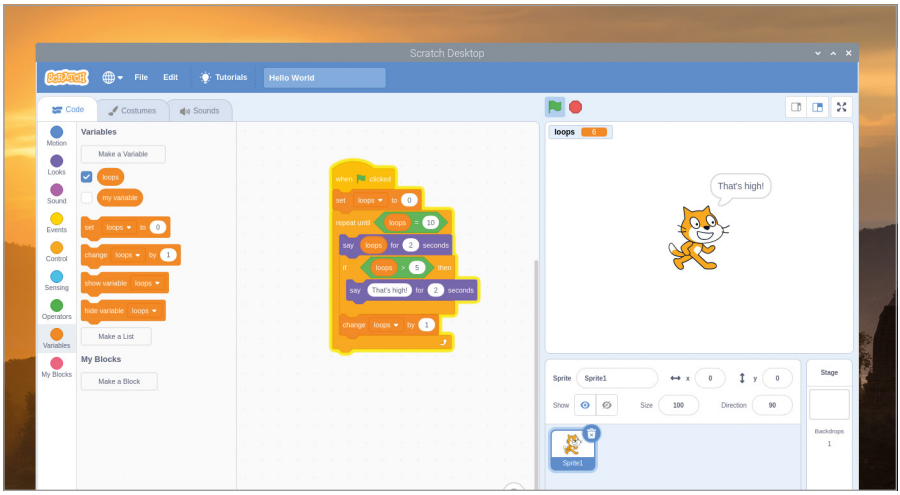
Klicka på kategorin Kontroll i blockpaletten, leta reda på om då-blocket och klicka sedan på det och dra det till kodområdet innan du släpper det direkt under säg öglor i 2 sekunder -blocket. Det omsluter automatiskt ändra öglor med 1-blocket, så klicka på det och dra för att flytta det så att det istället ansluts till nederdelen av om då-blocket. Klicka sedan på kategorin Utseende i blockpaletten och klicka på och dra ett säg Hej! i 2 sekunder -block för att släppa det inuti om då-blocket. Klicka på kategorin Operatorer i blockpaletten och klicka sedan på och dra  -blocket in i det diamantformade hålet i om då-blocket.



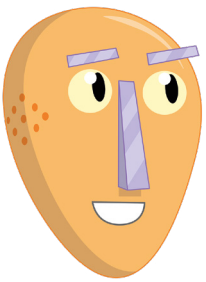

om då-blocket är ett villkorligt block, vilket innebär att blocken inuti det bara kommer att köras om ett visst villkor är uppfyllt. Klicka på kategorin Variabler i blockpaletten, dra öglor -rapporteringsblocket till det tomma utrymmet i = -blocket, klicka sedan på utrymmet med ”50” i sig och skriv siffran ”5”. Klicka slutligen på ordet ”Hej!” i säg Hej! i 2 sekunder -blocket och skriv ”Det är högt!”.



Klicka på den gröna flaggan. Först kommer programmet att fungera som tidigare med kattsprajten som räknar uppåt från noll. När siffran når 6, den första siffran som är större än 5, börjar om då-blocket att utlösas och kattsprajten kommer att kommentera hur höga siffrorna börjar bli (**Bild 4-9**). Grattis: du kan nu arbeta med variabler och villkor!




▲ **Bild 4-9:** Katten kommenterar när den når siffran 6



**UTMANING:** HÖGT OCH LÅGT

Hur skulle du kunna ändra programmet så att kattsprajten kommenterar hur låga siffrorna under 5 är istället? Kan du ändra så att katten kommenterar både höga och låga siffror? Experimentera med blocket om då annars för att göra detta enklare!

## Projekt 1: Reaktionstimer för astronaut

Nu när du förstår hur Scratch fungerar är det dags att göra något lite mer interaktivt: en reaktionstimer som är utformad för att hedra den brittiska ESA-astronauten Tim Peake och hans tid ombord på den internationella rymdstationen.

Spara ditt befintliga program, om du vill behålla det, och öppna sedan ett nytt projekt genom att klicka på Arkiv och Nytt. Innan du börjar ska du namnge projektet genom att klicka på Arkiv och ”Spara till din dator”. Kalla det för ”Reaktionstimer för astronaut”.

Detta projekt bygger på två bilder – en som scenbakgrund, en som en sprajt – som inte ingår i Scratchs inbyggda resurser. För att ladda ner dem ska du klicka på hallonikonen för att läsa in Raspberry Pi OS-menyn, flytta muspekaren till internet och klicka på webbläsaren Chromium. När webbläsaren har lästs in skriver du **rpf.io/astronaut-backdrop** i adressfältet, följt av **ENTER**-tangenten. Högerklicka på bilden av rymden och klicka på ”Spara bild som...” och klicka sedan på knappen Spara (**Bild 4-10**). Klicka bakåt i adressfältet och skriv **rpf.io/astronaut-sprite** följt av **ENTER**-tangenten.

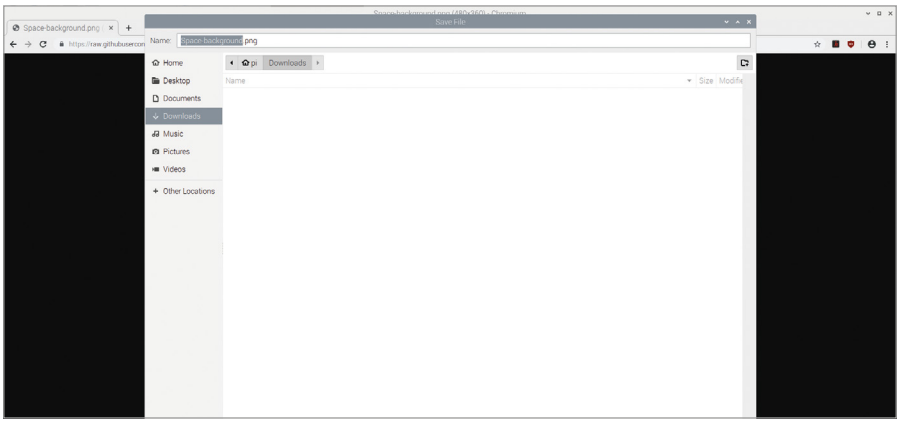
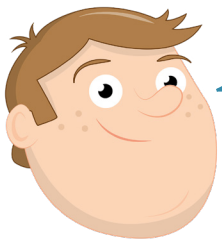



**Bild 4-10:** Spara bakgrundsbilden

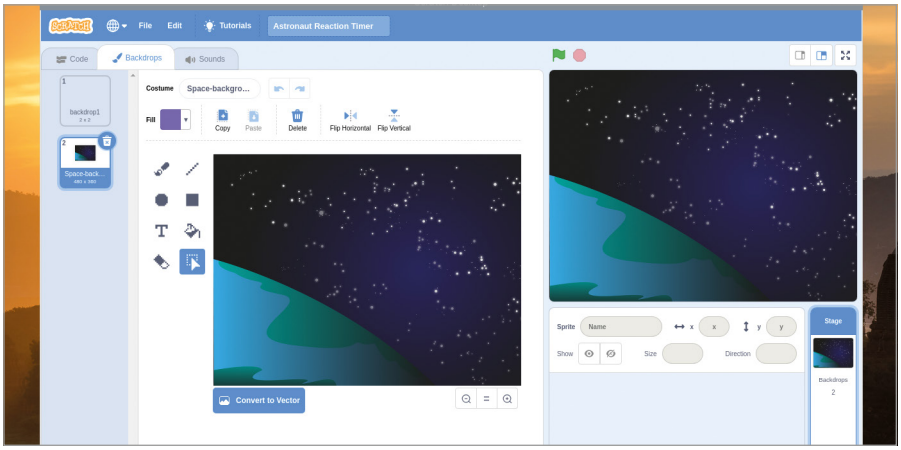
Högerklicka på bilden av Tim Peake och klicka på ”Spara bild som...”, välj sedan mappen Downloads och klicka på knappen Spara. Med dessa två bilder sparade kan du stänga Chromium eller lämna den öppen och använda aktivitetsfältet för att växla tillbaka till Scratch 3.



## ANVÄNDARGRÄNSSNITT

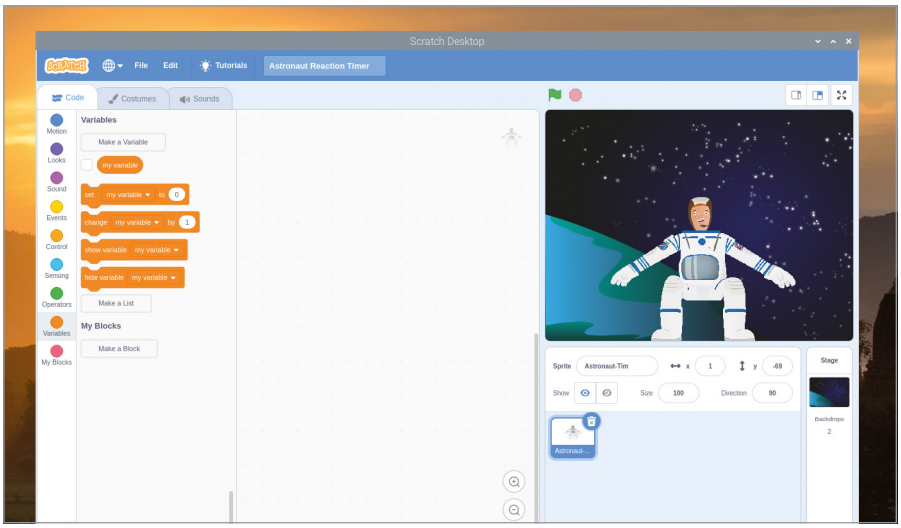
Om du har följt med i detta kapitel från början bör du känna till användargränssnittet för Scratch 3. Följande projektinstruktioner förlitar sig på att du vet var sakerna finns. Om du glömmer var du hittar något kan du gå tillbaka till bilden av användargränssnittet i början av detta kapitel för att få en påminnelse.

Högerklicka på kattsprajten i listan och klicka på ”radera”. Håll muspekaren över ikonen Välj en bakgrund och klicka sedan på ikonen Ladda upp bakgrund från listan som visas. Leta upp filen **Space-background.png** i mappen Downloads, klicka på den för att välja den och klicka sedan på OK. Den vanliga vita scenbakgrunden ändras till bilden av rymden och kodområdet kommer att ersättas med bakgrundsområdet (**Bild 4-11**). Här kan du rita över bakgrunden, men för tillfället klickar du bara på fliken märkt Kod högst upp i Scratch 3-fönstret.




▲ **Bild 4-11:** Rymdbakgrunden visas på scenen

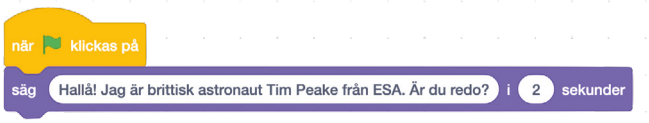
Ladda upp den nya sprajten genom att hålla muspekaren över ikonen Välj en sprajt och sedan klicka på ikonen Ladda upp sprajt högst upp i listan som visas. Leta upp filen **Astronaut-Tim.png** i mappen Downloads, klicka på den för att välja den och klicka sedan på OK. Sprajten visas automatiskt på scenen, men kanske inte i mitten: klicka på och dra den med musen och släpp den så att den hamnar nära mitten av den nedre delen av scenen (**Bild 4-12**).



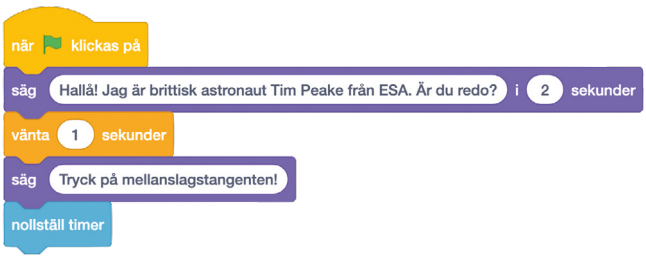

▲ **Bild 4-12:** Dra astronautsprajten till mitten av den nedre delen av scenen

Med denn nya bakgrunden och sprajten på plats är du redo att skapa programmet. Börja med att skapa en ny variabel som kallas ”tid” och se till att ”För alla sprajtar” är markerad innan du klickar på OK. Klicka på sprajten – antingen på scenen eller i sprajtrutan – för

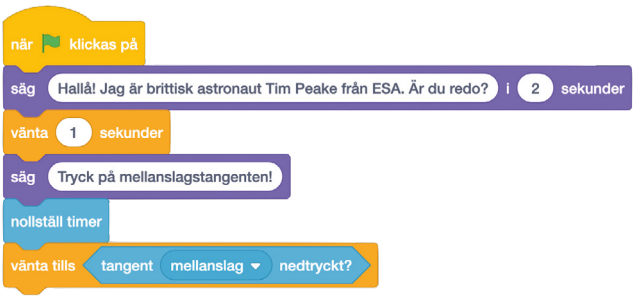
att välja den och lägg sedan till ett [när klickas på]-block från kategorin Händelser till kodområdet. Lägg sedan till ett [säg Hej! i 2 sekunder]-block från kategorin Utseende och klicka sedan på det för att ändra det så att det säger ”Hallå! Jag är brittisk astronaut Tim Peake från ESA. Är du redo?”




Lägg till ett [vänta 1 sekunder]-block från kategorin Kontroll och sedan ett [säg Hej!]-block. Ändra detta block så att det säger ”Tryck på mellanslagstangenten!” och lägg sedan till ett [nollställ timer]-block från kategorin Känna av. Detta styr en speciell variabel som är inbyggd i Scratch för att tajma saker och kommer att användas för att tajma hur snabbt du kan reagera i spelet.

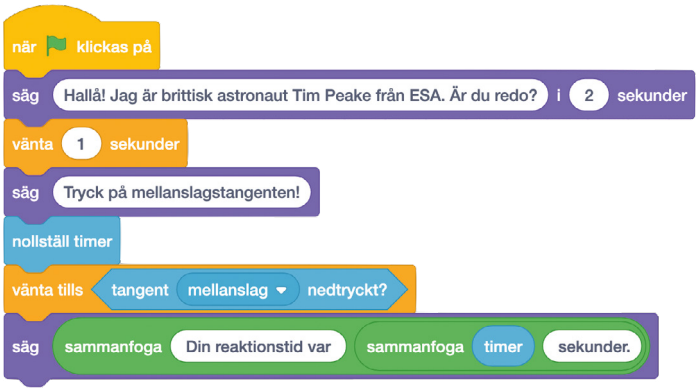



Lägg till ett [vänta tills]-Kontroll-block och dra sedan ett [tangent mellanslag nedtryckt?]-Känna av-block till det vita utrymmet. Det kommer att pausa programmet tills du trycker på tangenten **MELLANSLAG** på tangentbordet, men timern fortsätter att gå – den räknar exakt hur lång tid det är mellan meddelandet som säger ”Tryck på mellanslagstangenten!” till dig och att du faktiskt trycker på tangenten **MELLANSLAG**.




Nu är det dags för Tim att berätta för dig hur lång tid det tog för dig att trycka på tangenten **MELLANSLAG**, men på ett sätt som är lätt att läsa. För att göra det behöver du ett [sammanfoga]-Operatorer-block. Detta kräver två värden, inklusive variabler, och sammanfogar dem det ena efter det andra – så kallad *sammanfogning*.

Börja med ett säg Hej! -block, dra och släpp sedan ett sammanfoga -Operatorer-block över ordet ”Hallå!”. Klicka på ”äpple” och skriv ”Din reaktionstid var”, se till att lägga till ett mellanslag i slutet och dra sedan ett annat sammanfogningsblock över den övre delen av ”banan” i den andra rutan. Dra ett timer -rapporteringsblock från kategorin Känna av till det som nu är mittrutan och skriv ”sekunder.” i den sista rutan – se till att inkludera ett mellanslag i början.




Slutligen drar du ett sätt min variabel till 0 -Variabel-block till slutet av sekvensen. Klicka på rullgardinspilen bredvid ”min variabel” och klicka på ”tid” i listan och ersätt sedan ”0” med ett timer-rapporteringsblock från kategorin Känna av. Spelet är nu redo för test genom att du klickar på den gröna flaggan ovanför scenen. Gör dig redo, och så snart du ser meddelandet ”Tryck på mellanslagstangenten!” ska du trycka på tangenten **MELLANSLAG** så fort du kan (**Bild 4-13**). Se om du kan slå vår bästa poäng!

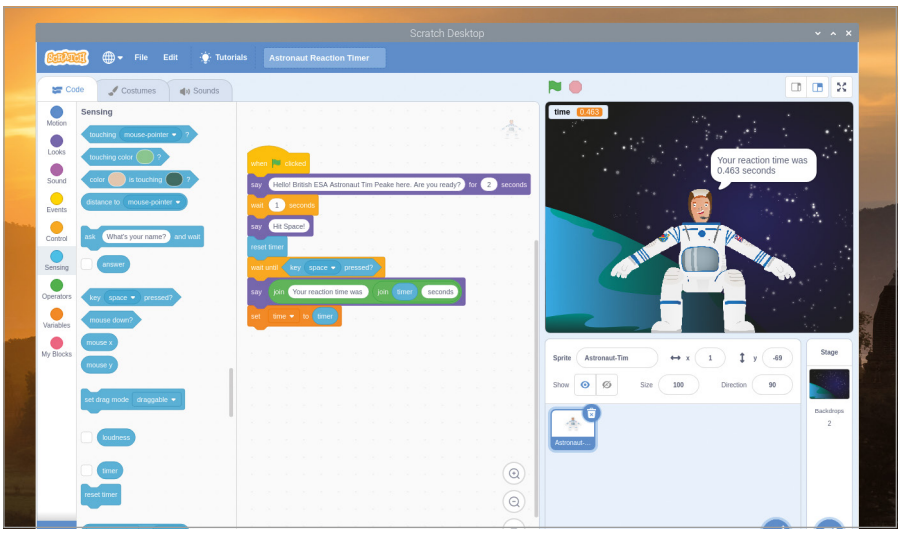
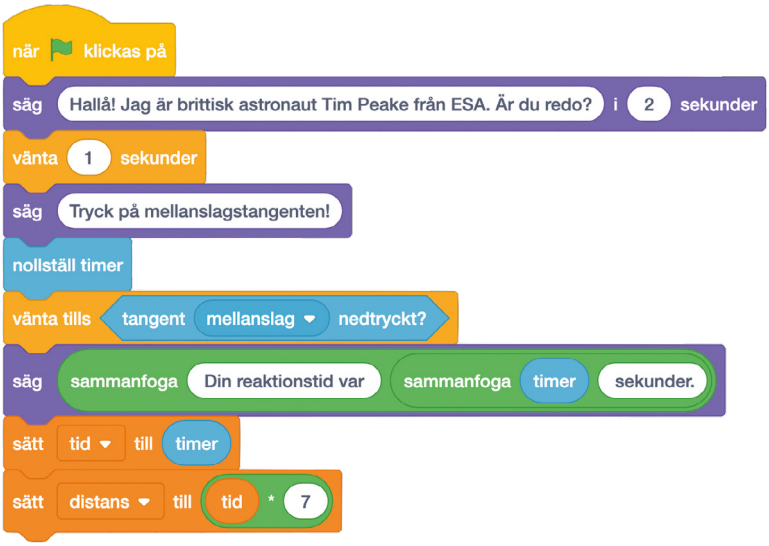


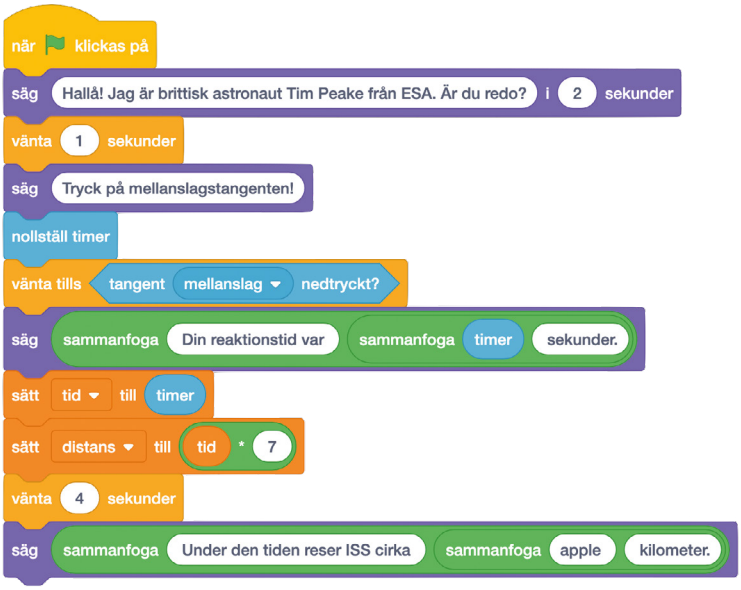

**Bild 4-13:** Dags att spela spelet!

Du kan förlänga projektet ytterligare genom att låta det beräkna ungefär hur långt den internationella rymdstationen har rest under den tid det tog för dig att trycka på tangenten **MELLANSLAG**, baserat på stationens publicerade hastighet på sju kilometer per sekund. Skapa först en ny variabel som heter ”distans”. Lägg märke till hur blocken i kategorin Variabler automatiskt ändras för att visa den nya variabeln, men de befintliga tid -variabelblocken i programmet förblir desamma.

Lägg till ett sätt distans till 0 -block och dra sedan ett ( ) * ( ) -Operatorer-block – som indikerar multiplikation – över ”0”. Dra ett tid -rapporteringsblock över det första mellanslaget och skriv sedan in siffran ”7” i det andra mellanslaget. När du är klar står det sätt distans till tid * 7 i det kombinerade blocket. Detta tar den tid det tog dig att trycka på tangenten **MELLANSLAG** och multiplicerar den med sju för att få avståndet i kilometer som ISS har rest.

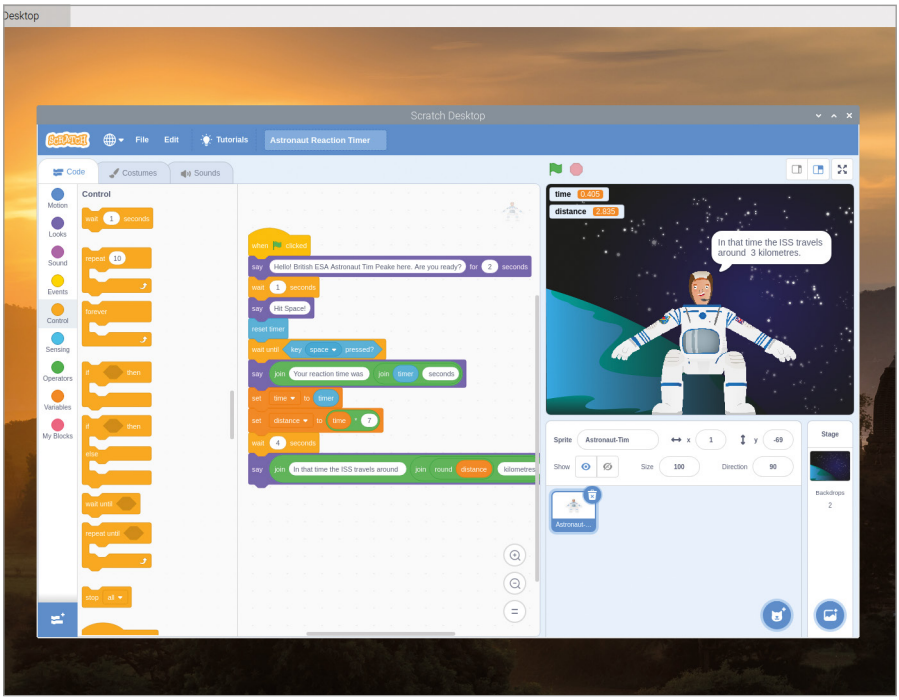



Lägg till ett vänta 1 sekunder -block och ändra det till ”4”. Slutligen ska du dra ytterligare ett säg Hej! -block till slutet av sekvensen och lägga till två sammanfoga -block, precis som du gjorde tidigare. I det första utrymmet, över ”äpple”, ska du skriva ”Under den tiden reser ISS cirka”. Kom ihåg att inkludera mellanslaget i slutet. I ”banan”-utrymmet ska du skriva ”kilometer.”. Kom ihåg mellanrummet i början.




Slutligen drar du ett **avrunda**-Operatorer-block till det mellersta utrymmet och drar sedan ett **distans**-rapporteringsblock till det nya utrymme det skapar. **avrunda**-blocket avrundar siffror uppåt eller nedåt till närmaste heltal, så i stället för ett hyperexakt men svårläst antal kilometer får du ett lättläst heltal.

Klicka på den gröna flaggan för att köra programmet och se hur långt ISS reser på den tid det tar för dig att trycka på tangenten **MELLANSLAG**. Kom ihåg att spara programmet när du är klar så att du enkelt kan läsa in det igen i framtiden utan att behöva börja från början!




▲ **Bild 4-14:** Tim berättar hur långt ISS har rest

## UTMANING: VEM ÄR SNABB?

Förutom astronaut, vilka andra yrken kräver blixtsnabba reflexer? Kan du rita egna sprajtar och bakgrunder för att visa ett av dessa yrken?

# Projekt 2: Konstsim

De flesta spel använder mer än en enda knapp, och detta projekt visar det genom att erbjuda tvåknappskontroll med tangenterna ← och → på tangentbordet.

## ONLINE-PROJEKT

Detta projekt finns också tillgängligt online på **rpf.io/synchro-swimming**

Skapa ett nytt projekt och spara det som ”Konstsim”. Klicka på scenen i scenkontrollsektionen och klicka sedan på fliken Bakgrunder längst upp till vänster. Klicka på knappen Gör till bitmapp under bakgrunden. Välj en vattenliknande blå färg från paletten Fyll, klicka på ikonen Fyll och klicka sedan på den rutiga bakgrunden för att fylla den med blått (**Bild 4-15**).

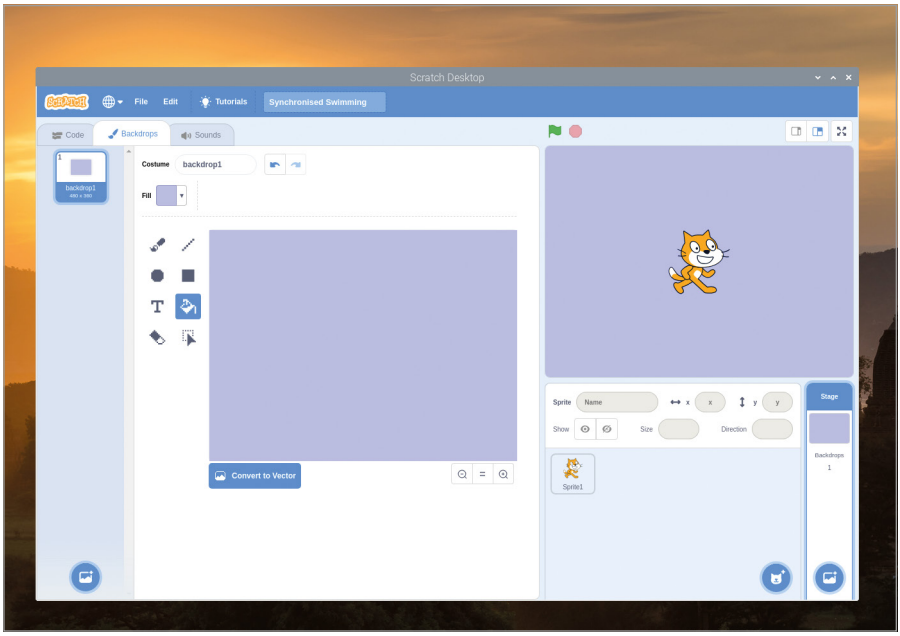



**Bild 4-15:** Fyll bakgrunden med en blå färg

Högerklicka på kattsprajten i listan och klicka på ”radera”. Klicka på ikonen Välj en sprajt för att se en lista med inbyggda sprajtar. Klicka på kategorin Djur, sedan på ”Cat Flying” (**Bild 4-16**) och sedan på OK. Denna sprajt fungerar även bra för simprojekt.


**Bild 4-16:** Välj en sprajt från biblioteket

Klicka på den nya sprajten och dra sedan två **när mellanslag tangenten trycks ned** -Händelser-block till kodområdet. Klicka på den lilla nedåtpilen bredvid ordet ”mellanslag” i det första blocket och välj ”vänsterpil” i listan över möjliga alternativ. Dra ett **rotera 15 grader** -Rörelse-block under **när vänsterpil tangenten trycks ned** -blocket, gör sedan detsamma med det

andra Händelser-blocket, förutom att du väljer ”högerpil” från listan och använder ett rotera 15 grader-Rörelseblock..


Tryck på ← eller → för att testa programmet. När du gör det ser du kattsprajten vända i den riktning du väljer på tangentbordet. Lägg märke till hur du inte behövde klicka på den gröna flaggan den här gången. Detta beror på att de utlösarblock för Händelser som du har använt är aktiva hela tiden, även om programmet inte körs i normal mening.

Gör samma steg två gånger igen, men den här gången väljer du ”uppåtpil” och ”nedåtpil” för Händelser-utlösarblocken och sedan gå 10 steg och gå -10 steg för Rörelser-blocken. Tryck på piltangenterna nu så kan du se att katten kan vända och simma framåt och bakåt också!


För att göra kattsprajtens rörelser mer realistiska kan du ändra hur den ser ut, vilket i Scratch-termer kallas för *klädsel*. Klicka på kattsprajten och sedan på fliken Klädslar ovanför blockpaletten. Klicka på klädseln ”cat flying-a” och klicka på ikonen X-i-en-papperskorg som

visas längst upp till höger för att radera den. Klicka sedan på klädseln ”cat flying-b” och använd namnrutan högst upp för att byta namn på den till ”höger” (**Bild 4-17**).

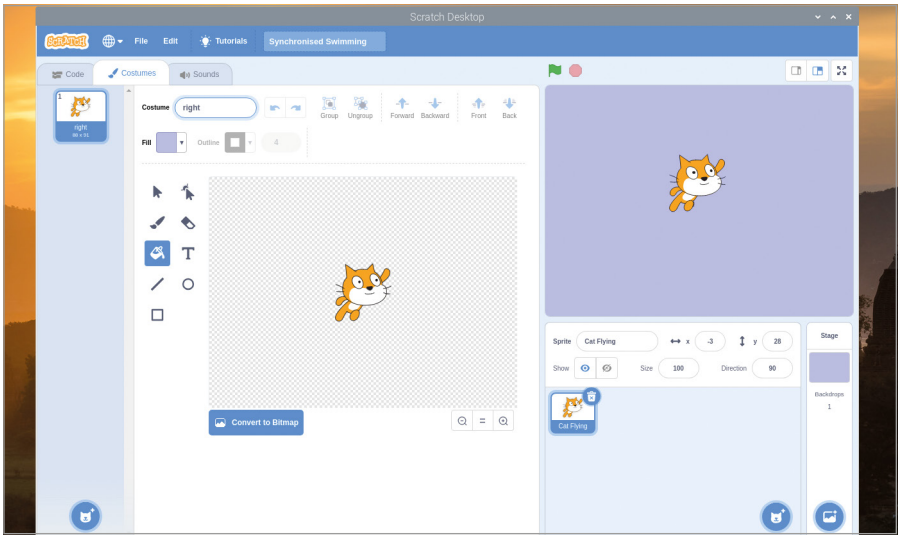



**Bild 4-17:** Byt namn på klädseln till ”höger”

Högerklicka på den nyligen omdöpta kostymen ”höger” och klicka på ”kopiera” för att skapa en kopia. Klicka på kopian för att markera den, klicka på ikonen Välj , klicka på Vänd vågrätt och byt namn på den till ”vänster” (**Bild 4-18**). Du kommer nu att ha två ”klädslar” till sprajten, som är exakta spegelbilder: en som heter ”höger” med katten vänd åt höger och en som heter ”vänster” med katten vänd åt vänster.

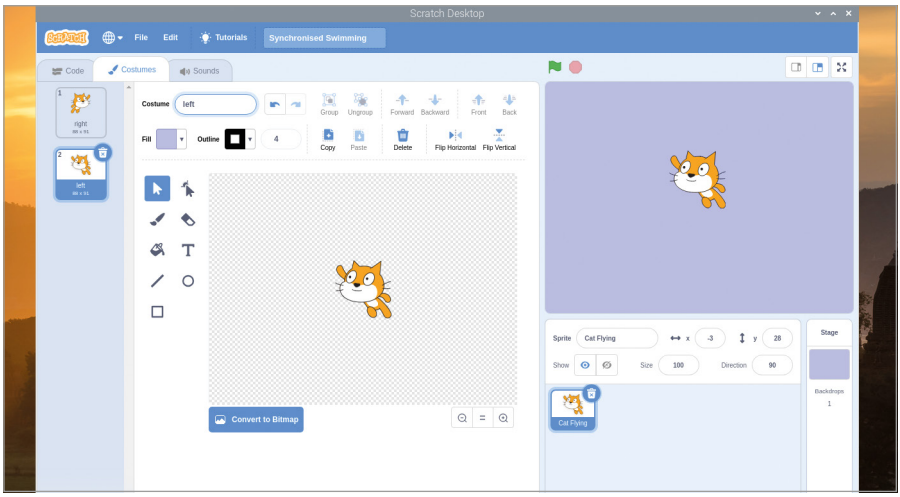



**Bild 4-18:** Kopiera klädseln, vänd den och döp den till ”vänster”

Klicka på fliken Kod ovanför klädselområdet. Dra sedan två **ändra klädsel till vänster** -Utseende-block under Händelser-blocken vänsterpil och högerpil och ändra den under högerpilen så att det står **ändra klädsel till höger**. Prova piltangenterna igen. Katten verkar nu vända sig mot den riktning den simmar i.

För konstsim i olympisk stil behöver vi dock fler simmare, och vi behöver ett sätt att återställa kattsprajtens position. Lägg till ett när klickas på-Händelser-block och lägg sedan till ett gå till x: 0 y: 0-Rörelse-block undertill – ändra värdena vid behov – och ett peka i riktning 90 -Rörelse-block. Nu när du klickar på den gröna flaggan flyttas katten till mitten av scenen och pekar åt höger.

Om du vill skapa fler simmare lägger du till ett repetera 6-block – ändra från standardvärdet ”10” – och lägger till ett skapa klon av mig själv-Kontroll-block inuti det. För att göra så att alla simmare inte simmar i samma riktning lägger du till ett rotera 60 grader -block ovanför skapa klon -blocket men fortfarande inne i repetera 6-blocket. Klicka på den gröna flaggan och prova piltangenterna nu för att se hur simmarna vaknar till liv!

För att göra den olympiska känslan fullständig måste du lägga till lite musik. Klicka på fliken Ljud ovanför blockpaletten och klicka sedan på ikonen ”Välj ett ljud” . Klicka på kategorin Loopar och bläddra sedan igenom listan (**Bild 4-19**) tills du hittar lite musik du gillar. Vi har valt ”Dance Around”. Klicka på knappen OK för att välja musiken och klicka sedan på fliken Kod för att öppna kodområdet igen.

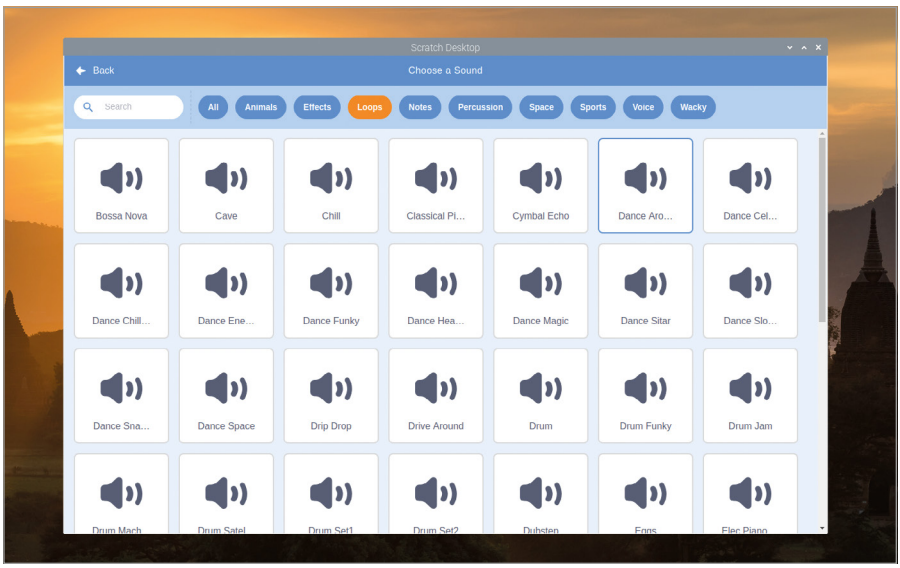



**Bild 4-19:** Välj en musikloop från ljudbiblioteket

Lägg till ännu ett när klickas på-Händelser-block i kodområdet och lägg sedan till ett för alltid-Kontroll-block. Inuti Kontroll-blocket ska du lägga till ett spela ljudet Dance Around tills färdigt-block – leta upp namnet på det musikstycke du valde – och klicka på den gröna flaggan för att testa det nya programmet. Om du vill stoppa musiken klickar du på den röda oktagonen för att stoppa programmet och stänga av ljudet!

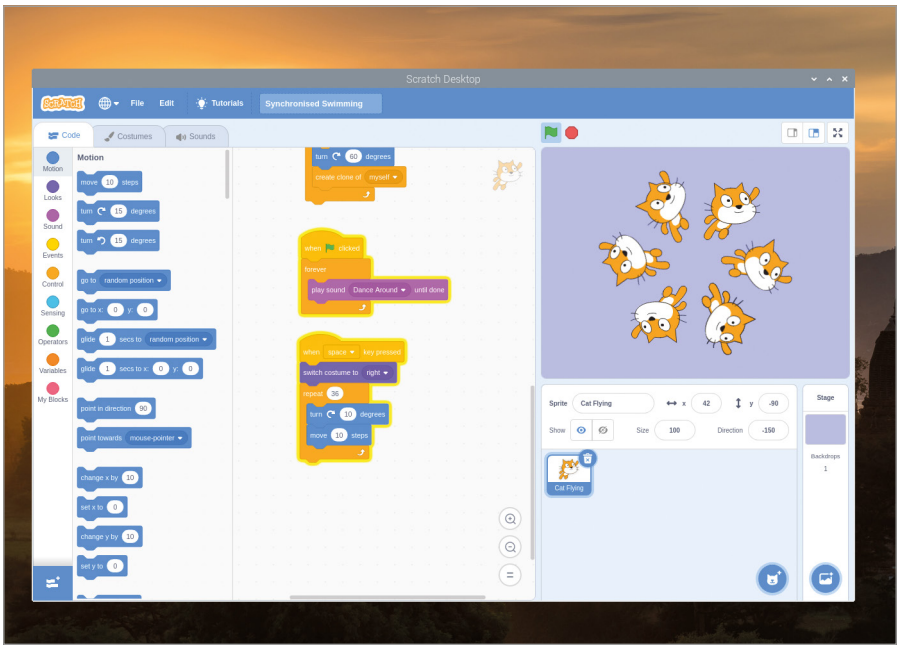

Slutligen kan du simulera en fullständig dansrutin genom att lägga till en ny händelseutlösare i programmet. Lägg till ett **när mellanslag tangenten trycks ned** -Händelser-block och sedan ett **ändra klädsel till höger** -block. Under detta ska du lägga till ett **repetera 36** -block – kom ihåg att ändra värdet från standardvärdet – och inuti detta ett **rotera 10 grader** -block och ett **gå 10 steg** -block.



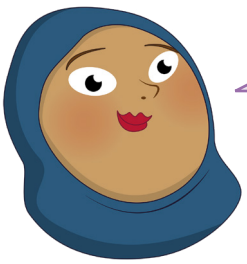





Klicka på den gröna flaggan för att starta programmet och tryck sedan på tangenten **MELLANSLAG** för att prova den nya rutinen (**Bild 4-20**, nästa sida)! Glöm inte att spara programmet när du är klar.




▲ **Bild 4-20:** Den färdiga rutinen med konstsim



**UTMANING:** ANPASSA RUTIN

Kan du skapa en egen konstsimrutin med öglor? Vad skulle du behöva ändra om du vill ha fler eller färre simmare? Kan du lägga till flera simrutiner som kan utlösas med olika tangenter på tangentbordet?

## Projekt 3: Bågskyttespel

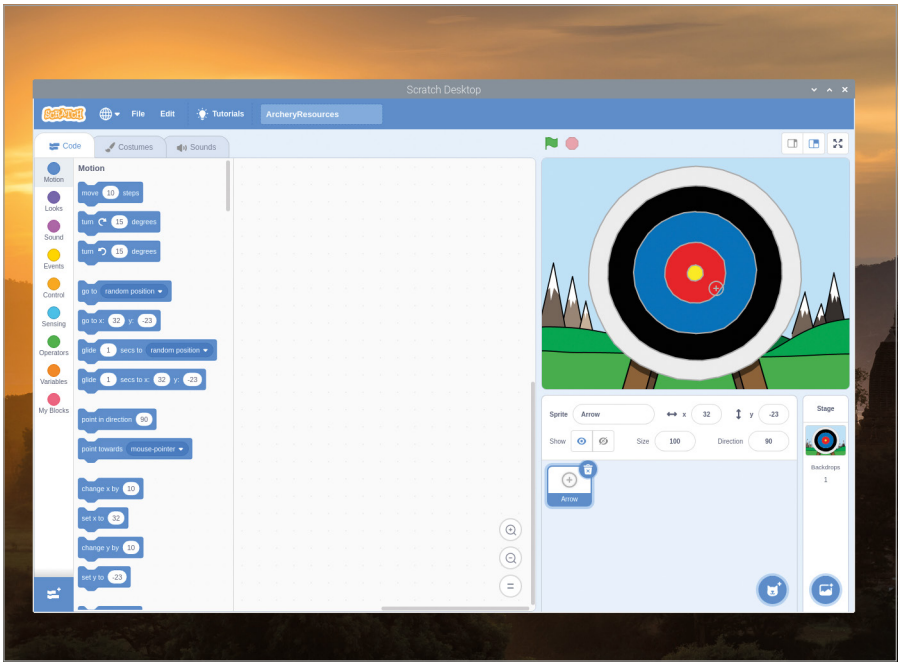
Nu när du börjar bli något av expert på Scratch är det dags att jobba med något lite mer utmanande: ett bågskyttespel, där spelaren måste träffa ett mål med en slumpmässigt svängande pil och båge.

**ONLINE-PROJEKT**

Detta projekt finns också tillgängligt online på **rpf.io/archery**

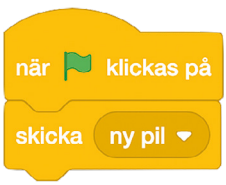
Börja med att öppna Chromium-webbläsaren och skriva **rpf.io/p/en/archery-go** följt av **ENTER**-tangenten. Resurserna för spelet hämtas som en zip-fil, så du måste packa upp den (högerklicka på den och välj Extrahera här). Växla tillbaka till Scratch 3 och klicka på Arkiv-menyn följt av ”Ladda

upp från din dator”. Klicka på **ArcheryResources.sb3** följt av knappen Öppna. Du kommer att bli tillfrågad om du vill ersätta innehållet i ditt aktuella projekt: om du inte har sparat ändringarna klickar du på Avbryt och sparar dem, annars klickar du på OK.




▲ **Bild 4-21:** Resursprojekt laddat för bågskyttespelet

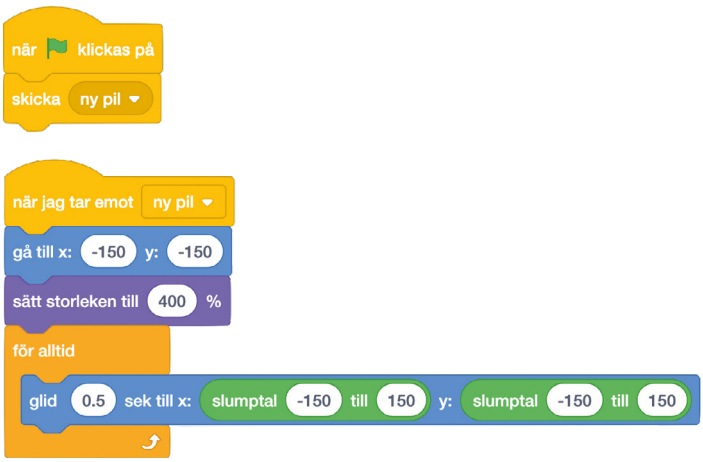
Projektet du just laddat innehåller en bakgrund och en sprajt (**Bild 4-21**), men ingen av den faktiska kod som behövs för att skapa ett spel: att lägga till den är ditt jobb. Börja med att lägga till ett när klickas på-block och sedan ett skicka meddelande1-block. Klicka på nedåtpilen i slutet av blocket och sedan på ”Nytt meddelande” och skriv in ”ny pil” innan du klickar på OK-knappen. Nu står det skicka ny pil i blocket.



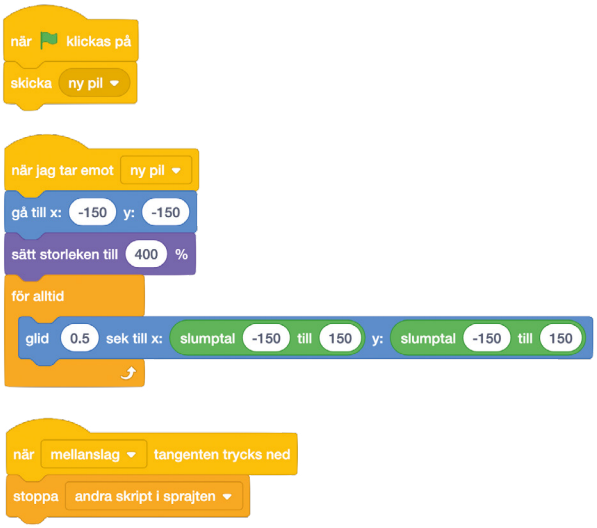

Ett skickande är ett meddelande från en del av programmet som kan tas emot av en annan del av programmet. För att få det att faktiskt göra något ska du lägga till ett när jag tar emot meddelande1-block och ändra det igen så att det står när jag tar emot ny pil. Den här gången kan du bara klicka på nedåtpilen och välja ”ny pil” från listan. Du behöver inte skapa meddelandet på nytt.

Under **när jag tar emot ny pil** -blocket ska du lägga till ett **gå till x: -150 y: -150** -block och ett **sätt storleken till 400 %** -block. Kom ihåg att dessa inte är standardvärdena för block, så du måste ändra dem när du har dragit dem till kodområdet. Klicka på den gröna flaggan för att se vad du har gjort hittills: pilsprajten, som spelaren använder för att sikta mot målet, hoppar längst ner till vänster på scenen och fyrdubblas i storlek.

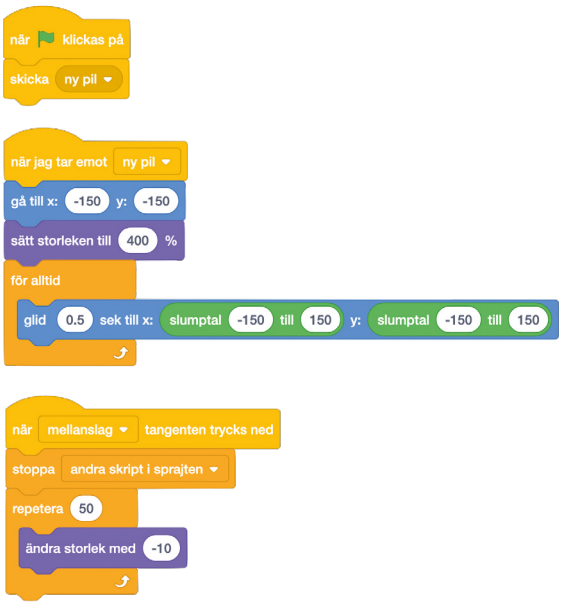

Om du vill ge spelaren en utmaning kan du lägga till rörelse som simulerar svängning när bågen dras och bågskytten siktar. Dra ett **för alltid** -block, följt av ett **glid 1 sek till x: -150 y: -150** -block. Redigera den första vita rutan så att det står ”0,5” istället för ”1”, placera sedan ett **slumptal -150 till 150** -Operatorer-block i var och en av de andra två vita rutorna. Det innebär att pilen kommer att driva runt på scenen i slumpmässig riktning, på ett slumpmässigt avstånd, vilket gör det mycket svårare att träffa målet!




Klicka på den gröna flaggan igen så ser du vad det blocket gör: pilsprajten driver nu runt på scenen och täcker olika delar av målet. För närvarande har du dock inget sätt att skjuta pilen mot målet. Dra ett **när mellanslag tangenten trycks ned** -block till kodområdet följt av ett **stoppa alla** -Kontroll-block. Klicka på nedåtpilen i slutet av blocket och ändra det till ett **stoppa andra skript i sprajten** -block.

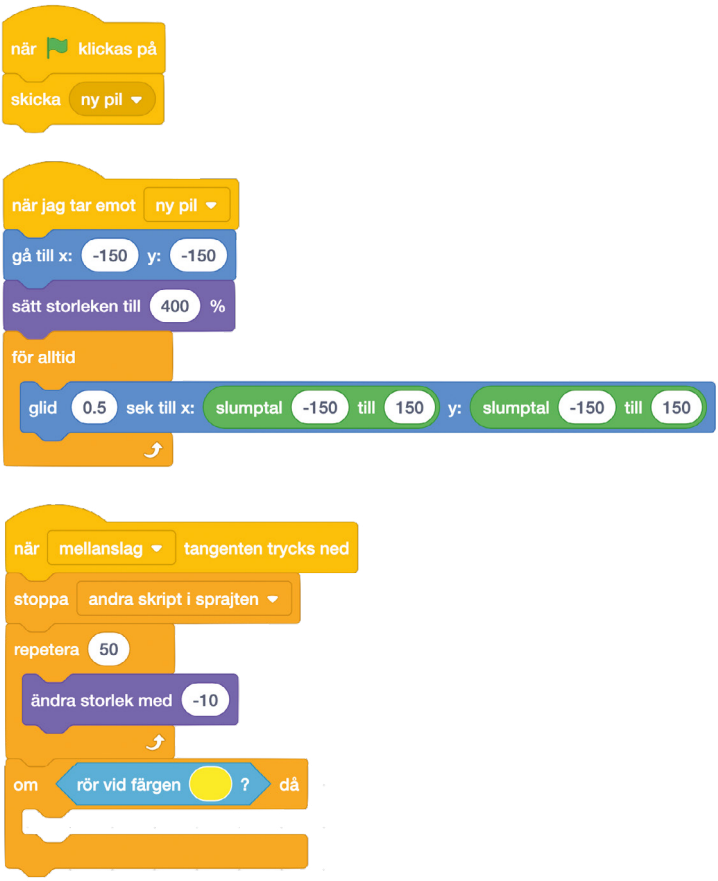



Om du stoppade programmet för att lägga till de nya blocken klickar du på den gröna flaggan för att starta det igen och trycker sedan på tangenten **MELLANSLAG**. Pilsprajten slutar då röra sig. Det är en början, men du måste se till att det ser ut som om pilen flyger till målet. Lägg till ett repetera 50 -block följt av ett ändra storlek med -10 -block och klicka sedan på den gröna flaggan för att testa spelet igen. Den här gången verkar pilen flyga bort från dig och mot målet.

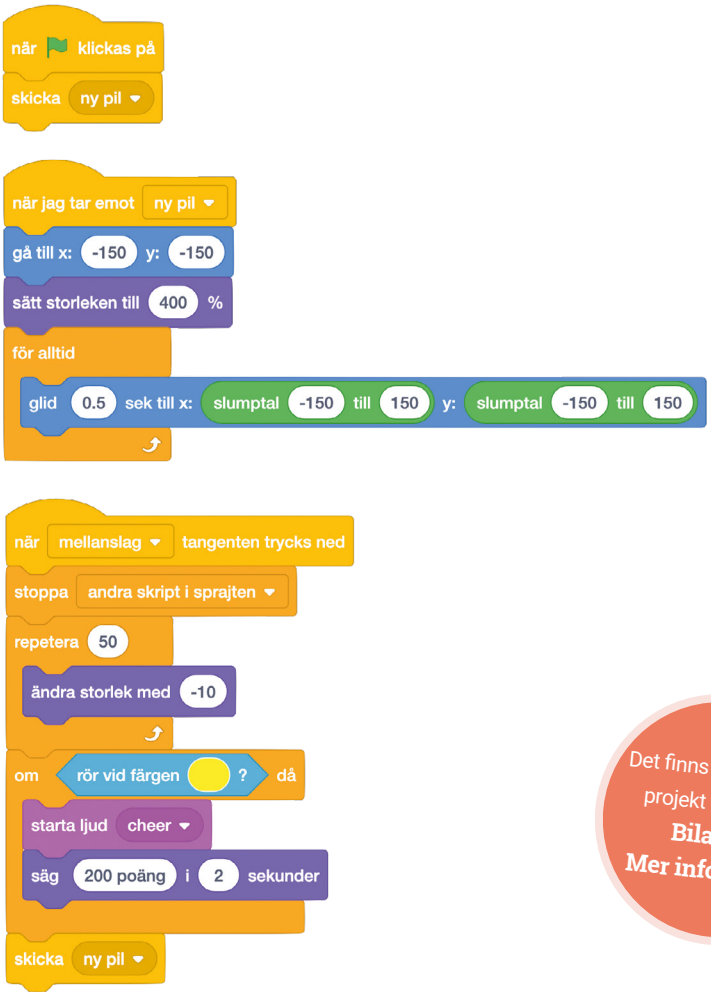



För att göra spelet roligt måste du lägga till ett sätt att räkna poäng. Fortfarande i samma blockstapel lägger du till ett om då -block och ser till att det är under repetera 50 -blocket och

inte inuti det, med ett rör vid färgen?-Känna av-block i den diamantformade luckan. För att välja rätt färg klickar du på färgad ruta i slutet av Känna av-blocket, sedan på pipettikonen och klicka sedan på den gula mittpunkten på måltavlan på scenen.




För att låta spelaren veta att det blev en träff ska du lägga till ett starta ljud cheer -block och ett säg 200 poäng i 2 sekunder -block inuti om då -blocket. Slutligen ska du lägga till ett skicka ny pil -block längst ner i blockstapeln, under och utanför om då -blocket, för att ge spelaren ytterligare en pil varje gång denne skjuter iväg en. Klicka på den gröna flaggan för att starta spelet och försök att träffa den gula mittpunkten: när du gör det, kommer du att belönas med jubel från publiken och 200 poäng!




Det finns fler Scratch-projekt att prova i **Bilaga D: Mer information**

Spelet fungerar, men är lite utmanande. Använd det du har lärt dig i det här kapitlet och försök att bygga ut det så att man kan få poäng när man träffar andra delar av målet än mittpunkten: 100 poäng för rött, 50 poäng för blått och så vidare.

**UTMANING:** KAN DU FÖRBÄTTRA DET?

Hur skulle du göra spelet enklare? Hur skulle du göra det svårare? Kan du använda variabler för att öka spelarens poäng när de avfyrar fler pilar? Kan du lägga till en nedräkningstimer för att sätta mer press på spelaren?

## Kapitel 5

# Programmering med Python

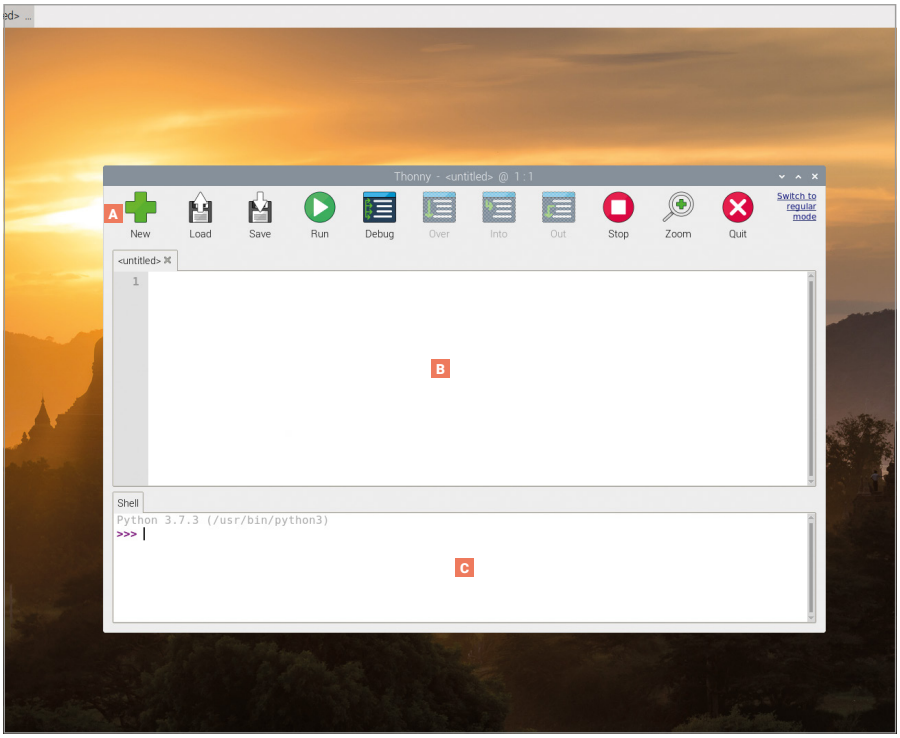
**Nu när du har fått grepp om Scratch ska vi visa dig hur man gör textbaserad kodning med Python**

Guido van Rossums Python, som döptes efter humorgruppen Monty Python, har utvecklats från ett hobbyprojekt som först släpptes för allmänheten 1991 till ett mycket omtyckt programmeringsspråk som driver ett brett spektrum av projekt. Till skillnad från Scratchs visuella miljö är Python textbaserat: du skriver instruktioner med ett förenklat språk och specifikt format som datorn sedan utför.

Python är ett utmärkt nästa steg för dem som redan har använt Scratch och erbjuder ökad flexibilitet och en mer ”traditionell” programmeringsmiljö. Det betyder inte att det är svårt att lära sig. Med lite övning kan vem som helst skriva Python-program, allt från enkla beräkningar till förvånansvärt komplicerade spel.

Detta kapitel bygger på termer och koncept som har introducerats i **Kapitel 4, Programmering med Scratch 3**. Om du inte har arbetat igenom övningarna i det kapitlet ännu, kommer du att märka att det här kapitlet är lättare att följa om du går tillbaka och gör det först.

# Introduktion av Thonny Python IDE




**A** **Verktygsfält** – Thonnys ”Simple Mode”-gränssnitt har ett fält med vänliga ikoner som meny, så att du kan skapa, spara, läsa in och köra Python-programmen samt testa dem på olika sätt.

**B** **Skriptområde** – Skriptområdet är den plats där Python-programmen skrivs, och det är uppdelat i ett huvudområde för programmet och en liten sidomarginal för att visa radnummer.

**C** **Python Shell** – Python Shell gör det möjligt för dig att skriva enskilda instruktioner som sedan körs så snart du trycker på **ENTER**-tangenten och ger också information om program som körs.

## THONNY-VERSIONER

Thonny har två gränssnittsversioner: ”Regular Mode” och ett ”Simple Mode” som är bättre för nybörjare. I det här kapitlet används Simple Mode, som läses in som standard när du öppnar Thonny från avsnittet Programmering i hallonmenyn.

# Ditt första Python-program: Hej, världen!

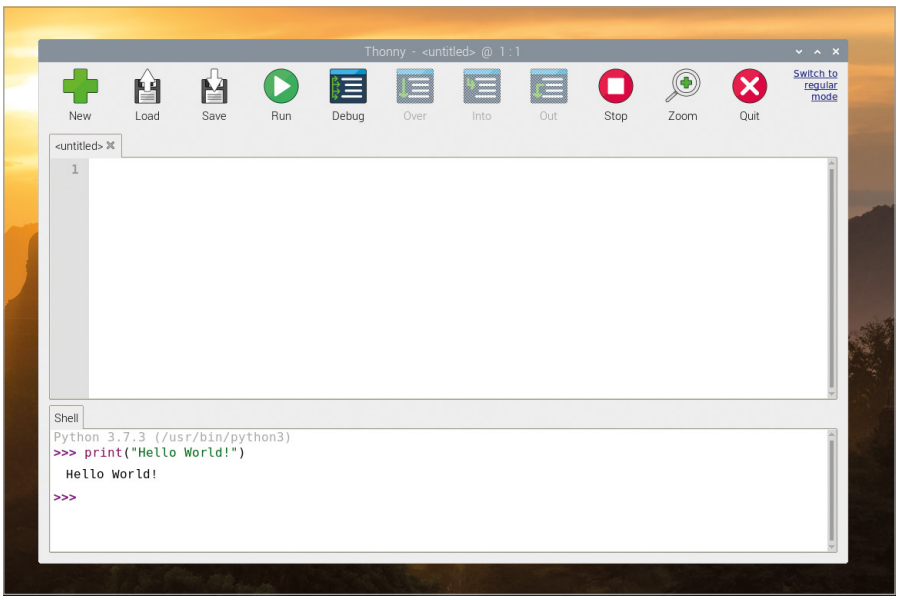
Precis som alla andra förinstallerade program på Raspberry Pi finns Thonny i menyn: klicka på hallonikonen, flytta markören till avsnittet Programmering och klicka på Thonny Python IDE. Efter några sekunder läses Thonnys användargränssnitt in (Simple Mode som standard).

Thonny är ett paket som kallas en *integrerad utvecklingsmiljö (IDE)*, ett namn som låter komplicerat men som har en enkel förklaring: det samlar ihop, eller *integrerar*, alla olika verktyg du behöver för att skriva, eller *utveckla*, programvara till ett enda användargränssnitt, eller *miljö*. Det finns många IDE:er tillgängliga. Några har stöd för många olika programmeringsspråk medan andra, som Thonny, fokuserar på att stödja ett enda språk.

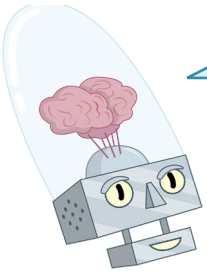
Till skillnad från Scratch, som ger dig visuella byggstenar som grund för programmet, är Python ett mer traditionellt programmeringsspråk där allt skrivs ner. Starta ditt första program genom att klicka på Shell-tPython shell area, eller skalområdet, längst ner i Thonny-fönstret och skriv sedan följande instruktion innan du trycker på **ENTER**-tangenten:

```
print("Hej, världen!")
```

När du trycker på **ENTER** ser du att programmet börjar köras direkt: Python kommer att svara i samma shell-kalområde med meddelandet ”Hej, världen” (**Bild 5-1**), precis som du bad om. Detta beror på att skalet är en direkt linje till Python-*tolken*, vars uppgift är att titta på instruktionerna och *tolka* vad de innebär. Detta kallas *interaktivt läge*, och du kan se det som en konversation ansikte mot ansikte med någon: så snart du har sagt vad du vill svarar den andra personen och väntar sedan på vad du ska säga härnäst.




▲ **Bild 5-1:** Python skriver meddelandet ”Hej, världen!” i skalområdet



## SYNTAXFEL

Om programmet inte körs, utan istället skriver meddelandet ”syntax error” till skalområdet, finns det ett fel någonstans i det du har skrivit. Instruktionerna i Python måste skrivas på ett mycket specifikt sätt: om du missar en parentes eller ett citattecken, stavar ”print” fel eller med ett stort P eller lägger till extra symboler någonstans i instruktionen så går den inte att köra. Försök att skriva instruktionen igen och se till att den matchar versionen i den här boken innan du trycker på **ENTER**-tangenten!

Du behöver dock inte använda Python i interaktivt läge. Klicka på skriptområdet mitt i Thonny-fönstret och skriv sedan programmet igen:

```
print("Hej, världen!")
```

När du trycker på tangenten **ENTER** den här gången händer ingenting, förutom att du får en ny, tom rad i skriptområdet. För att få den här versionen av programmet att fungera måste du klicka på ikonen Run i Thonnys verktygsfält. När du gör det blir du ombedd att först spara programmet. Skriv in ett beskrivande namn, som ”Hej världen” och klicka på knappen Save. När programmet har sparats visas två meddelanden i Pythons skalområde (**Bild 5-2**):

```
>>> % Run 'Hej världen.py'
    Hej, världen!
```

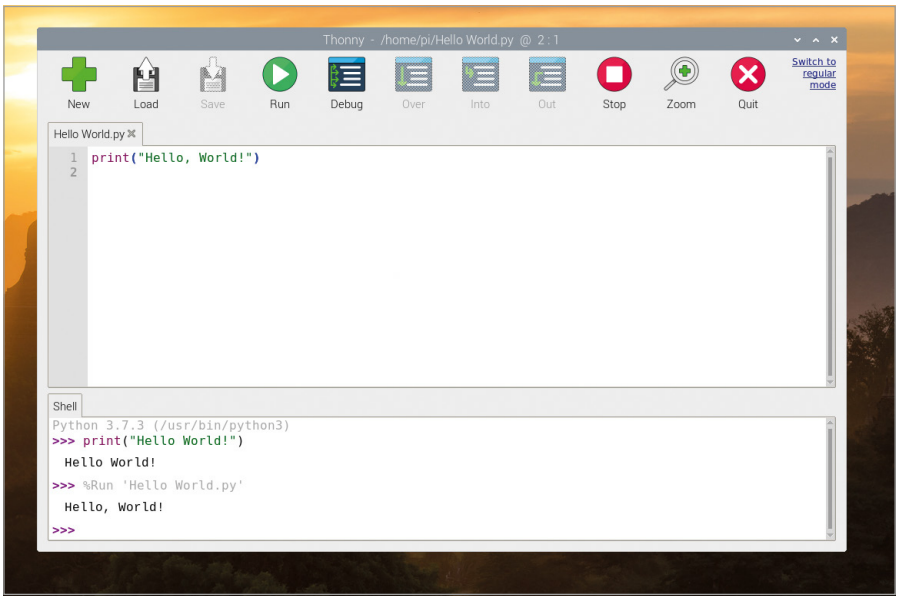



**Bild 5-2:** Köra det enkla programmet

Den första av dessa rader är en instruktion från Thonny som säger åt Python-tolken att köra programmet du just sparat. Den andra är utdata från programmet, meddelandet du sa åt Python att skriva. Grattis: du har nu skrivit och kört ditt första Python-program i både interaktivt läge och skriptläge!

**UTMANING:** NYTT MEDDELANDE

Kan du ändra meddelandet som Python-programmet skriver som utdata? Om du vill lägga till fler meddelanden, ska du då använda interaktivt läge eller skriptläge? Vad händer om du tar bort parenteserna eller citattecknen från programmet och sedan försöker köra det igen?

## Nästa steg: loopar och kodindrag

Precis som Scratch använder staplar av pusselliknande block för att styra vilka bitar i programmet som sammankopplas med andra bitar, har Python sitt eget sätt att styra sekvensen i vilken programmen körs: *indrag*. Skapa ett nytt program genom att klicka på ikonen New i Thonnys verktygsfält. Du kommer inte att förlora det befintliga programmet. Istället skapar Thonny en ny flik ovanför skriptområdet. Börja med att skriva in följande:

```
print("Loopen börjar!")
for i in range(10):
```

Den första raden skriver ut ett enkelt meddelande till skalet, precis som Hej, världen-programmet. Den andra påbörjar en *finit* loop, som fungerar på samma sätt som i Scratch: en räknare, **`i`**, tilldelas loopen och ges en serie siffror – instruktionen för **`range`**, som ska börja vid siffran 0 och arbeta uppåt mot, men aldrig nå, nummer 10 – att räkna. Kolonsymbolen (**`:`**) säger åt Python att nästa instruktion ska vara en del av loopen.

I Scratch finns instruktionerna som ska ingå i loopen bokstavligen inuti det C-formade blocket. Python använder en annan metod: indragning av kod. Nästa rad börjar med fyra blanksteg, som Thonny bör ha lagt till när du tryckte på **ENTER** efter rad 2:

```
    print("Loopens nummer", i)
```

Blankstegen skjuter denna linje inåt jämfört med de andra linjerna. Med indragningen talar Python om skillnaden mellan instruktioner utanför loopen och instruktioner inuti loopen. Den indragna koden kallas även för *kapslad*.

Du kommer att märka att när du trycker på **ENTER** i slutet av den tredje raden har Thonny automatiskt dragit in nästa rad, eftersom Thonny har förutsatt att den ska vara en del av loopen. För att ta bort detta, tryck bara på tangenten **BACKSTEG** en gång innan du skriver den fjärde raden:

```
print("Loopen är klar!")
```

Det fyrradiga programmet är nu klart. Den första raden sitter utanför loopen och körs bara en gång. Den andra raden ställer in loopen, den tredje sitter inne i loopen och kommer att köras en gång för varje gång loopen loopas, och den fjärde raden sitter återigen utanför loopen.

```
print("Loopen börjar!")
for i in range(10):
    print("Loopens nummer", i)
print("Loopen är klar!")
```

Klicka på ikonen Run, spara programmet som **Indrag** och visa skalområdet för dess utdata (**Bild 5-3**):

```
Loopen börjar!
Loopens nummer 0
Loopens nummer 1
Loopens nummer 2
Loopens nummer 3
Loopens nummer 4
Loopens nummer  5
Loopens nummer 6
Loopens nummer 7
Loopens nummer 8
Loopens nummer 9
Loopen är klar!
```

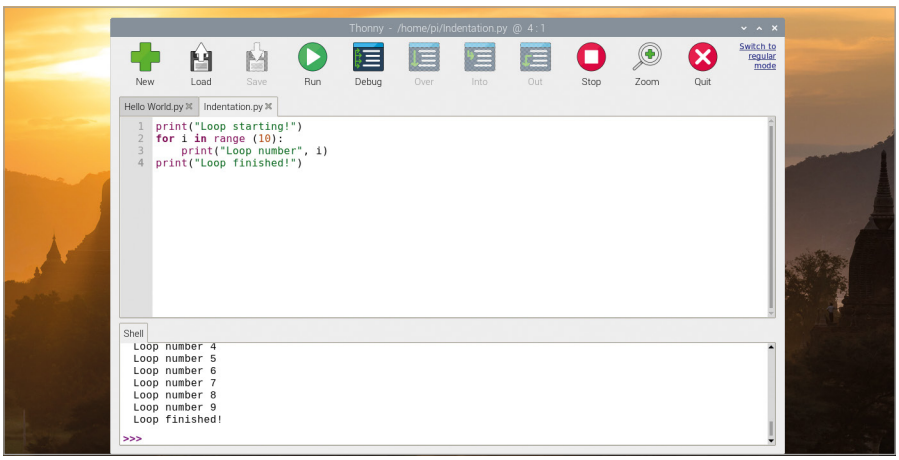
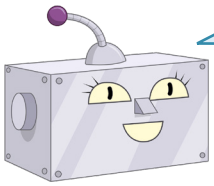



**Bild 5-3:** Exekvera en loop



### RÄKNA FRÅN NOLL

Python är ett nollindexerat språk, vilket betyder att det börjar räkna från 0, inte från 1. Därför kommer programmet att skriva ut siffrorna 0 till 9 istället för 1 till 10. Om du vill kan du ändra detta beteende genom att växla instruktionen för `range(10)` till `range(1, 11)` eller andra siffror du gillar.

Indrag är en kraftfull del av Python och en av de vanligaste orsakerna till att ett program inte fungerar som du förväntat dig. När du letar efter problem i ett program, en process som kallas *felsökning*, ska du alltid dubbelkolla indragningen – särskilt när du börjar kapsla in loopar i loopar.

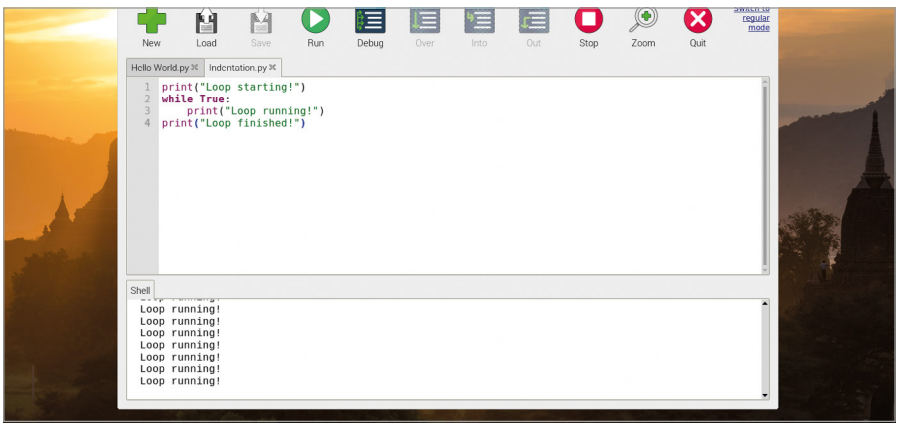
Python har även stöd för *oändliga* loopar, som körs utan slut. Om du vill ändra programmet från en finit loop till en oändlig loop ska du redigera rad 2 så att det står:

```
while True:
```

Om du klickar på ikonen Run nu får du ett fel: **`name 'i' is not defined`**. Det beror på att du har tagit bort raden som skapade och tilldelade ett värde till variabeln **`i`**. För att åtgärda detta redigerar du bara rad 3 så att den inte längre använder variabeln:

```
    print("Loopen körs!")
```

Klicka på ikonen Run, och om du är snabb kan du se meddelandet ”Loopen börjar!” följt av en oändlig loop med ”Loopen körs”-meddelanden (**Bild 5-4**). Meddelandet ”Loopen är klar!” kommer aldrig att skrivas eftersom loopen inte har något slut: varje gång Python har skrivit meddelandet ”Loopen körs!” går det tillbaka till loopens början och skriver det igen.




▲ **Bild 5-4:** En oändlig loop fortsätter tills du stoppar programmet

Klicka på Stop-ikonen i Thonnys verktygsfält för att be programmet att stoppa det som händer. Det kallas att avbryta programmet. Du ser ett meddelande visas i Pythons skalområde, och programmet kommer att stanna utan att nå rad 4.

### UTMANING: LOOPA LOOPEN

Kan du ändra tillbaka loopen till en finit loop igen? Kan du lägga till en andra finit loop i programmet? Hur skulle du lägga till en loop i en loop, och hur tror du att det blir?

## Villkor och variabler

Variabler, som i alla programmeringsspråk, gör mer än att bara kontrollera loopar. Starta ett nytt program genom att klicka på ikonen New i Thonny-menyn och skriv sedan följande i skriptområdet:

```
userName = input ("Vad heter du? ")
```

Klicka på ikonen Run, spara programmet som **Namntest** och se vad som händer i skalområdet: du blir ombedd att ange ditt namn. Skriv ditt namn i skalområdet, följt av **ENTER**. Eftersom det är den enda instruktionen i programmet kommer inget annat att hända (**Bild 5-5**). Om du faktiskt vill göra något med de data du har placerat i variabeln behöver du fler rader i programmet.

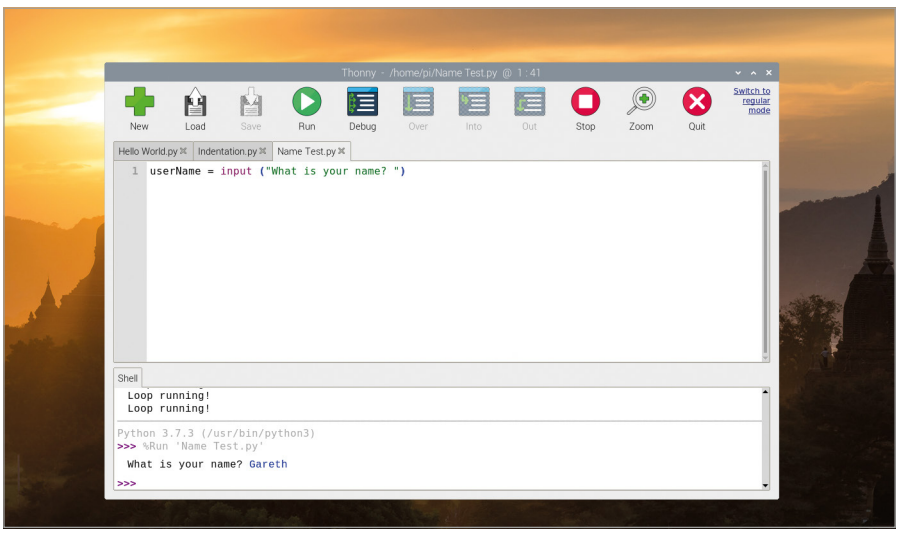


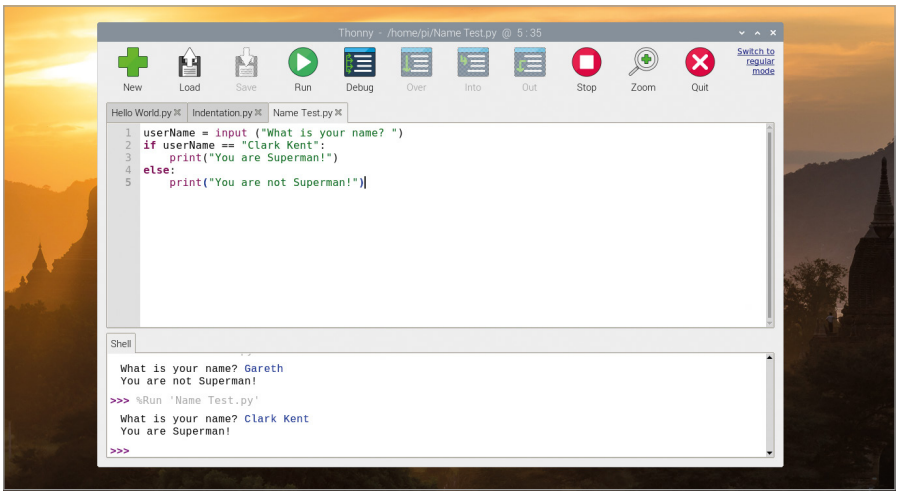

**Bild 5-5:** Med funktionen `input` kan du be en användare om textinmatning

För att programmet ska göra något användbart med namnet, ska du lägga till ett *villkorligt påstående* genom att skriva följande:

```
if userName == "Clark Kent":
    print("Du är Superman!")
else:
    print("Du är inte Superman!")
```

Kom ihåg att när Thonny ser att din kod måste dras in kommer den att göra det automatiskt, men programmet vet inte när koden behöver sluta vara indragen, så du måste ta bort blankstegen själv.

Klicka på ikonen Run och ange ditt namn i skalområdet. Om inte ditt namn råkar vara Clark Kent ser du meddelandet ”Du är inte Superman!”. Klicka på Run igen och skriv den här gången namnet ”Clark Kent”. Se till att skriva det exakt som i programmet, med C och K som stora bokstäver. Den här gången känner programmet igen att du faktiskt är Superman (**Bild 5-6**).




▲ **Bild 5-6:** Borde inte du vara ute och rädda världen?

Symbolerna **==** ber Python att göra en direkt jämförelse och se om variabeln **userName** matchar texten, som kallas en *sträng*, i programmet. Om du arbetar med siffror finns det andra jämförelser du kan göra: **>** för att se om ett tal är större än ett annat tal, **<** för att se om det är mindre än, **=>** för att se om det är lika med eller större än, **=<** för att se om det är lika med eller mindre än. Det finns även **!=**, som betyder inte lika med, det är den exakta motsatsen till **==**. Dessa symboler är tekniskt sett kända som *jämförelseoperatorer*.

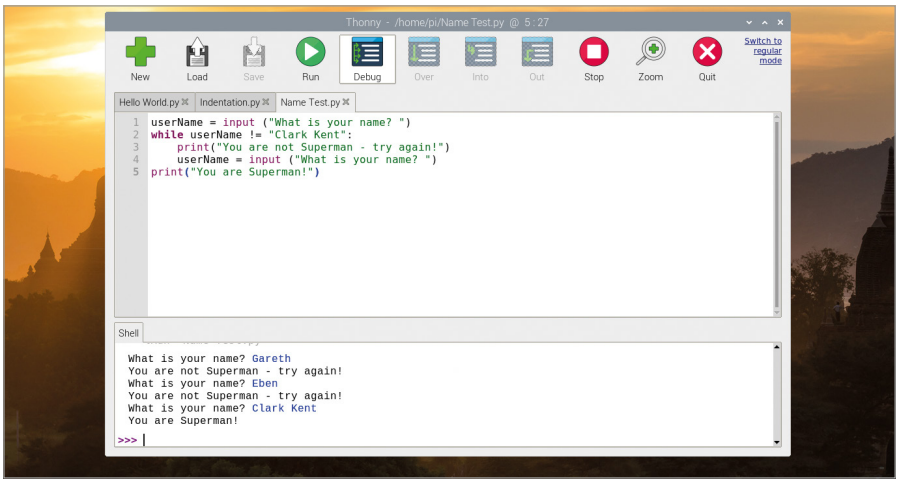
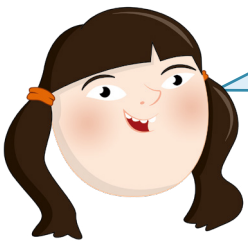



**Bild 5-7:** Den kommer att fortsätta fråga efter ditt namn tills du säger att det är ”Clark Kent”



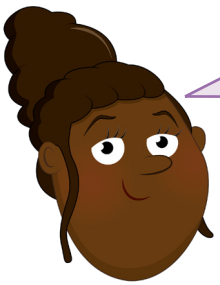
### ANVÄNDA = OCH ==

Nyckeln till att använda variabler är att lära sig skillnaden mellan = och ==. Kom ihåg: = betyder ”gör denna variabel lika med detta värde”, medan == betyder ”kontrollera om variabeln är lika med detta värde”. Om du blandar ihop dem får du garanterat ett program som inte fungerar!

Jämförelseoperatorer kan även användas i loopar. Radera raderna 2 till 5 och skriv sedan följande på deras plats:

```
while userName != "Clark Kent":
    print("Du är inte Superman – försök igen!")
    userName = input ("Vad heter du? ")
print("Du är Superman!")
```

Klicka på ikonen Run igen. Den här gången, istället för att sluta, kommer programmet att fortsätta fråga efter ditt namn tills det bekräftar att du är Superman (**Bild 5-7**) – ungefär som ett mycket enkelt lösenord. För att komma ur loopen skriver du antingen ”Clark Kent” eller klickar på ikonen Stop i Thonnys verktygsfält. Grattis: nu vet du hur man använder villkor och jämförelseoperatorer!



### UTMANING: LÄGG TILL FLER FRÅGOR

Kan du ändra programmet så att det ställer mer än en fråga och lagrar svaren i flera variabler? Kan du skapa ett program som använder villkor och jämförelseoperatorer för att skriva ut om en siffra som skrivs in av användaren är högre eller lägre än 5, som programmet du skapade i **kapitel 4, Programmering med Scratch**?

## Projekt 1: Sköldpaddssnöflingor

Nu när du förstår hur Python fungerar är det dags att leka med grafik och skapa en snöflinga med ett verktyg som kallas en *turtle* (sköldpadda).

### ONLINE-PROJEKT

Detta projekt finns också tillgängligt online på **rpf.io/turtle-snowflakes**

Sköldpaddor var ursprungligen fysiska robotar som var formade som det djur de fick namn efter, och de är utformade för att röra sig i en rak linje, vända och för att lyfta och sänka en penna. I den digitala versionen ska sköldpaddan helt enkelt börja eller sluta rita en linje när den rör sig. Till skillnad från vissa andra språk, nämligen Logo och dess många varianter, har Python inget inbyggt sköldpaddsverktyg – men det har ett *bibliotek* med tilläggskod så att det kan få sköldpaddskraft. Bibliotek är kodbuntar som lägger till nya instruktioner för att utöka Pythons funktioner och förs in i dina egna program med ett importkommando.

Skapa ett nytt program genom att klicka på ikonen New och skriv följande:

```
import turtle
```

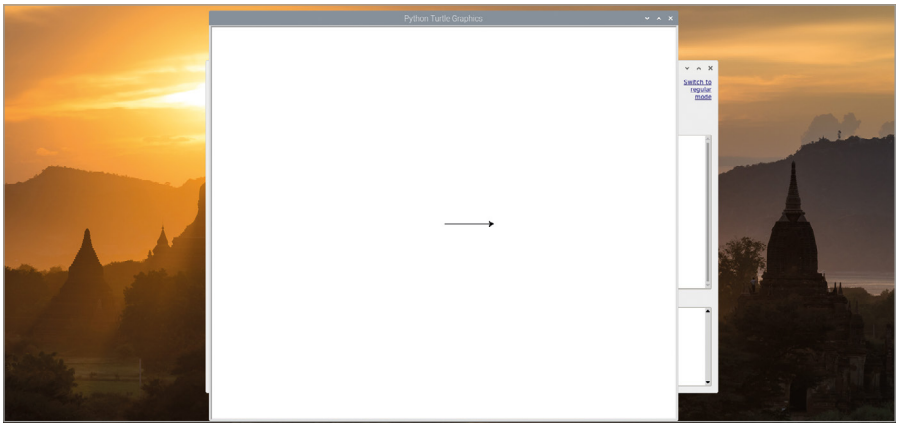
När du använder instruktioner som ingår i ett bibliotek måste du använda biblioteksnamnet följt av punkt och sedan instruktionsnamnet. Det kan vara irriterande att skriva det varje gång, så du kan tilldela ett kortare variabelnamn istället. Det kan vara bara en bokstav, men vi tänkte att det kunde vara trevligt för det att fungera som ett smeknamn för sköldpaddan. Skriv följande:

```
pat = turtle.Turtle()
```

För att testa programmet måste du ge sköldpaddan något att göra. Skriv:

```
pat.forward(100)
```

Klicka på ikonen Run och spara programmet som **Sköldpaddssnöflingor**. När programmet har sparats visas ett nytt fönster som heter ”Turtle Graphics” och du ser resultatet av programmet: sköldpaddan, Pat, kommer att flytta sig framåt 100 enheter och rita en rak linje (**Bild 5-8**).



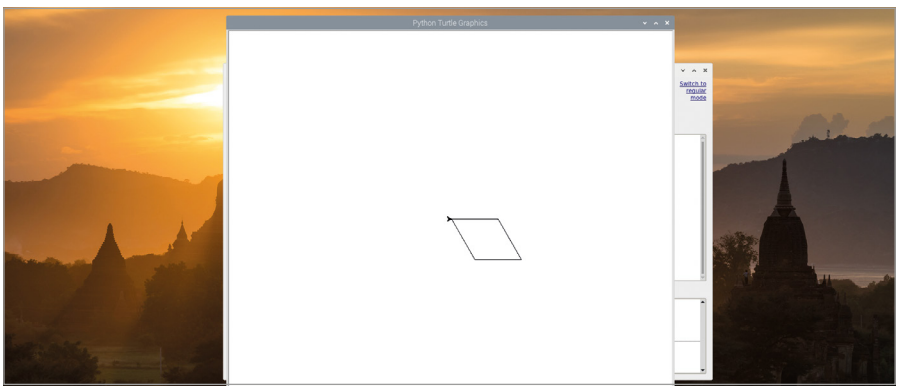

▲ **Bild 5-8:** Sköldpaddan rör sig framåt för att rita en rak linje

Växla tillbaka till Thonnys huvudfönster – om det är dolt bakom Turtle Graphics-fönstret klickar du antingen på minimeringsknappen i Turtle Graphics-fönstret eller på Thonny-posten i aktivitetsfältet högst upp på skärmen – och klicka på Stop-knappen för att stänga Turtle Graphics-fönstret.

Att skriva varenda rörelseinstruktion för hand skulle vara tråkigt, så ta bort rad 3 och skapa en loop som ska utföra det hårda arbetet att skapa former:

```
for i in range(2):
    pat.forward(100)
    pat.right(60)
    pat.forward(100)
    pat.right(120)
```

När du kör programmet ritar Pat ett enda parallellogram (**Bild 5-9**).




▲ **Bild 5-9:** Genom att kombinera svängar och rörelser kan du rita former

För att göra det till en snöflingliknande form ska du klicka på Stop-ikonen i Thonnys huvudfönster och skapa en loop runt din loop genom att lägga till följande rad som rad 3:

```
for i in range(10):
```

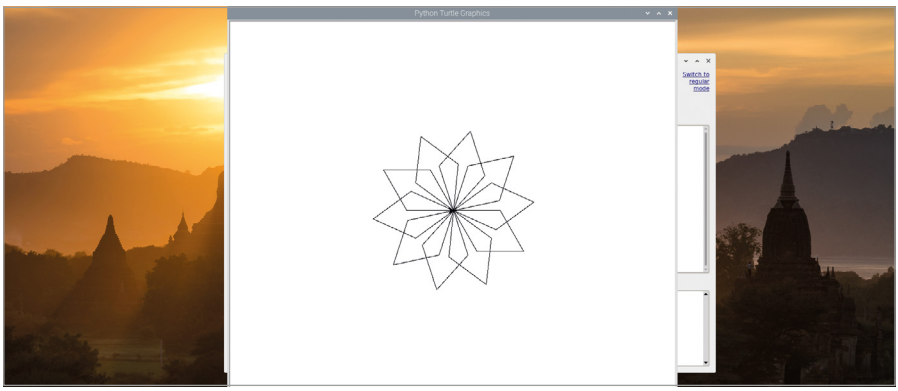
...och följande rad längst ned i programmet:

```
    pat.right(36)
```

Programmet körs inte som det är, eftersom den befintliga loopen inte är indragen korrekt. För att åtgärda det ska du klicka på början på varje rad i den befintliga loopen – rad 4 till 8 – och trycka på tangenten **MELLANSLAG** fyra gånger för att korrigera indragningen. Programmet ska nu se ut så här:

```
import turtle
pat = turtle.Turtle()
for i in range(10):
    for i in range(2):
        pat.forward(100)
        pat.right(60)
        pat.forward(100)
        pat.right(120)
    pat.right(36)
```

Klicka på ikonen Run och titta på sköldpaddan: den ritar ett parallellogram, som tidigare, men när det är klart kommer den att svänga 36 grader och rita ytterligare ett, sedan ytterligare ett och så vidare tills det finns tio överlappande parallellogram på skärmen – och det ser lite ut som en snöflinga (**Bild 5-10**).




▲ **Bild 5-10:** Upprepa formen så att den blir mer komplex

En robotsköldpadda ritar i en enda färg på ett stort papper, men Pythons simulerade sköldpadda kan använda ett antal färger. Lägg till en ny rad 3 och 4 genom att trycka de befintliga linjerna nedåt:

```
turtle.Screen().bgcolor("blue")
pat.color("cyan")
```

Kör programmet igen så ser du effekten av den nya koden: bakgrundsfärgen i Turtle Graphics-fönstret har ändrats till blå och snöflingan är nu cyanfärgad (**Bild 5-11**).

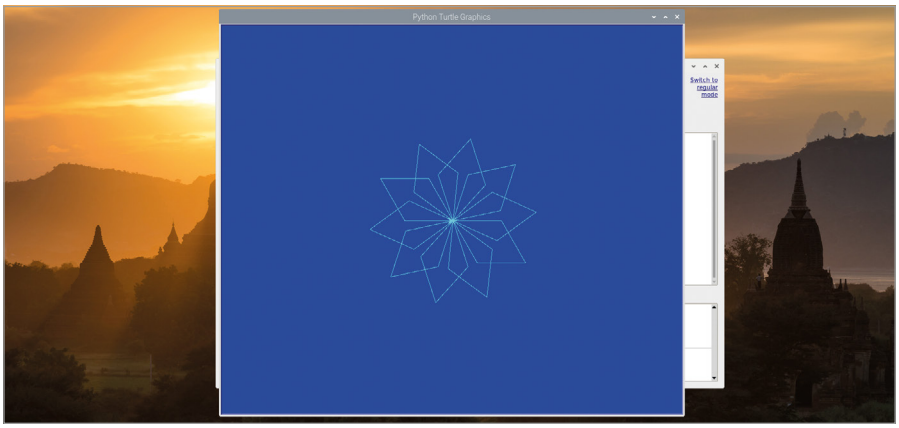



**Bild 5-11:** Ändra färg på bakgrund och snöflinga

Du kan också välja färgerna slumpmässigt från en lista med biblioteket `random`. Gå tillbaka till övre delen av programmet och infoga följande som rad 2:

```
import random
```

Ändra bakgrundsfärgen i det som nu är rad 4 från ”blå” till ”grå” och skapa sedan en ny variabel som heter ”färger” genom att infoga en ny rad 5:

```
colours = ["cyan", "purple", "white", "blue"]
```

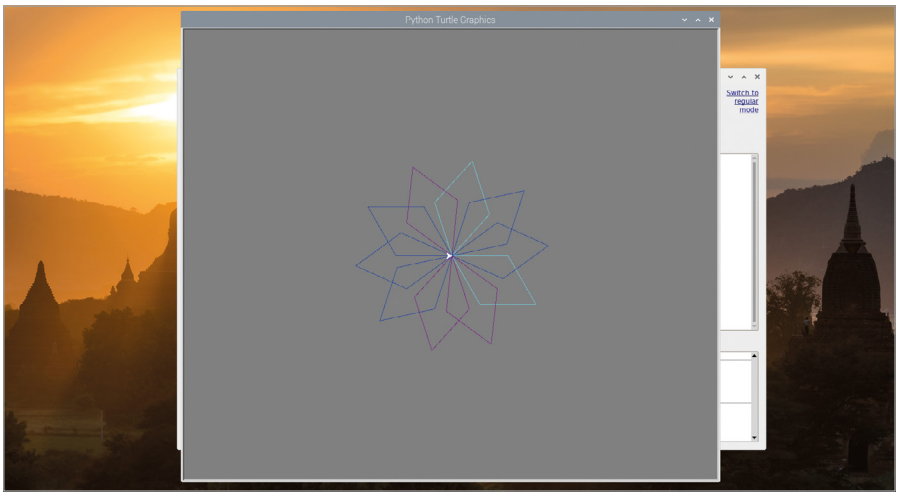
### AMERIKANSK STAVNING

Många programmeringsspråk använder amerikansk stavning, och Python är inget undantag: kommandot för att ändra färgen på sköldpaddans penna stavas *color*, och om du stavar det på brittisk engelska, *colour*, fungerar det helt enkelt inte. Variabler kan dock ha vilken stavning som helst. Det är därför du kan kalla din nya variabel *colours* och ändå få Python att förstå.

Denna typ av variabel brukar kallas en lista och markeras med hakparenteser. I det här fallet är listan fylld med möjliga färger för snöflingsegmenten, men du måste fortfarande be Python att välja en varje gång loopen upprepas. I slutet av programmet anger du följande och ser till att det är indraget med fyra blanksteg så att det utgör en del av den yttre loopen, precis som linjen ovanför den:

```
    pat.color(random.choice(colours))
```

Klicka på ikonen Run så kommer snöflingan/ninjakaststjärnan att ritas igen. Den här gången väljer Python dock en slumpmässig färg från listan när varje kronblad ritas. Det ger snöflingan en tilltalande, mångfärgad finish (**Bild 5-12**).




▲ **Bild 5-12:** Använda slumpmässiga färger till ”kronbladen”

För att få snöflingan att se mindre ut som en ninjakaststjärna och mer som en riktig snöflinga ska du lägga till en ny rad 6, direkt under listan över **`colours`**, och skriva in följande:

```
pat.penup()
pat.forward(90)
pat.left(45)
pat.pendown()
```

Instruktionerna **`penup`** och **`pendown`** skulle flytta en fysisk penna till och från papperet om en sköldpaddrobot användes, men i den virtuella världen ber du helt enkelt sköldpaddan att sluta och börja rita linjer. Den här gången kommer du istället för att använda en loop skapa en *funktion*: ett kodsegment som du kan kalla på när som helst, som att skapa en helt egen Python-instruktion.

Börja med att radera koden för att rita de parallellogrambaserade snöflingorna: det är allt mellan och inklusive instruktionen **`pat.color("cyan")`** på rad 10 till och med **`pat.right(36)`** på rad 17. Lämna kvar instruktionen **`pat.color(random.choice(colours))`**, men lägg till en hash-symbol (**#**) i början av raden. Detta kallas för att *kommentera ut* en instruktion, och det innebär att Python kommer att ignorera den. Du kan använda kommentarer för att lägga till förklaringar till koden, vilket gör det mycket lättare att förstå när du kommer tillbaka till den några månader senare eller när du skickar den till någon annan!

Skapa funktionen, vilken kommer att kallas ”gren”, genom att skriva följande instruktion på rad 10, nedanför **`pat.pendown():`**

```
def branch():
```

Detta *definierar* funktionen, **`grenen`**. När du trycker på **ENTER** lägger Thonny automatiskt till indrag för funktionens instruktioner. Skriv in följande och se till att du är noga med indragningarna, för vid något tillfälle kommer du att bygga in kod som är tre indragningsnivåer djup!

```
    for i in range(3):
        for i in range(3):
            pat.forward(30)
            pat.backward(30)
            pat.right(45)
        pat.left(90)
        pat.backward(30)
        pat.left(45)
    pat.right(90)
    pat.forward(90)
```

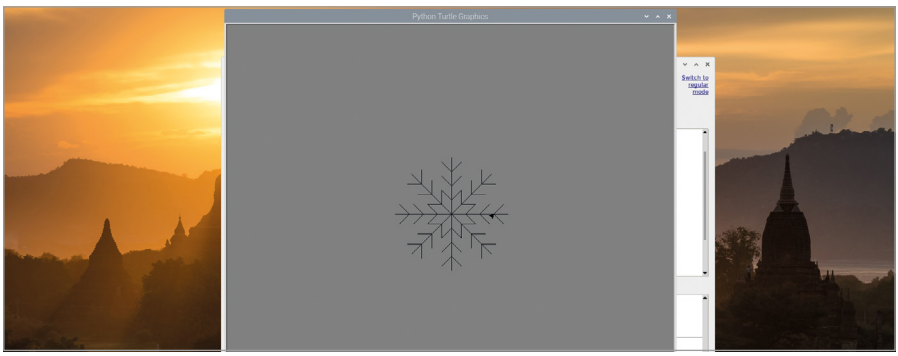
Slutligen skapar du en ny loop längst ner i programmet, men ovanför den utkommenterade färglinjen, för att köra, eller *anropa*, den nya funktionen:

```
for i in range(8):
    branch()
    pat.left(45)
```

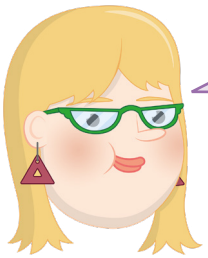
Det färdiga programmet ska nu se ut så här:

```
import turtle
import random
pat = turtle.Turtle()
turtle.Screen().bgcolor("grey")
colours = ["cyan", "purple", "white", "blue"]
pat.penup()
pat.forward(90)
pat.left(45)
pat.pendown()
def branch():
    for i in range(3):
        for i in range(3):
            pat.forward(30)
            pat.backward(30)
            pat.right(45)
        pat.left(90)
        pat.backward(30)
        pat.left(45)
    pat.right(90)
    pat.forward(90)
for i in range(8):
    branch()
    pat.left(45)
#    pat.color(random.choice(colours))
```

Klicka på Run och titta på grafikfönstret när Pat ritar genom att följa dina instruktioner. Grattis: snöflingan ser nu mycket mer ut som en snöflinga (**Bild 5-13**)!



▲ **Bild 5-13:** Extra grenar gör att det ser ut som en snöflinga



### UTMANING: VAD HÄNDER NU?

Kan du använda den utkommenterade instruktionen för att få snöflingans grenar att ritas i olika färger? Kan du skapa en ”snöfling”efunktion” och använda den för att rita massor av snöflingor på skärmen? Kan du göra så att programmet ändrar storlek och färg på snöflingorna slumpmässigt?

## Projekt 2: Läskigt finn fel

Python kan hantera bilder och ljud såväl som sköldpaddsbaserad grafik, vilket kan användas på ett effektfullt sätt som ett upptåg mot dina kompisar: ett spel där man ska hitta fel med en läskig hemlighet i centrum, perfekt för Halloween!

### ONLINE-PROJEKT

Detta projekt finns också tillgängligt online på **rpf.io/scary-spot**

Det här projektet behöver två bilder – en bild där man ska hitta felen plus en ”läskig” överraskningsbild – och en ljudfil. Klicka på hallonikonen för att läsa in Raspberry Pi OS-menyn, välj kategorin internet och klicka på webbläsaren Chromium. När webbläsaren har lästs in skriver du **rpf.io/spot-pic** i adressfältet, följt av **ENTER**-tangenten. Högerklicka på bilden och klicka på ”Spara bild som...”, välj sedan Home-mappen från listan på vänster sida och klicka sedan på Spara. Klicka tillbaka på Chromiums adressfält och skriv sedan **rpf.io/scary-pic** följt av **ENTER**-tangenten. Som tidigare ska du högerklicka på bilden och klicka på ”Spara bild som...”, välja Home-mappen och sedan klicka på Spara.

För ljudfilen du behöver klickar du bakåt i adressfältet och skriver **rpf.io/scream** följt av **ENTER**-tangenten. Den här filen, ljudet av ett skrik som ska ge spelaren en rejäl överraskning, spelas upp automatiskt, men den måste sparas innan du kan använda den. Högerklicka på den lilla ljudspelaren, klicka på ”Spara som...”, välj Home-mappen och klicka på Spara. Nu kan du stänga Chromium-fönstret.

Klicka på ikonen New i Thonnys verktygsfält för att påbörja ett nytt projekt. Som tidigare ska du att använda ett bibliotek för att utöka Pythons möjligheter: Pygame-biblioteket, vilket precis som namnet antyder skapades med spel i åtanke. Skriv följande:

```
import pygame
```

Du behöver vissa delar av andra bibliotek och även från ett underavsnitt av Pygame-biblioteket. Importera dessa genom att skriva följande:

```
from pygame.locals import *
from time import sleep
from random import randrange
```

Instruktionen **`from`** fungerar på ett annat sätt än instruktionen **`import`**, så du kan importera endast de delar av ett bibliotek som du behöver istället för hela biblioteket. Därefter måste du ställa in Pygame, vilket kallas *initialisering*. Pygame behöver veta bredden och höjden på spelarens bildskärm eller tv, vilket kallas *upplösning*. Skriv följande:

```
pygame.init()
width = pygame.display.Info().current_w
height = pygame.display.Info().current_h
```

Det sista steget i att ställa in Pygame är att skapa dess fönster, vilket Pygame kallar en skärm. Skriv följande:

```
screen = pygame.display.set_mode((width, height))
```

```
pygame.quit()
```

Observera den tomma raden i mitten; det är här programmet ska vara. Just nu ska du dock klicka på ikonen Run, spara programmet som **Finn fel** och titta: Pygame skapar ett fönster och fyller det med en svart bakgrund, som sedan nästan omedelbart försvinner när den når instruktionen att sluta. Bortsett från ett kort meddelande i skalet (**Bild 5-14**) har programmet ännu inte åstadkommit mycket.

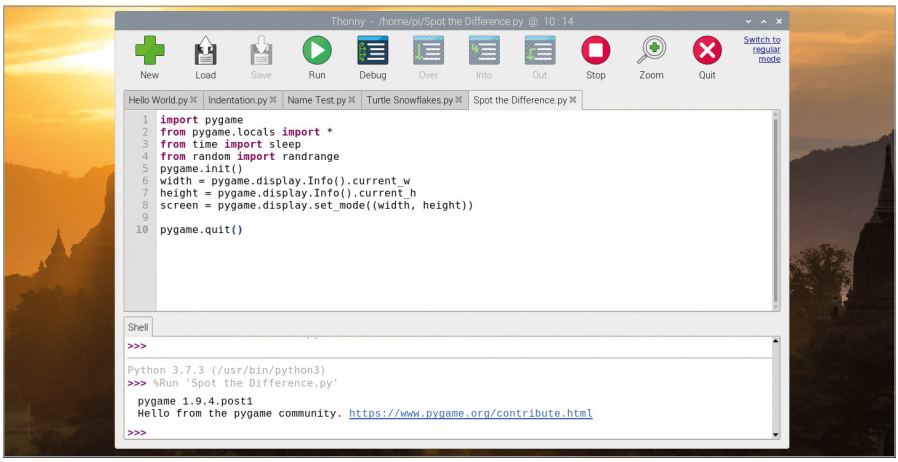



**Bild 5-14:** Programmet är funktionellt, men gör inte mycket ännu

För att visa finn fel-bilden ska du skriva in följande rad i utrymmet ovanför `pygame.quit()`:

```
difference = pygame.image.load('spot_the_diff.png')
```

För att säkerställa att bilden fyller skärmen måste du skala den så att den matchar upplösningen på bildskärmen eller tv:n. Skriv följande:

```
difference = pygame.transform.scale(difference, (width, height))
```

Nu när bilden är i minnet måste du säga åt Pygame att faktiskt visa den på skärmen – en process som kallas för *blitting*, eller en *bitblocköverföring*. Skriv följande:

```
screen.blit(difference, (0, 0))
pygame.display.update()
```

Den första av dessa rader kopierar bilden till skärmen med början i det övre vänstra hörnet. Den andra säger åt Pygame att rita upp skärmen igen. Utan denna andra rad kommer bilden att vara på rätt plats i minnet, men du kommer aldrig att se den!

Klicka på ikonen Run så visas bilden kort på skärmen (**Bild 5-15**).

▲ **Bild 5-15:** Finn fel-bilden

För att se bilden längre lägger du till följande rad precis ovanför `pygame.quit()`:

```
sleep(3)
```

Klicka på Run igen så stannar bilden kvar på skärmen längre. Lägg till överraskningsbilden genom att skriva följande precis under raden `pygame.display.update()`:

```
zombie = pygame.image.load('scary_face.png')
zombie = pygame.transform.scale(zombie, (width, height))
```

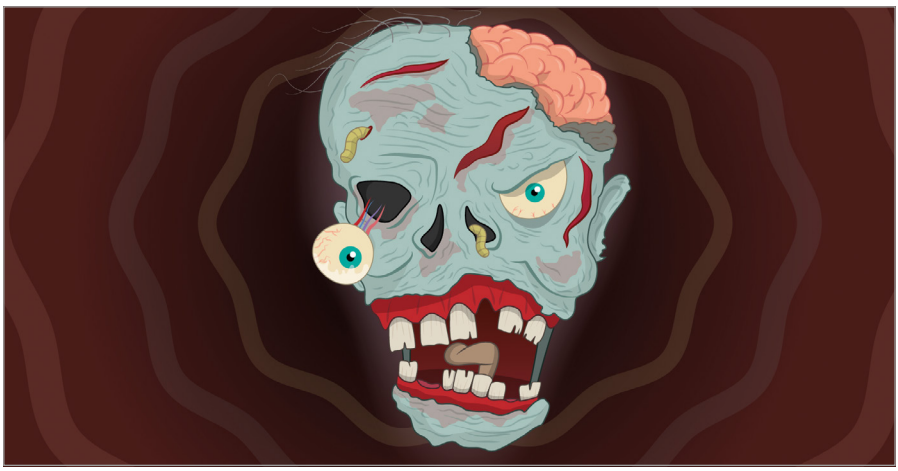
Lägg till en fördröjning så att zombiebilden inte visas direkt:

```
sleep(3)
```

För sedan bilden till skärmen via blitting och uppdatera så att den visas för spelaren:

```
screen.blit(zombie, (0,0))
pygame.display.update()
```

Klicka på ikonen Run och se vad som händer: Pygame läser in finn fel-bilden, men efter tre sekunder kommer den att ersättas med den läskiga zombien (**Bild 5-16**)!



▲ **Bild 5-16:** Den kommer att ge någon en läskig överraskning

Att ha fördröjningen inställd på tre sekunder gör dock saker och ting lite förutsägbara. Ändra raden **`sleep(3)`** ovanför **`screen.blit(zombie, (0,0))`** till:

```
sleep(randrange(5, 15))
```

Detta väljer ett slumpmässigt tal mellan 5 och 15 och fördröjer programmet så länge. Lägg sedan till följande rad precis ovanför instruktionen **`sleep`** för att läsa in skrikljudfilen:

```
scream = pygame.mixer.Sound('scream.wav')
```

Flytta dig nedanför instruktionen **`sleep`** och skriv följande på en ny rad för att starta ljudet,

så att det sätts igång precis innan den läskiga bilden faktiskt visas för spelaren:

```
scream.play()
```

Slutligen ska du säga åt Pygame att sluta spela ljudet genom att skriva följande rad precis ovanför **pygame.quit()**:

```
scream.stop()
```

Klicka på ikonen Run och beundra vad du har gjortditt verk: efter några sekunder av oskyldigt finn fel-kul kommer din läskiga zombie att dyka upp tillsammans med ett blodisande skrik. Det kommer garanterat att skrämma slag på dina kompisar! Om du upptäcker att zombiebilden visas innan ljudet börjar spelas kan du kompensera genom att lägga till en liten fördröjning strax efter instruktionen **scream.play()** och före **screen.blit**:

```
sleep(0.4)
```

Det färdiga programmet ska nu se ut så här:

```
import pygame
from pygame.locals import *
from time import sleep
from random import randrange
pygame.init()
width = pygame.display.Info().current_w
height = pygame.display.Info().current_h
screen = pygame.display.set_mode((width, height))
difference = pygame.image.load('spot_the_diff.png')
difference = pygame.transform.scale(difference, (width, height))
screen.blit(difference, (0, 0))
pygame.display.update()
zombie = pygame.image.load('scary_face.png')
zombie = pygame.transform.scale (zombie, (width, height))
scream = pygame.mixer.Sound('scream.wav')
sleep(randrange(5, 15))
scream.play()
screen.blit(zombie, (0,0))
pygame.display.update()
sleep(3)
scream.stop()
pygame.quit()
```

Det enda som återstår nu är att bjuda in dina kompisar att spela finn fel – och se till att volymen för högtalarna är uppskruvad, naturligtvis!

### UTMANING: ÄNDRA UTSEENDET

Kan du ändra bilderna så att upptåget blir lämpligare för andra evenemang, som julafton? Kan du rita en egen hitta fel-bild och läskiga bilder (med hjälp av en grafikredigerare som GIMP)? Kan du spåra när användaren klickar på ett fel för att göra det mer övertygande?

## Projekt 3: RPG Maze

Nu när börjar du få koll på Python är det dags att använda Pygame för att göra något lite mer komplicerat: ett fullt fungerande textbaseratmaze- labyrintspel baserat på klassiska rollspel. De kallas för textäventyr eller interaktiv fiktion. Denna typ av spel går tillbaka till när datorer inte kunde hantera grafik, men de har anhängare som hävdar att ingen grafik någonsin kommer att vara så levande som de bilder du har i din fantasi!

### ONLINE-PROJEKT

Detta projekt finns också tillgängligt online på **rpf.io/python-rpg**

Det här programmet är bra mycket mer komplext än de andra i det här kapitlet, så för att göra det enklare börjar du med en version som delvis redan är skriven. Öppna webbläsaren Chromium och gå till följande adress: **rpf.io/rpg-code**.

Chromium-webbläsaren laddar automatiskt ner koden för programmet till mappen Downloads, men varnar dig för att filtypen – ett Python-program – kan skada datorn. Du har laddat ner filen från Raspberry Pi Foundation, en betrodd källa, så klicka på Keep-knappen i varningsmeddelandet som visas längst ner på skärmen. Gå tillbaka till Thonny och klicka sedan på ikonen Load . Leta upp filen **rpg-rpg.py** i mappen Downloads och klicka på knappen Load.

Börja med att klicka på ikonen Run för att bekanta dig med hur ett textäventyr fungerar. Spelets utdata visas i skalområdet längst ner i Thonny-fönstret. Gör Thonny-fönstret större genom att klicka på maximeringsknappen för att göra det lättare att läsa.

Spelet, som det ser ut nu, är väldigt enkelt: det finns två rum och inga föremål. Spelaren startar i  HallHall (hallen), det första av de två rummen. För att gå till , Kitchen, (köket) skriver du helt enkelt ”go south” följt av **ENTER**-tangenten (**Bild 5-17**). När du är i köket kan du skriva ”go north” för att återvända till hallen. Du kan också prova att skriva ”go west” och ”go east”, men eftersom det inte finns några rum i dessa riktningar kommer spelet att visa dig ett felmeddelande.

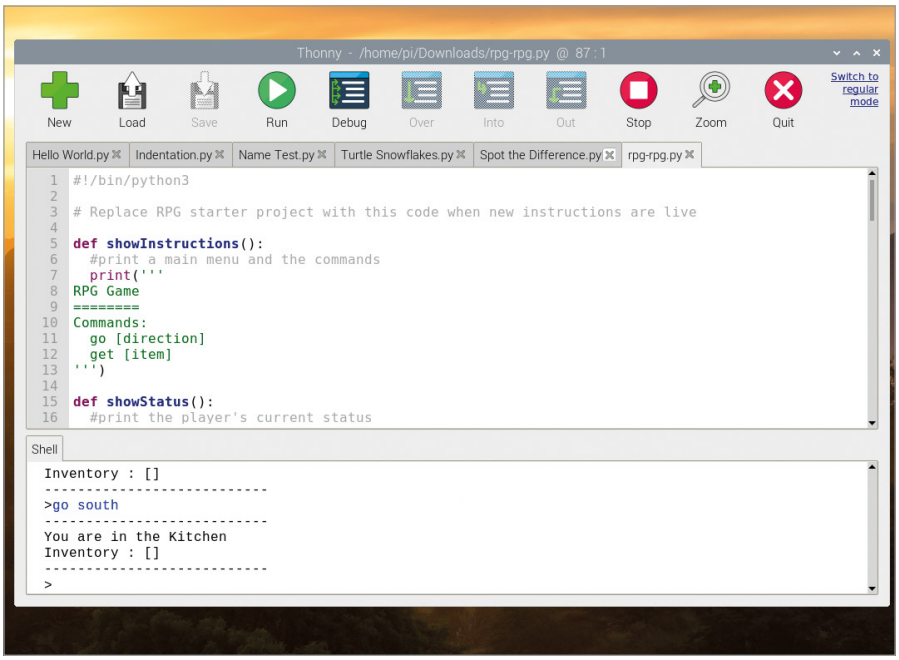



**Bild 5-17:** Det finns bara två rum hittills

Bläddra ner till rad 29 i programmet i skriptområdet för att hitta en variabel som heter **`rooms`**. Denna typ av variabel brukar kallas en *ordlista* och talar om för spelet vilka rum som finns, deras utgångar och vilket rum en given utgång leder till.

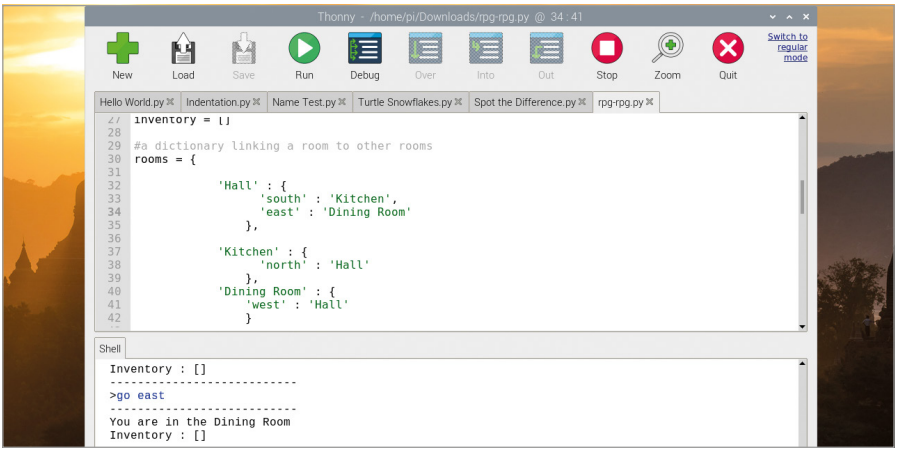
För att göra spelet mer intressant ska du lägga till ytterligare ett rum: Dining Room, (matsal) öster om hallen. Hitta variabeln **`rooms`** i skriptområdet och utöka den genom att lägga till en kommasymbol (**`,`**) efter **`}`** på rad 38 och skriv sedan följande (exakt indragning är inte nödvändigt i en ordlista):

```
'Dining Room' : {
      'west' : 'Hall'
   }
```

Du behöver även en ny utgång i hallen, eftersom den inte skapas automatiskt åt dig. Gå till slutet av rad 33, lägg till ett komma och lägg sedan till följande rad:

```
'east' : 'Dining Room'
```

Klicka på ikonen Run och prova det nya rummet: skriv ”go east” när du är i hallen för att komma in i matsalen (**Bild 5-18**, på nästa sida) och skriv ”go west” när du är i matsalen för att komma in i hallen. Grattis: du har skapat ett eget rum!

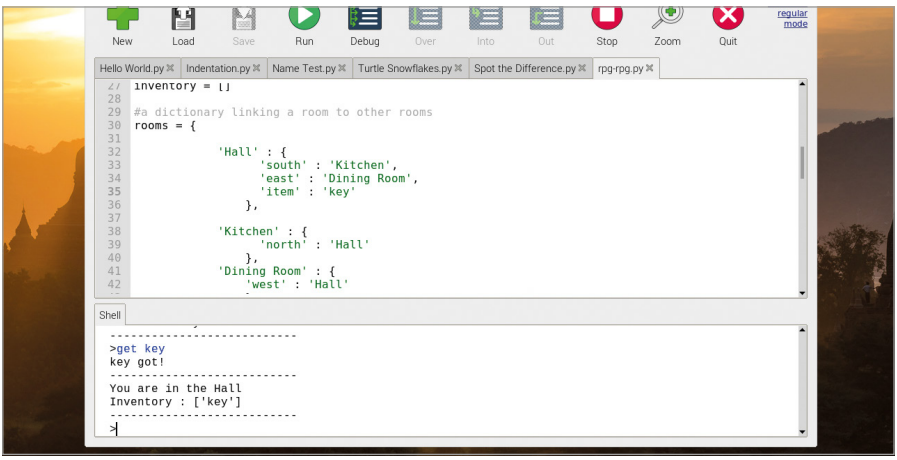



▲ **Bild 5-18:** Du har lagt till ytterligare ett rum

Tomma rum är dock inte så roliga. Om du vill lägga till ett objekt i ett rum måste du ändra det rummets ordlista. Stoppa programmet genom att klicka på Stop-ikonen. Hitta ordlistan för **`Hall`** i skriptområdet och lägg sedan till ett kommatecken i slutet av raden **`'east' : 'Dining Room'`** innan du trycker på **ENTER** och skriver följande rad:

```
'item' : 'key'
```

Klicka på Run igen. Den här gången kommer spelet att berätta att du kan se det nya objektet: key (nyckel). Skriv ”get key” (**Bild 5-19**) så att du kan hämta den och lägga till den i listan över objekt du har på dig – det kallas ditt *lager*. Lagret stannar kvar hos dig när du rör dig från rum till rum.




▲ **Bild 5-19:** Den hämtade nyckeln läggs till i lagret

Klicka på Stop-ikonen och gör spelet mer intressant genom att lägga till ett monster som man ska undvika. Leta upp ordlistan för **`Kitchen`** och lägg till objektet ”monster” på samma sätt som du lade till objektet ”key”, och kom ihåg att lägga till ett komma i slutet av raden ovan:

```
'item' : 'monster'
```

För att monstret ska kunna attackera spelaren måste du lägga till logik i spelet. Rulla allra längst ner i programmet i skriptområdet och lägg till följande rader, inklusive kommentaren som är markerad med en hash-symbol, som hjälper dig att förstå programmet om du kommer tillbaka till det en annan dag – och se till att ha indrag på raderna:

```
    # player loses if they enter a room with a monster
    if 'item' in rooms[currentRoom] and 'monster' in
rooms[currentRoom]['item']:
        print('A monster has got you ... SPELET ÄR SLUT!')
        break
```

Klicka på Run och försök gå in i köket (**Bild 5-20**) – monstret blir inte särskilt imponerat när du gör det!

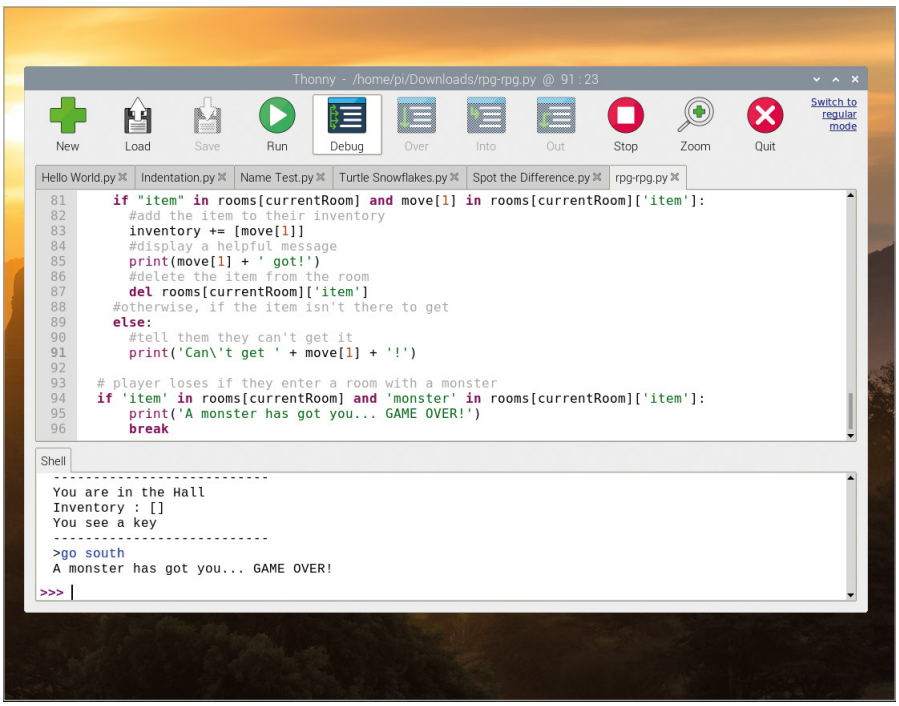



**Bild 5-20:** Vem bryr sig om råttor när det finns ett monster i köket?

För att göra detta äventyr till ett ordentligt spel behöver du fler föremål, ett till rum och möjligheten att ”vinna” genom att lämna huset med alla föremål i säkert förvar i lagret. Börja med att lägga till ytterligare ett rum, precis som för matsalen – men den här gången är det , Garden (trädgård). Lägg till en utgång från ordlistan för Dining Room, och kom ihåg att lägga till ett komma i slutet av linjen ovanför:

```
'south' : 'Garden'
```

Lägg sedan till det nya rummet i huvudordlistan **`rooms`**, och kom ihåg att lägga till ett komma efter **`}`** på raden ovanför som tidigare:

```
'Garden' : {
      'north' : 'Dining Room'
    }
```

Lägg till ett trolldrycks”potion”-objekt, ”potion”, (trolldryck) i ordlistan Dining Room och kom ihåg att lägga till det nödvändiga kommat i raden ovan:

```
'item' : 'potion'
```

Slutligen ska du rulla till nederdelen av programmet och lägga till den logik som krävs för att kontrollera om spelaren har alla objekt och, i så fall, berätta för denne att spelet är vunnet:

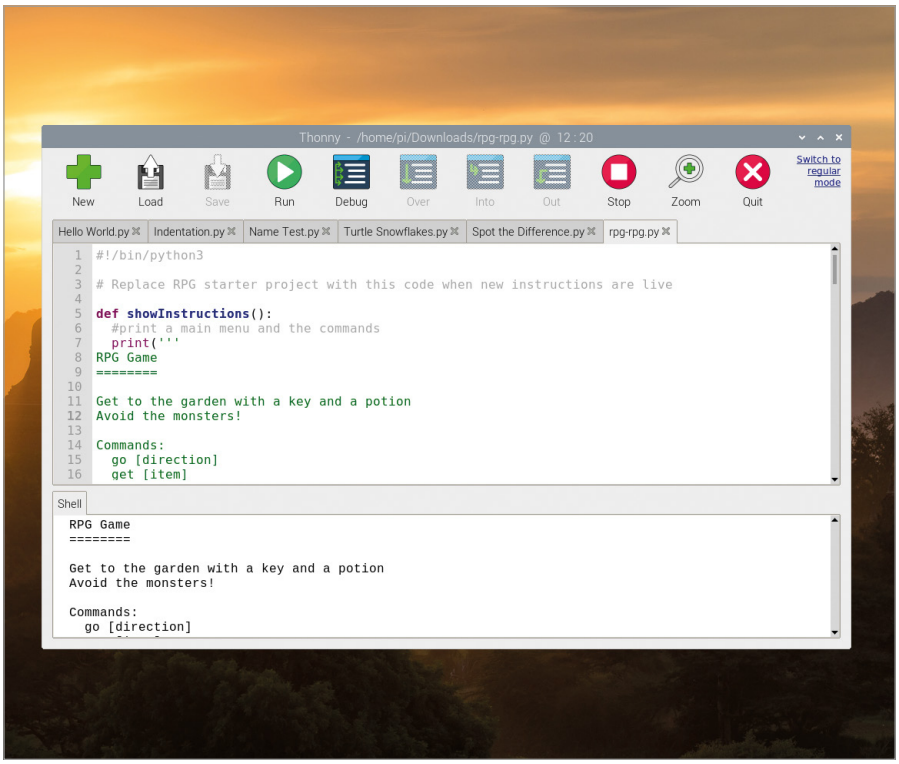
```
  # player wins if they get to the garden with a key and a potion
    if currentRoom == 'Garden' and 'key' in inventory and
'potion' in inventory:
        print('You escaped the house ... DU VANN!')
        break
```

Klicka på Run och försök avsluta spelet genom att plocka upp nyckeln och trolldrycken innan du går till trädgården. Glöm inte att du inte ska gå in i köket, för det är där monstret är!

Som en sista justering av spelet ska du lägga till några instruktioner som berättar för spelaren hur man ska slutföra spelet. Bläddra till överdelen av programmet, där funktionen **`showInstructions()`** är definierad och lägg till följande:

```
Get to the Garden with a key and a potion
Avoid the monsters!
```

Kör spelet en sista gång så ser du de nya instruktionerna visas i början (**Bild 5-21**). Grattis: du har skapat ett interaktivt textbaserat labyrintspel!




▲ **Bild 5-21:** Nu vet spelaren vad som måste göras

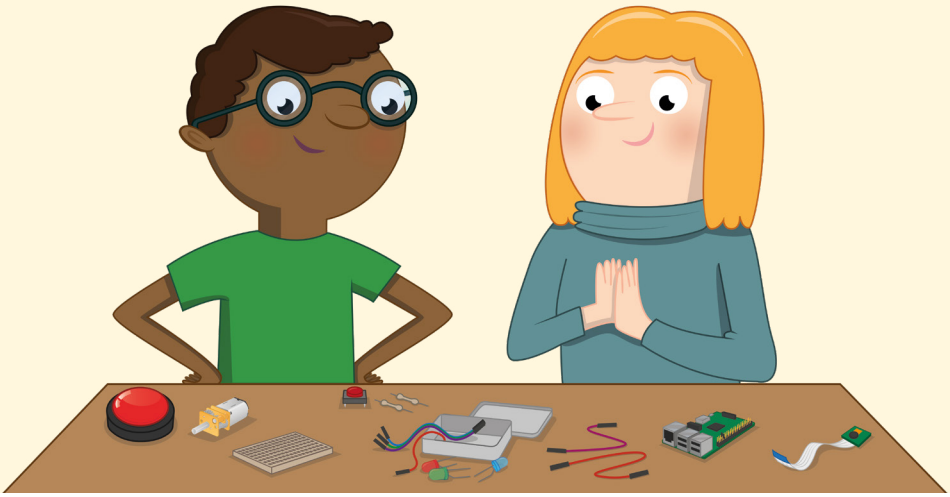
## UTMANING: UTÖKA SPELET

Kan du lägga till fler rum så att spelet varar längre? Kan du lägga till ett föremål som kan skydda dig mot monstret? Hur skulle du lägga till ett vapen som kan döda monstret? Kan du lägga till rum som ligger över och under befintliga rum, som nås via trappor?

Kapitel 6

# Fysisk databehandling med Scratch och Python

**Kodning handlar inte bara om att göra saker på skärmen. Du kan också styra elektroniska komponenter som är anslutna till Raspberry Pi:s GPIO-stift**



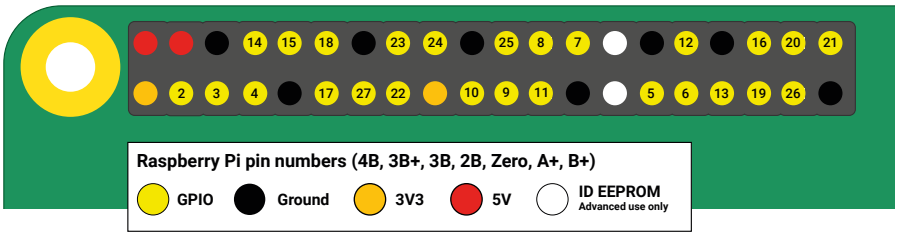
När folk tänker på ”programmering” eller ”kodning” tänker de vanligtvis, helt naturligt, på programvara. Kodning kan dock handla om annat än bara programvara: det kan påverka den verkliga världen genom maskinvara. Detta brukar kallas *fysisk databehandling*.

Som namnet antyder handlar fysisk databehandling om att kontrollera saker i den verkliga världen med programmen: maskinvara istället för programvara. När du ställer in programmet på tvättmaskinen, ändrar temperaturen på en programmerbar termostat eller trycker på en knapp vid trafikljus för att korsa vägen på ett säkert sätt, använder du fysisk databehandling.

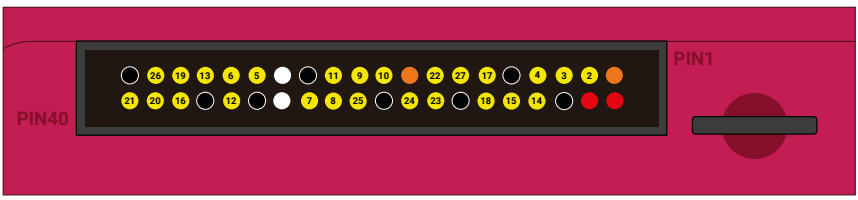
Raspberry Pi är en bra enhet för att lära sig mer om fysisk databehandling tack vare en nyckelfunktion: den *allmänna ingången/utgången* (*GPIO*).

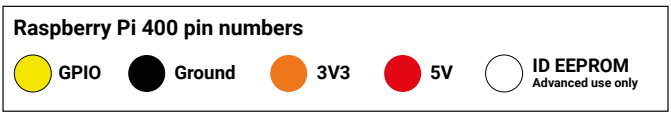
# Introduktion av GPIO

GPIO sitter på den övre kanten av Raspberry Pi:s kretskort, eller på baksidan av Raspberry Pi 400, och ser ut som två långa rader av metallstift. Med GPIO (allmän ingång/utgång) kan du ansluta maskinvara som lysdioder och brytare till Raspberry Pi som du kan styra med programmen du skapar. Stiften kan användas för både ingång och utgång.




Raspberry Pi:s GPIO består av 40 hanstift. Vissa stift är tillgängliga för dig att använda i fysiska datorprojekt, vissa stift ger ström, medan andra stift är reserverade för att kommunicera med tilläggsmaskinvara som Sense HAT (se **Kapitel 7**).




Raspberry Pi 400 har samma GPIO med samma stift, men den är vänd upp och ner jämfört med andra Raspberry Pi-modeller. Detta diagram förutsätter att du tittar på GPIO på baksidan av Raspberry Pi 400. Kontrollera alltid ledningarna när du ansluter något till GPIO på Raspberry Pi 400. Det är lätt att glömma trots etiketterna ”Pin 40” och ”Pin 1” som sitter på höljet!

## GPIO-TILLÄGG

Det är fullt möjligt att använda Raspberry Pi 400:s GPIO som den är, men det kan vara lättare att använda ett tillägg. Med ett tillägg flyttas stiften runt till sidan av Raspberry Pi 400, vilket innebär att du kan kontrollera och justera ledningarna utan att hela tiden behöva kolla baksidan.

Kompatibla tillägg inkluderar Black HAT Hack3r från **pimoroni.com** och Pi T-Cobbler Plus från **adafruit.com**.

Om du köper ett tillägg ska du alltid kontrollera hur det ansluts. En del av dem, som Pi T-Cobbler Plus, ändrar layouten för GPIO-stiften. Använd alltid tillverkarens instruktioner om du är osäker.

Det finns flera kategorier av stifttyper, som alla har en särskild funktion:

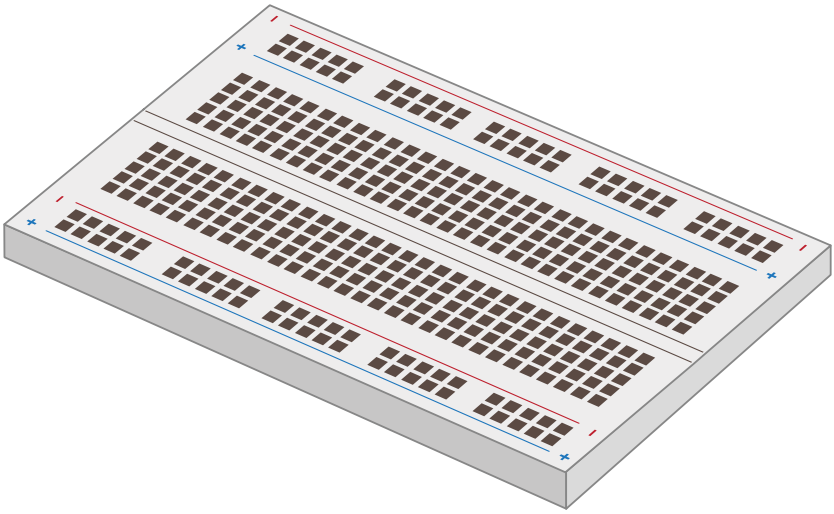
| | | |
|---|---|---|
| 3V3 | **3,3 volts ström** | En spänningskälla på 3,3 V som alltid är på, samma spänning som Rapberry Pi körs med internt |
| 5V | **5 volts ström** | En spänningskälla på 5 V som alltid är på, samma spänning som Rapberry Pi tar in vid micro-USB-strömkontakten |
| Jord (GND) | **0 volt jord** | En jordanslutning som används till att sluta en krets som är ansluten till strömkällan |
| GPIO XX | **Allmänt in-/ utgångsstift nummer ”XX”** | GPIO-stiften som är tillgängliga för programmen identifieras med en siffra från 2 till 27 |
| ID EEPROM | **Reserverade stift med särskilda syften** | Stift som är reserverade för användning med Hardware Attached on Top (HAT) och andra tillbehör |

## VARNING!

Raspberry Pi:s GPIO är ett roligt och säkert sätt att experimentera med fysisk datortillämpning, men det måste behandlas med försiktighet. Var försiktig så att du inte böjer stiften när du ansluter och kopplar bort maskinvara. Anslut aldrig två stift direkt tillsammans, oavsiktligt eller medvetet, såvida du inte uttryckligen uppmanas att göra det i ett projekts instruktioner: detta kallas kortslutning och kan, beroende på stiften, skada Raspberry Pi permanent.

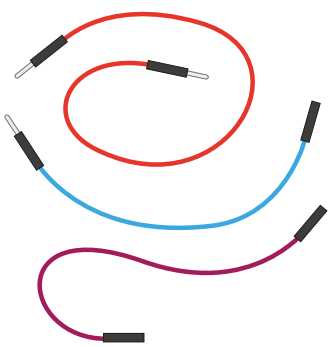
## Elektroniska komponenter

GPIO är bara en del av det du behöver för att börja arbeta med fysisk databehandling. Den andra halvan består av elektriska komponenter, enheterna som du styr från GPIO. Det finns tusentals olika komponenter tillgängliga, men de flesta GPIO-projekt är gjorda med följande vanliga delar.

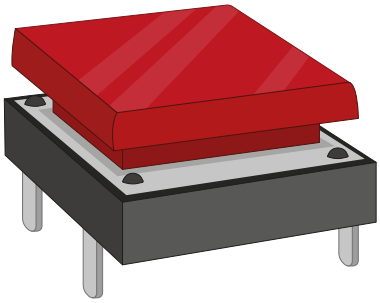


En *kopplingsplatta*, som även brukar kallas en *lödningsfri kopplingsplatta*, kan göra fysiska databehandlingsprojekt betydligt enklare. Istället för att ha en massa separata komponenter som måste anslutas med ledningar kan du sätta i komponenter på en kopplingsplatta och få dem anslutna genom metallspår som är dolda under dess yta. Många kopplingsplattor innehåller även sektioner för strömfördelning, vilket gör det lättare att bygga kretsar. Du behöver inte någon kopplingsplatta för att komma igång med fysisk databehandling, men det underlättar verkligen.

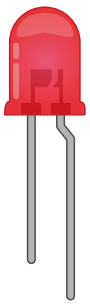
Du kan använda *bygelkablar*, som även kallas *bygelledningar*, för att ansluta komponenter till Raspberry Pi och, om du inte använder en kopplingsplatta, till varandra. De finns i tre versioner: hane-till-hona (M2F), som du behöver för att ansluta en kopplingsplatta till GPIO-stiften; hona-till-hona (F2F), som kan användas för att koppla samman enskilda komponenter om du inte använder en kopplinsplatta; och hane-till-hane (M2M), som används för att skapa anslutningar från en del av en kopplingsplatta till en annan. Beroende på projektet kan du behöva alla tre typer av bygelkablar. Om du använder en kopplingsplatta kan du vanligtvis komma undan med bara M2F- och M2M-bygelkablar.



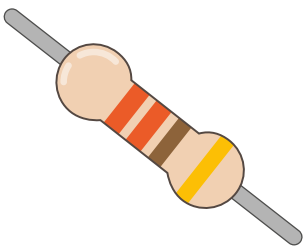
En *tryckknappsbrytare*, som även kallas en *momentan brytare*, är den typ av brytare som du skulle använda för att styra en spelkonsol. Tryckknappen har vanligtvis två eller fyra ben, och båda typerna fungerar med Raspberry Pi. Tryckknappen är en indataenhet: du kan tala om för programmen att det ska vänta på att den trycks in och sedan utföra en uppgift. En annan vanlig omkopplare är en *spärrbrytare*. Medan en tryckknapp bara är aktiv när du håller den nere är en spärrbrytare, som du hittar i en lampkontakt, aktiv ändå tills du stänger av den med ett nytt tryck.



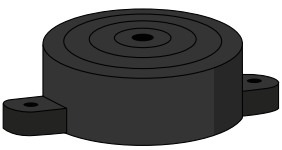
En *lysdiod* (*LED*) är en *utdataenhet* som du styr direkt från programmet. En lysdiod tänds när den är på och du hittar dem överallt i ditt hus, allt från de små som talar om för dig när du har lämnat tvättmaskinen påslagen till de stora du kanske har för att lysa upp rummen. Lysdioder finns i ett brett utbud av former, färger och storlekar, men alla är inte lämpliga för användning med Raspberry Pi. Undvik dem som är utformade för strömförsörjning på 5 V eller 12 V.



*Resistorer* är komponenter som styr flödet av *elektrisk ström* och finns med olika värden som mäts i en enhet som kallas *ohm* (*Ω*). Resistansen blir större ju högre ohm-talet blir. I Raspberry Pi:s fysiska datorprojekt är deras vanligaste användning att skydda lysdioder från att dra för mycket ström och skada sig själva eller Raspberry Pi. Till detta behöver du resistorer med en märkning på cirka 330 Ω. Många elvaruhus säljer praktiska förpackningar som innehåller ett antal olika vanliga värden så att du får större flexibilitet.



En *piezoelektrisk summer*, som vanligtvis kallas bara en summer eller en sounder, är en annan utgående enhet. En LED-lampa producerar ljus medan en summer producerar ett surrande ljud. Inuti summerns plasthölje finns ett par metallplattor. När de är aktiva vibrerar dessa plattor mot varandra för att producera det surrande ljudet. Det finns två summertyper: *aktiv summer* och *passiv summer*. Se till att skaffa en aktiv summer, eftersom de är enklast att använda.



Andra vanliga elektriska komponenter inbegriper motorer som behöver ett speciellt kontrollkort innan de kan anslutas till Raspberry Pi, infraröda sensorer som detekterar rörelse, temperatur- och luftfuktighetssensorer som kan användas för att förutsäga vädret och ljusberoende resistorer (LDR) – indataenheter som fungerar som en omvänd lysdiod genom att detektera ljus.

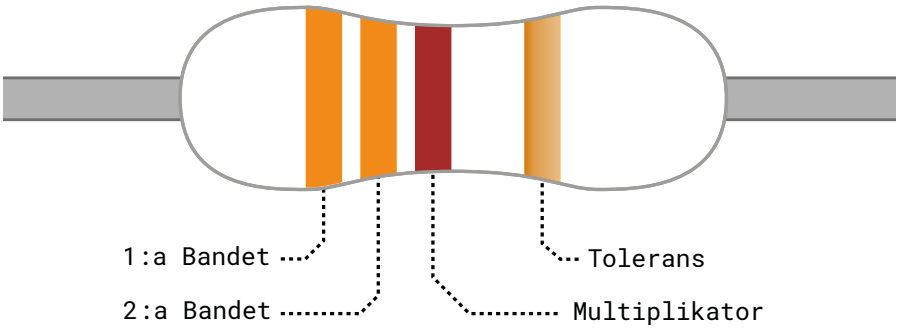
Försäljare över hela världen tillhandahåller komponenter för fysisk databehandling med Raspberry Pi, antingen som enskilda delar eller i kit som innehåller allt du behöver för att komma igång. För att hitta försäljare, gå till **rpf.io/products** och klicka på Raspberry Pi 4 så får du en lista över Raspberry Pi-nätbutiker (godkända återförsäljare) för ditt land eller din region.

För att slutföra projekten i detta kapitel bör du ha minst:

- 3 × lysdioder: röda, gröna och gula eller orangea
- 2 × tryckknappsbrytare
- 1 × aktiv summer
- Bygelkablar hane-till-hona (M2F) och hona-till-hona (F2F)
- Tillval, en kopplingsplatta och bygelkablar hane-till-hane (M2M)

## Läsning av resistorernas färgkoder

Resistorer finns i många olika värden, från nollresistansversioner som i själva verket bara är trådbitar till högresistansversioner som är lika tjocka som ditt ben. Mycket få av dessa resistorer har värdena tryckta i siffror på sig. Istället används en särskild kod med färgade ränder eller band runt resistorns kropp.




| | 1:a/2:a Bandet | Multiplikator | Tolerans |
|---|---|---|---|
| Svart | 0 | $\times 10^{0}$ | - |
| Brun | 1 | $\times 10^{1}$ | ±1% |
| Röd | 2 | $\times 10^{2}$ | ±2% |
| Orange | 3 | $\times 10^{3}$ | - |
| Gul | 4 | $\times 10^{4}$ | - |
| Grön | 5 | $\times 10^{5}$ | ±0.5% |
| Blå | 6 | $\times 10^{6}$ | ±0.25% |
| Violett | 7 | $\times 10^{7}$ | ±0.1% |
| Grå | 8 | $\times 10^{8}$ | ±0.05% |
| Vit | 9 | $\times 10^{9}$ | - |
| Guld | - | $\times 10^{-1}$ | ±5% |
| Silver | - | $\times 10^{-2}$ | ±10% |
| Ingen | - | - | ±20% |

För att läsa av värdet på en resistor ska du placera den så att gruppen av band är till vänster och det ensamma bandet är till höger. Börja med det första bandet och leta upp dess färg i kolumnen ”1:a/2:a bandet” i tabellen för att få första och andra siffran. Detta exempel har två orange band, som båda betyder ett värde på ”3” som tillsammans blir ”33”. Om resistorn har fyra grupperade band istället för tre, anteckna då även värdet på det tredje bandet (för resistorer med fem/sex band, se **rpf.io/5-6band**).

Gå vidare till det sista grupperade bandet, det tredje eller fjärde. Leta upp dess färg i kolumnen ”Multiplikator”. Detta berättar vilken siffra du ska multiplicera ditt nuvarande nummer med för

att få resistansens faktiska värde. Detta exempel har ett brunt band, vilket betyder ”×10¹”. Det kan se förvirrande ut, men det är helt enkelt *grundpotensform*: ”×10¹” betyder ”lägg till en nolla i slutet av ditt nummer”. Om det var blått, med ”×10⁶”, skulle det betyda ”lägg till sex nollor i slutet av ditt nummer”.

33, från de orangea banden, plus den tillagda nollan från det bruna bandet ger oss 330, vilket är resistorns värde mätt i ohm. Det sista bandet, till höger, är resistorns *tolerans*. Det är helt enkelt hur nära det nominella värdet det sannolikt kommer att vara. Billigare resistorer kan ha ett silverband, vilket indikerar att den kan vara 10 procent högre eller lägre än märkningen, eller inget sista band alls, vilket anger att den kan vara 20 procent högre eller lägre. De dyraste resistorerna har ett grått band, vilket indikerar att den kommer att ligga inom 0,05 procent av märkningen. För hobbyprojekt är noggrannhet inte så viktigt: vilken tolerans som helst brukar fungera.

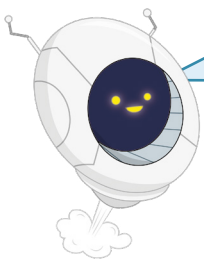
Om resistorns värde överstiger 1 000 ohm (1 000 Ω) klassas det vanligtvis i kiloohm (kΩ); om det överstiger en miljon ohm är de megaohm (MΩ). En resistor på 2 200 Ω skulle skrivas som 2,2 kΩ. En resistor på 2 200 000 Ω skulle skrivas som 2,2 MΩ.

### KAN DU LISTA UT DET?

Vilka färgband skulle en resistor på 100 Ω ha? Vilket färgband skulle en resistor på 2,2 MΩ ha? Om du vill hitta de billigaste resistorerna, vilket färg på toleransbandet ska du leta efter?

## Ditt första fysiska datortillämpningsprogram: Hej, lysdiod!

Precis som att skriva ”Hej, världen” på skärmen är ett fantastiskt första steg i att lära sig ett programmeringsspråk, så är den traditionella introduktionen till att lära sig fysisk datortillämpning att tända en lysdiod. Till detta projekt behöver du en lysdiod och en resistor på 330 ohm (330 Ω), eller så nära 330 Ω som du kan komma, plus bygelkablar hona-till-hona (F2F).

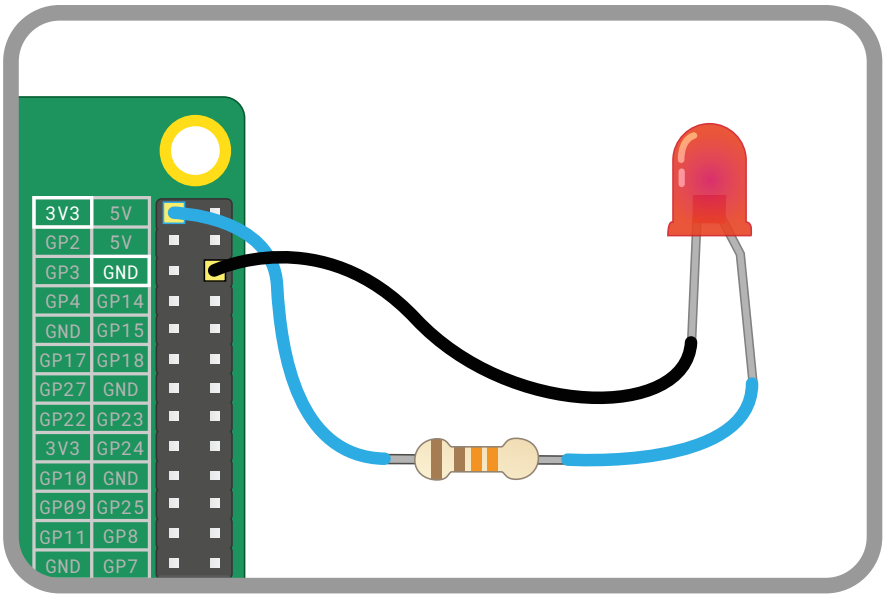


### RESISTANS ÄR AVGÖRANDE

Resistorn är en viktig komponent i kretsen: den skyddar Raspberry Pi och lysdioden genom att begränsa mängden elektrisk ström som lysdioden kan dra till sig. Utan den kan lysdioden dra till sig för mycket ström och bränna ut sig eller Raspberry Pi. När resistorn används så här kallas den för en *strömbegränsande resistor*. Det exakta resistorvärde som du behöver beror på vilken lysdiod du använder, men 330 Ω fungerar för de vanligaste lysdioderna. Ju högre värde, desto dunklare lyser lysdioden; ju lägre värde, desto ljusare blir lysdioden.

Anslut aldrig en lysdiod till en Raspberry Pi utan en strömbegränsande resistor, såvida du inte vet att lysdioden har en inbyggd resistor med lämpligt värde.

Börja med att kontrollera att lysdioden fungerar. Vrid Raspberry Pi så att GPIO är i två vertikala rader till höger. Anslut den ena änden av 330 Ω-resistorn till det första 3,3 V-stiftet (märkt 3V3 på **Bild 6-1**) med en hona-till-hona-bygelkabel och anslut sedan den andra änden till det långa benet – positiva, eller anoden – på lysdioden med en annan hona-till-hona-bygelkabel. Ta en sista hona-till-hona-bygelkabel och anslut det korta benet – negativa, eller katoden – på lysdioden till det första jordstiftet (märkt GND på **Bild 6-1**).




▲ **Bild 6-1:** Anslut lysdidoden till dessa stift. Glöm inte resistorn!

Så länge Raspberry Pi är på bör lysdioden tändas. Om den inte gör det dubbelkollar du kretsen: se till att du inte har använt för högt resistorvärde, att alla ledningar är ordentligt anslutna och att du definitivt har valt rätt GPIO-stift som matchar kopplingsschemat. Kontrollera även lysdiodens ben, eftersom lysdioder bara fungerar åt ena hållet: med det längre benet anslutet till kretsens positiva sida och det kortare benet till den negativa.

När lysdioden fungerar är det dags att programmera den. Koppla bort bygelkabeln från 3,3 V-stiftet (märkt 3V3 på **Bild 6-2**) och anslut den till GPIO 25-stiftet (märkt GP25 på **Bild 6-2**). Lysdioden stängs av, men oroa dig inte – det är normalt.

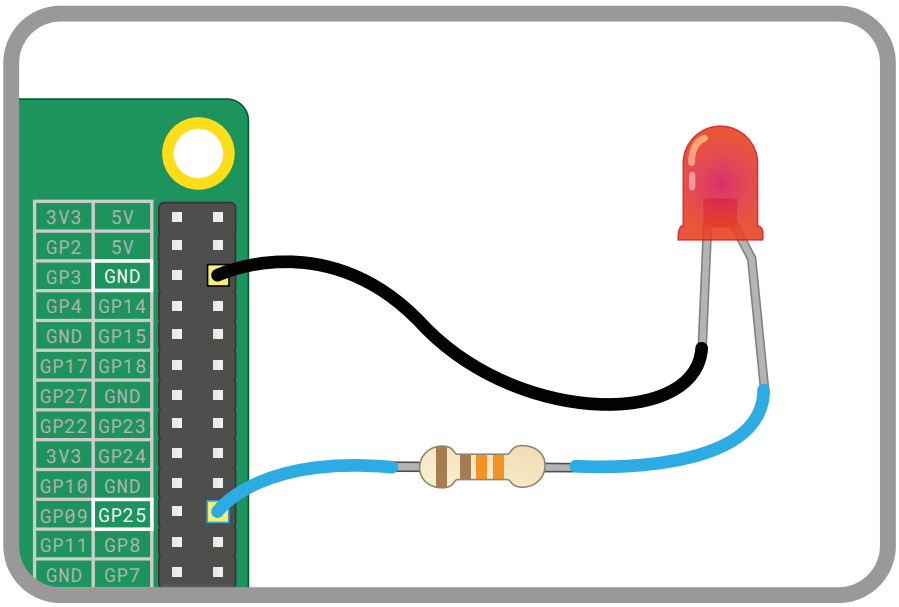
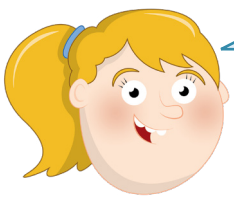



**Bild 6-2:** Koppla bort bygelkabeln från 3V3 och anslut den till GPIO 25-stiftet

Du är nu redo att skapa ett Scratch- eller Python-program för att aktivera och inaktivera lysdioden.



### KODNINGSKUNSKAP

Projekten i det här kapitlet kräver att du känner dig bekväm med Scratch 3 och den integrerade utvecklingsmiljön Thonny Python (IDE). Gå till **Kapitel 4, Programmering med Scratch 3** eller **Kapitel 5, Programmering med Python**, om du inte redan har gjort det, och arbeta igenom dessa projekt först.

## Styrning av lysdiod i Scratch

Läs in Scratch 3 och klicka på ikonen Lägg till ett tillägg. Rulla ner för att hitta tillägget ”Raspberry Pi GPIO” (**Bild 6-3**, på nästa sida) och klicka sedan på det. Detta läser in blocken du behöver för att styra Raspberry Pi:s GPIO från Scratch 3. De nya blocken visas i blockpaletten. När du behöver dem är de tillgängliga i kategorin Raspberry Pi GPIO.

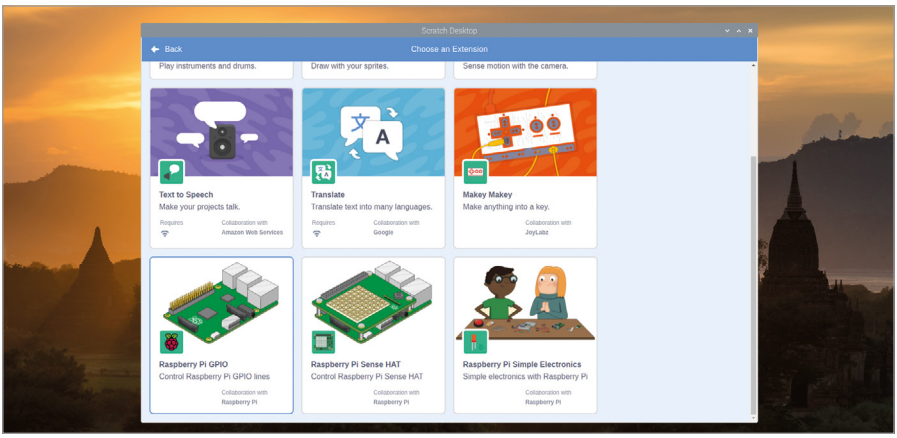
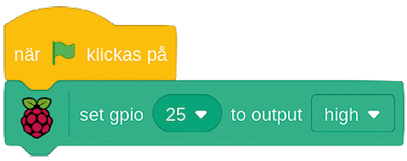
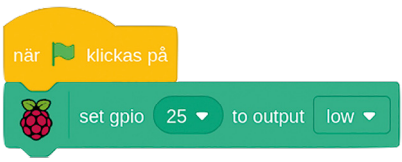



**Bild 6-3:** Lägg till Raspberry Pi GPIO-tillägget i Scratch 3

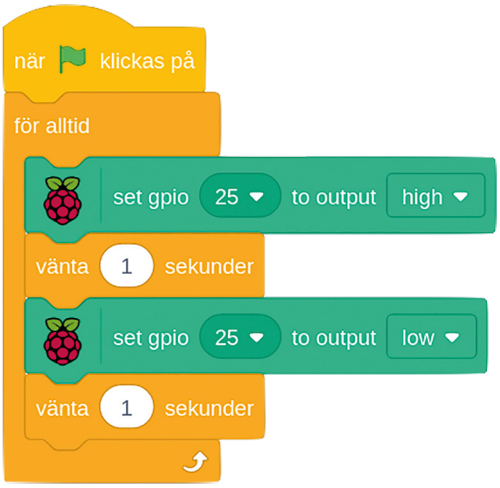
Börja med att dra ett **när klickas på**-Händelser-block till kodområdet och placera sedan ett **set gpio to output high**-block direkt inunder det. Du måste välja hur många stift du ska använda: klicka på den lilla pilen för att öppna listrutan och klicka på ”25” för att tala om för Scratch att du styr GPIO 25-stiftet.



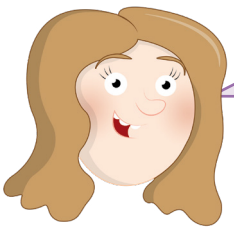

Klicka på den gröna flaggan för att köra programmet. Du får se lysdioden tändas: du har programmerat ditt första fysiska datortillämpningsprojekt! Klicka på den röda oktagonen för att stoppa programmet. Ser du att lysdioden fortsätter att lysa? Det beror på att programmet bara talade om för Raspberry Pi att tända lysdioden. Det är det den ”output high”-delen av **set gpio 25 to output high**-blocket betyder. För att stänga av den igen ska du klicka på nedåtpilen i slutet av blocket och välja ”low” i listan.




Klicka på den gröna flaggan igen. Den här gången släcker programmet lysdioden. För att göra saker mer intressanta ska du lägga till ett **för alltid**-Kontroll-block och ett par **vänta 1 sekunder**-block för att skapa ett program som får lysdioden att blinka på och av varje sekund.




Klicka på den gröna flaggan och titta på lysdioden: den tänds i en sekund, stängs av i en sekund, tänds i en sekund och fortsätter att upprepa mönstret tills du klickar på den röda oktagonen för att stoppa det. Se vad som händer när du klickar på oktagonen medan lysdioden är på eller av.



**UTMANING:** KAN DU ÄNDRA DET?

Hur kan man ändra programmet så att lysdioden lyser längre? Hur gör man så att den är avstängd längre? Vilken är den minsta fördröjningen du kan använda och ändå se lysdioden växla mellan av och på?

## Styrning av lysdiod i Python

Ladda Thonny från avsnittet Programmering på hallonmenyn och klicka sedan på knappen New för att starta ett nytt projekt och Save för att spara det som **Hej lysdiod**. För att använda GPIO-stiften från Python behöver du ett bibliotek som heter GPIO Zero. För detta projekt behöver du bara den del av biblioteket som gäller arbete med lysdioder. Importera bara den här delen av biblioteket genom att skriva följande i Pythons skalområde:

```
from gpiozero import LED
```

Därefter måste du tala om för GPIO Zero vilket GPIO-stift som lysdioden är ansluten till. Skriv följande:

```
led = LED(25)
```

Tillsammans ger dessa två rader Python möjlighet att styra lysdioder som är anslutna till Raspberry Pi:s GPIO-stift och tala om för det vilket/vilka stift, om du har mer än en lysdiod i kretsen, som ska styras. Skriv följande för att faktiskt styra lysdioden:

```
led.on()
```

För att stänga av lysdioden igen, skriv:

```
led.off()
```

Grattis, nu har du kontroll över Raspberry Pi:s GPIO-stift i Python! Försök skriva dessa två instruktioner igen. Om lysdioden redan är släckt kommer inte **`led.off()`** att göra någonting. Detsamma gäller om lysdioden redan är tänd och du skriver **`led.on()`**.

För att skapa ett riktigt program ska du skriva följande i skriptområdet:

```
from gpiozero import LED
from time import sleep

led = LED(25)

while True:
    led.on()
    sleep(1)
    led.off()
    sleep(1)
```

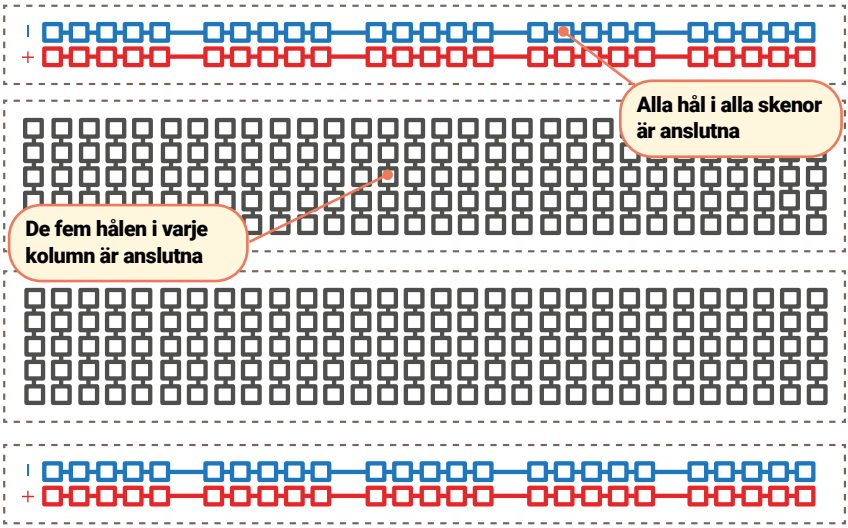
Detta program importerar lysdiodsfunktionen från biblioteket **`gpiozero`** (GPIO Zero) och funktionen **`sleep`** från biblioteket **`time`** och konstruerar sedan en oändlig loop som tänder lysdioden i en sekund, stänger av den i en sekund och sedan upprepar detta. Klicka på knappen Run för att se den i aktion: lysdioden börjar blinka. Precis som med Scratch-programmet ska du observera beteendet när du klickar på Stop-knappen medan lysdioden lyser kontra medan lysdioden är av.

**UTMANING:** LÄNGRE LYSTID

Hur kan man ändra programmet så att lysdioden lyser längre? Hur gör man så att den är avstängd längre? Vilken är den minsta fördröjningen du kan använda och ändå se lysdioden växla mellan av och på?

## Använda en kopplingsplatta

Nästa projekt i detta kapitel kommer att bli mycket lättare att slutföra om du använder en kopplingsplatta som du kan sätta fast komponenterna på och göra de elektriska anslutningarna.




En kopplinsplatta är täckt med hål som är 2,54 mm från varandra för att matcha komponenter. Under dessa hål finns metallremsor som fungerar som de bygeltrådar du använt hittills. Dessa löper i rader över plattan, och de flesta plattor har ett mellanrum i mitten så att de delas upp i två halvor. Många kopplingsplattor har också bokstäver överst och siffror längs sidorna. Dessa gör det möjligt för dig att hitta ett visst hål: A1 är det övre vänstra hörnet, B1 är nästa hål till höger medan B2 är ett hål ner därifrån. A1 är ansluten till B1 med de dolda metallremsorna, men inget 1-hål är anslutet till ett 2-hål, om du inte själv lägger till en bygeltråd.

Större kopplingsplattor har också rader med hål längs sidorna, vanligtvis markerade med röda och svarta eller röda och blå ränder. Detta är *strömskenorna*, och de är utformade för att göra kabeldragningen enklare: du kan ansluta en enda tråd från Raspberry Pi:s jordstift till en av strömskenorna – vanligtvis markerade med en blå eller svart rand och en minussymbol – för att tillhandahålla en *gemensam jord* för många komponenter på kopplingsplattan, och du kan göra detsamma om kretsen behöver spänning på 3,3 V eller 5 V.

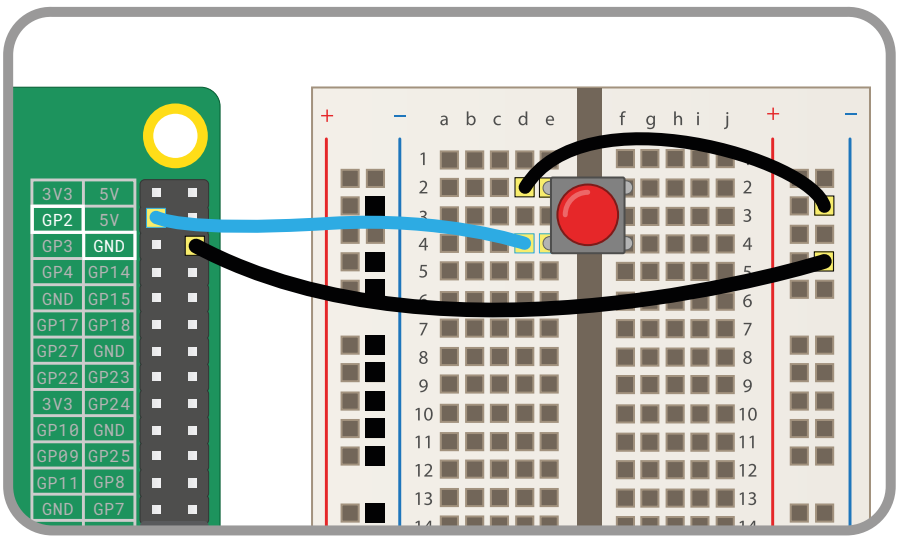
Det är enkelt att lägga till elektroniska komponenter på en kopplingsplatta: justera bara ledningarna (de utstickande metalldelarna) efter hålen och tryck försiktigt tills komponenten är på plats. För anslutningar du behöver göra utöver de som kopplingsplattan gör åt dig, kan du använda bygelkabeln hane-till-hane (M2M). För anslutningar från kopplingsplattan till Raspberry Pi ska du använda bygelkabeln hane-till-hona (M2F).

Försök aldrig klämma in mer än en komponentsladd eller bygelledning i ett enda hål på kopplingsplattan. Kom ihåg: hålen är anslutna i kolumner, bortsett från delningen i mitten, så en komponentledning i A1 är elektriskt ansluten till allt du kopplar till B1, C1, D1 och E1.

## Nästa steg: läsa en knapp

Utgångar som lysdioder är en sak, men ”in/ut”-delen av ”GPIO” betyder att du även kan använda stift som ingångar. Till det här projektet behöver du en kopplingsplatta, bygelkabel hane-till-hane (M2M) och bygelkabel hane-till-hona (M2F) och en tryckknappsbrytare. Om du inte har någon kopplingsplatta kan du använda bygelkablar hona-till-hona (F2F), men knappen blir mycket svårare att trycka in utan att man råkar bryta kretsen.

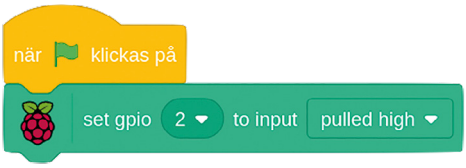
Börja med att lägga till tryckknappen på kopplingsplattan. Om tryckknappen bara har två ben, se till att de finns i olika numrerade rader på panelen. Men om den har fyra ben ska du vrida den så att sidorna som benen kommer ut ur går längs kopplingsplattans rader och de plana benfria sidorna är överst och nederst. Anslut jordskenan på kopplingsplattan till ett jordstift på Raspberry Pi (märkt GND på **Bild 6-4**) med en hane-till-hona-bygelkabel och anslut sedan ett ben på tryckknappen till jordskenan med en hane-till-hane-bygelkabel. Slutligen ansluter du det andra benet, det som är på samma sida som det ben du just anslutit om du använder en fyrbensomkopplare, till GPIO 2-stiftet (märkt GP2 på **Bild 6-4**) på Raspberry Pi med en hane-till-hona-bygelkabel.



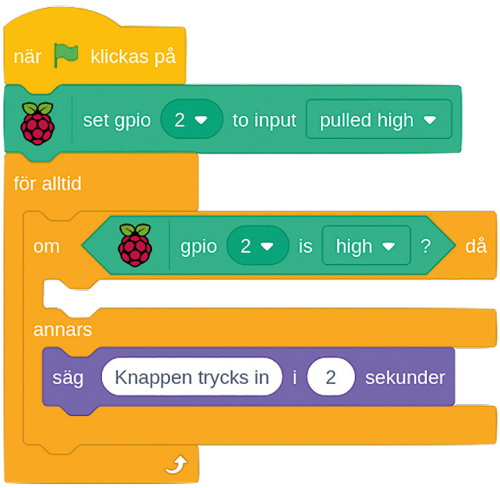

▲ **Bild 6-4:** Dra ledning från en tryckknapp till GPIO-stiften

## Läsa en knapp i Scratch

Starta ett nytt Scratch-program och dra ett när klickas på-block till kodområdet. Anslut ett set gpio to input pulled high-block och välj nummer 2 i listrutan för att matcha GPIO-stiftet som du använde för tryckknappen.




Om du klickar på den gröna flaggan nu kommer inget att hända. Det beror på att du har sagt åt Scratch att använda stiftet som en ingång, men inte vad som ska göras med den. Dra ett **för alltid**-block till slutet av sekvensen och dra sedan ett **om då annars**-block inuti det. Leta upp **gpio is high?**-blocket, dra in det i det diamantformade utrymmet i blockets **om då** -del och använd listrutan för att välja nummer 2 för att berätta vilket GPIO-stift som ska kontrolleras. Dra ett **säg Hej! i 2 sekunder**-block in i blockets **annars**-del och redigera det så att det står ”Knappen trycks in!”. Lämna ”om då”-delen av blocket tom för tillfället.




Det händer mycket nu, men börja med att testa det: klicka på den gröna flaggan och tryck sedan på knappen på kopplingsplattan. Sprajten borde tala om att knappen har tryckts in: du har nu läst en inmatning från GPIO-stiftet!

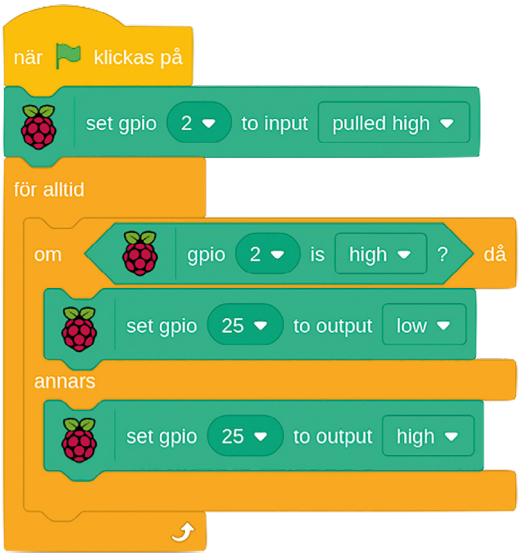
Du kanske har märkt att **om gpio 2 is high? då**-delen av blocket är tom. Koden som körs när knappen faktiskt trycks in befinner sig under tiden i blockets **annars**-del. Det kan verka förvirrande, eftersom att trycka på knappen borde få den att gå till på? Det är faktiskt precis tvärtom: Raspberry Pi:s GPIO-stift är normalt höga eller på, när de är inställda som en ingång, och när knappen trycks in dras de ner till av.

Titta på kretsen igen: se hur knappen är ansluten till GPIO 2-stiftet, som ger den positiva delen av kretsen, och jordstiftet. När du trycker på knappen dras spänningen på GPIO-stiftet ner genom jordstiftet och Scratch-programmet slutar köra koden i **om gpio 2 is high? då** -blocket och kör istället koden i **annars**-delen av blocket.

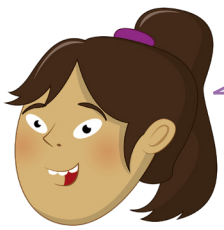
Om det låter förvirrande, kom bara ihåg det här: en knapp på ett Raspberry Pi GPIO-stift trycks in när stiftet blir lågt, inte när det går högt!

För att förlänga programmet ytterligare ska du lägga till lysdioden och resistorn i kretsen igen: kom ihåg att ansluta resistorn till GPIO 25-stiftet och det långa benet på lysdioden och den kortare delen av lysdioden till jordskenan på kopplingsplattan.

Dra **säg Knappen trycks in! i 2 sekunder**-blocket från kodområdet till blockpaletten för att radera det, ersätt det sedan med ett **set gpio 25 to output high**-block. Kom ihåg att du måste ändra GPIO-numret med hjälp av listrutan. Lägg till ett **set gpio 25 to output low**-block – kom ihåg att ändra värdena – till den för närvarande tomma **if gpio 2 is high? then**-delen av blocket.




Klicka på den gröna flaggan och tryck på knappen. Lysdioden tänds så länge du håller ned knappen. Om du släpper slocknar den igen. Grattis: du styr ett GPIO-stift baserat på en inmatning från ett annat!



## UTMANING: SE TILL ATT DEN FÖRBLIR TÄND

Hur kan man ändra programmet så att lysdioden lyser några sekunder, även efter att du släppt knappen? Vad skulle man behöva ändra för att lysdioden ska vara på när du inte trycker på knappen och av när du gör det?

## Läsa en knapp i Python

Klicka på knappen New i Thonny för att starta ett nytt projekt och på knappen Save för att spara det som **Knappinmatning**. Att använda ett GPIO-stift som ingång för en knapp liknar att använda ett stift som utgång för en lysdiod, men du måste importera en annan del av GPIO Zero-biblioteket. Skriv följande i skriptområdet:

```
from gpiozero import Button
button = Button(2)
```

För att köra koden när du trycker på knappen tillhandahåller GPIO Zero funktionen **wait_for_press**. Skriv följande:

```
button.wait_for_press()
print("Du tryckte på mig!")
```

Klicka på knappen Run och tryck sedan på tryckknappsbrytaren. Meddelandet kommer att skrivas till Python-skalet längst ner i Thonny-fönstret: du har nu läst en inmatning från GPIO-stiftet! Om du vill prova programmet igen måste du klicka på Run igen. Eftersom det inte finns någon loop i programmet avslutas det så snart det har skrivit meddelandet till skalet.

För att förlänga programmet ytterligare ska du lägga till lysdioden och resistorn i kretsen igen, om du inte redan gjort det: kom ihåg att ansluta resistorn till GPIO 25-stiftet och det långa benet på lysdioden, och den kortare delen av lysdioden till jordskenan på kopplingsplattan.

För att styra en lysdiod och läsa en knapp måste du importera båda funktionerna **Button** och **LED** från GPIO Zero-biblioteket. Du behöver även funktionen **sleep** från biblioteket **time**. Gå tillbaka till överdelen av programmet och skriv in följande som de två nya översta raderna:

```
from gpiozero import LED
from time import sleep
```

Nedanför linjen **button = Button(2)** ska du skriva:

```
led = LED(25)
```

Radera raden **print("Du tryckte på mig!")** och ersätt den med:

```
led.on()
sleep(3)
led.off()
```

Det färdiga programmet ska nu se ut så här:

```
from gpiozero import LED
from time import sleep
from gpiozero import Button

button = Button(2)
led = LED(25)
button.wait_for_press()
led.on()
sleep(3)
led.off()
```

Klicka på knappen Run och tryck sedan på tryckknappsbrytaren: lysdioden kommer att tändas i tre sekunder, sedan stängs den av igen och programmet stoppas. Grattis: du kan styra en lysdiod med en knappingång i Python!

### **UTMANING:** LÄGG TILL EN LOOP

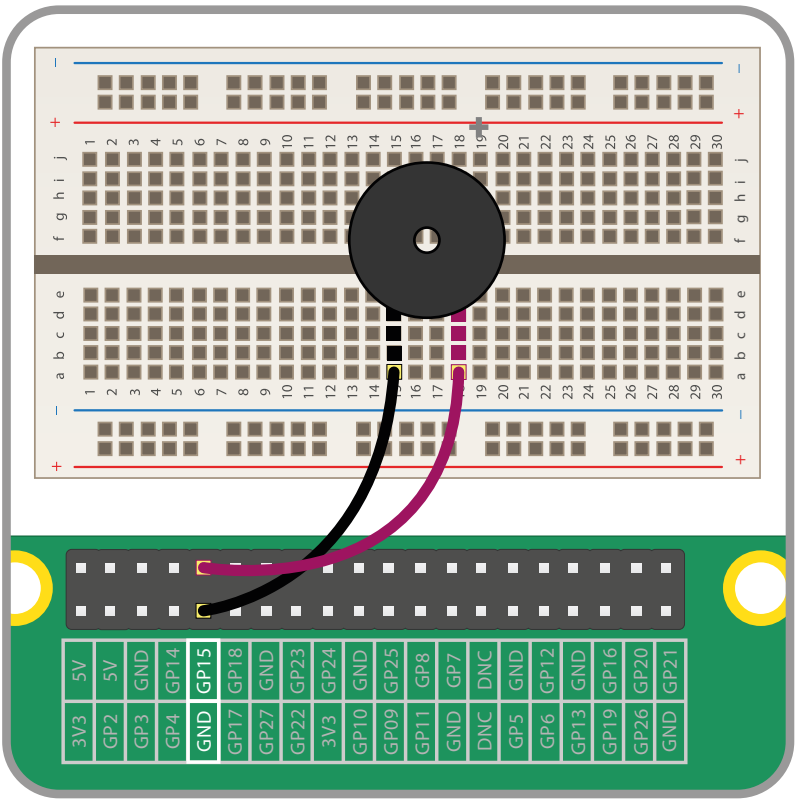
Hur skulle du lägga till en loop för att programmet ska upprepas istället för att avsluta efter en knapptryckning? Vad skulle man behöva ändra för att lysdioden ska vara på när du inte trycker på knappen och av när du gör det?

## Skapa oljud: att styra en summer

Lysdioder är en bra utdataenhet, men inte till mycket nytta om du tittar åt något annat håll. Lösningen: summer, som ger ifrån sig ett ljud som hörs i hela rummet. Till det här projektet behöver du en kopplingsplatta, bygelkabel hane-till-hona (M2F) och en aktiv summer. Om du inte har någon kopplingsplatta kan du ansluta summern med hjälp av bygelkablar hona-till-hona (F2F) istället.

En aktiv summer kan behandlas exakt som en lysdiod när det gäller kretsar och programmering. Upprepa kretsen du skapade för lysdioden, men ersätt lysdioden med den aktiva summern. Resistorn ska dock inte vara med eftersom summern behöver mer ström för att fungera. Anslut ett ben på summern till GPIO 15-stiftet (märkt GP15 på **Bild 6-5**) och det andra till jordstiftet (märkt GND i kopplingsschemat) med hjälp av kopplingsplattan och hane-till-hona-bygelkablar.

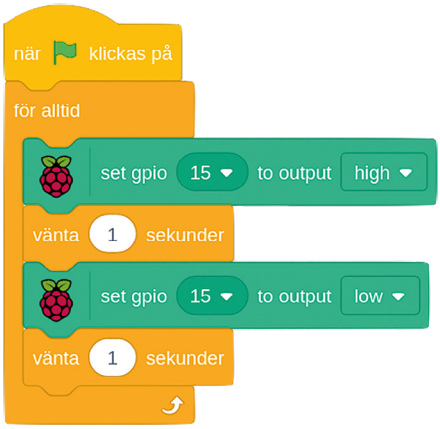
Om summern har tre ben, se till att benet som är märkt med en minussymbol (-) är anslutet till jordstiftet och att benet som är märkt med ”S” eller ”SIGNAL” är anslutet till GPIO 15, och sedan ska du ansluta det återstående benet – vanligtvis mittbenet – till 3,3 V-stiftet (märkt 3V3.)




▲ **Bild 6-5:** Ansluta en summer till GPIO-stiften

## Styra en summer i Scratch

Återskapa samma program som fick lysdioden att blinka – eller ladda det om du sparade det innan du skapade knapprojektet. Använd listrutan i **set gpio to output high**-blocken för att välja nummer 15, så att Scratch styr rätt GPIO-stift.




Klicka på den gröna flaggan så kommer summern att surra: en sekund på och en sekund av. Om du bara hör att summern klickar en gång i sekunden använder du en passiv summer istället för en aktiv summer. Där en aktiv summer genererar den snabbt föränderliga signalen, som även kallas *oscillation*, för att få metallplattorna att vibrera i sig själva, behöver en passiv summer en oscillerande signal. När du bara aktiverar den med Scratch rör sig plattorna bara en gång och stannar sedan, vilket ger ”klick”-ljudet, till nästa gång programmet aktiverar eller stänger av stiftet.

Klicka på den röda oktagonen för att stoppa summern, men se till att göra det när den inte avger ljud, annars fortsätter summern att surra tills du kör programmet igen!

### UTMANING: ÄNDRA SURRET

Hur kan man ändra programmet så att summern avger ljud en kortare tid? Kan du bygga en krets så att summern styrs av en knapp?

## Styra en summer i Python

Att styra en aktiv summer via GPIO Zero-biblioteket är nästan identiskt med att styra en lysdiod, eftersom den har på- och av-lägen. Du behöver dock en annan funktion: **`buzzer`**. Skapa ett nytt projekt i Thonny och spara det som **Buzzer**, skriv sedan följande:

```
from gpiozero import Buzzer
from time import sleep
```

Precis som med lysdioder behöver GPIO Zero veta vilket stift summern är ansluten till för att kunna styra den. Skriv följande:

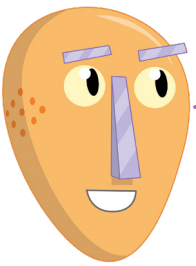
```
buzzer = Buzzer(15)
```

Härifrån är programmet nästan identiskt med det du skrev för att styra lysdioden. Den enda skillnaden (förutom ett annat GPIO-stiftnummer) är att du använder **`buzzer`** istället för **`led`**. Skriv följande:

```
while True:
    buzzer.on()
    sleep(1)
    buzzer.off()
    sleep(1)
```

Klicka på knappen Run så börjar summern att surra: en sekund på och en sekund av. Om du använder en passiv summer istället för en aktiv hörs bara ett kort klick varje sekund istället för ett kontinuerligt surrande: detta beror på att en passiv summer saknar en *oscillator* som kan skapa den snabbt föränderliga signalen som får plattorna inuti summern att vibrera.

Klicka på Stop-knappen för att avsluta programmet, men se till att summern inte låter vid detta tillfälle, annars fortsätter den att surra tills du kör programmet igen!



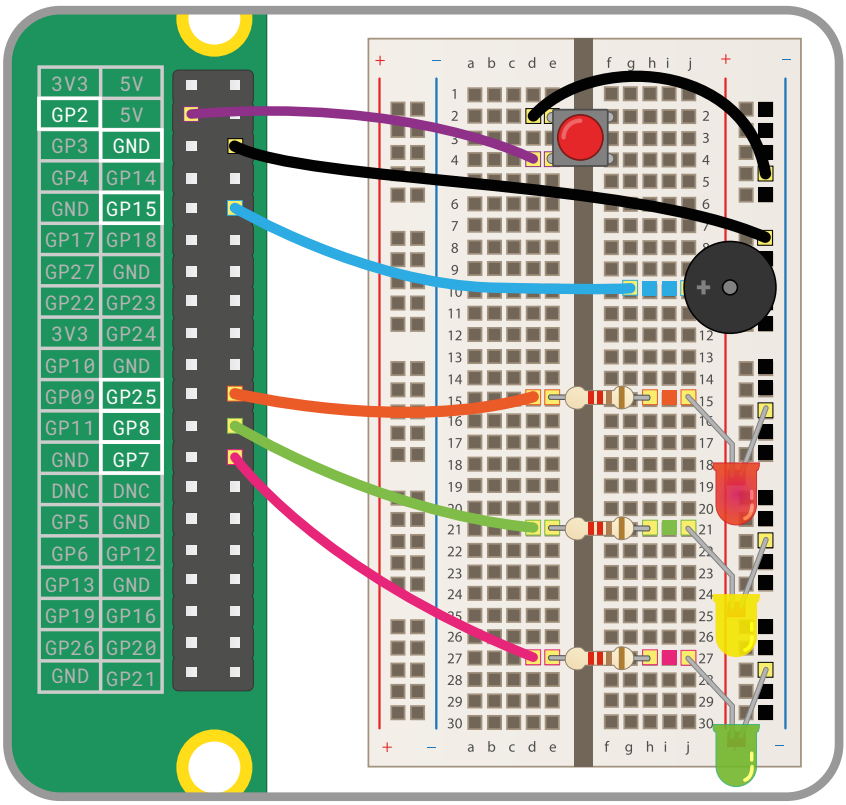
**UTMANING:** ETT BÄTTRE SURR

Hur kan man ändra programmet så att summern avger ljud en kortare tid? Kan du bygga en krets så att summern styrs av en knapp?

## Scratch-projekt: Trafikljus

Nu när du vet hur man använder knappar, summer och lysdioder som in- och utgångar är du redo att bygga ett exempel på verklig datortillämpning: trafikljus, komplett med en knapp som du kan trycka på för att kunna gå över vägen. Till det här projektet behöver du en kopplingsplatta, en röd, en gul och en grön lysdiod, tre resistorer på 330 Ω, en summer, en tryckknappsbrytare och ett urval av bygelkablar hane-till-hane (M2M) och hane-till-hona (M2F).

Börja med att bygga kretsen (**Bild 6-6**) genom att ansluta summern till GPIO 15-stiftet (märkt GP15 på **Bild 6-6**), den röda lysdioden till GPIO 25-stiftet (märkt GP25), den gula lysdioden till GPIO 8 (GP8), den gröna lysdioden till GPIO 7 (GP7) och brytaren till GPIO 2 (GP2). Kom ihåg att ansluta 330 Ω-resistorn mellan GPIO-stiften och de långa benen på lysdioderna och anslut de andra benen på alla komponenterna till jordskenan på kopplingsplattan. Slutligen ska du ansluta jordskenan till ett jordstift (märkt GND) på Raspberry Pi för att slutföra kretsen.



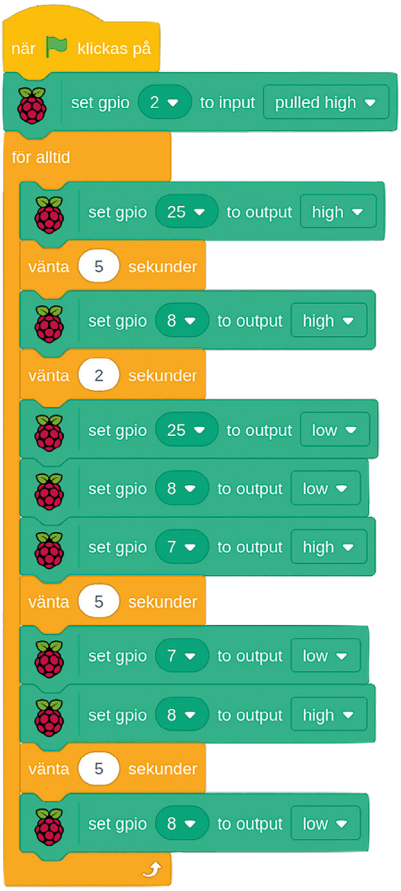

▲ **Bild 6-6:** Kopplingsschema för trafikljusprojektet

Starta ett nytt Scratch 3-program och dra ett 

-block till kodområdet. Därefter måste du tala om för Scratch att GPIO 2-stiftet, som är

anslutet till tryckknappsbrytaren i kretsen, är en ingång istället för en utgång: dra ett **set gpio to input pulled high**-block från Raspberry Pi GPIO-kategorin i blockpaletten under **när klickas på**-blocket. Klicka på nedåtpilen bredvid ”0” och välj nummer 2 i listrutan.

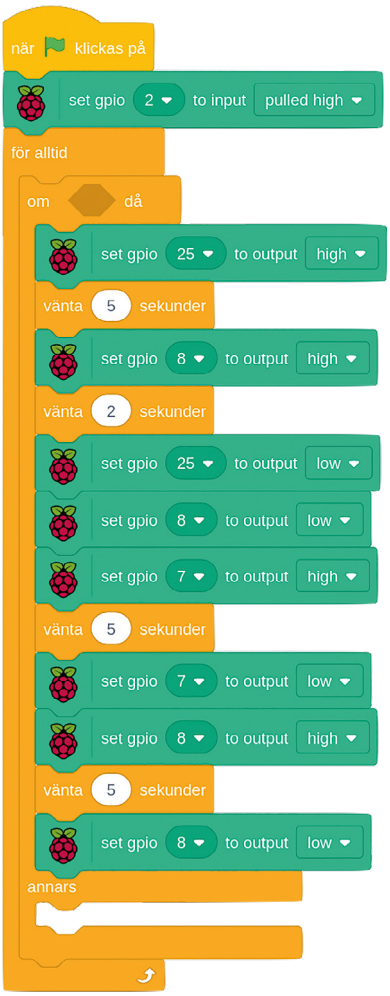

Därefter måste du skapa trafikljussekvensen. Dra in ett **för alltid**-block i programmet och fyll det sedan med block för att slå på och av trafikljuslamporna i ett mönster. Kom ihåg vilka komponenter som är anslutna till vilka GPIO-stift: när du använder stift 25 använder du den röda lysdioden, stift 8 den gula lysdioden och stift 7 den gröna lysdioden.



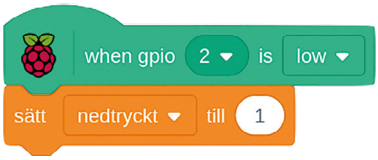

Klicka på den gröna flaggan och titta på lysdioderna: först tänds den röda, sedan både den röda och gula, sedan den gröna, sedan den gula och slutligen upprepas sekvensen med det

röda ljuset en gång till. Detta mönster matchar det som används av trafikljus i Storbritannien. Du kan redigera sekvensen så att den matchar mönster i andra länder om du vill.

För att simulera ett övergångsställe måste programmet hålla utkik efter att knappen trycks ned. Klicka på den röda oktagonen för att stoppa programmet om det för närvarande körs. Dra ett **om då annars**-block till skriptområdet och anslut det så att det ligger direkt under **för alltid**-blocket, med trafikljussekvensen i avsnittet ”om då”. Låt den diamantformade springan vara tom för tillfället.

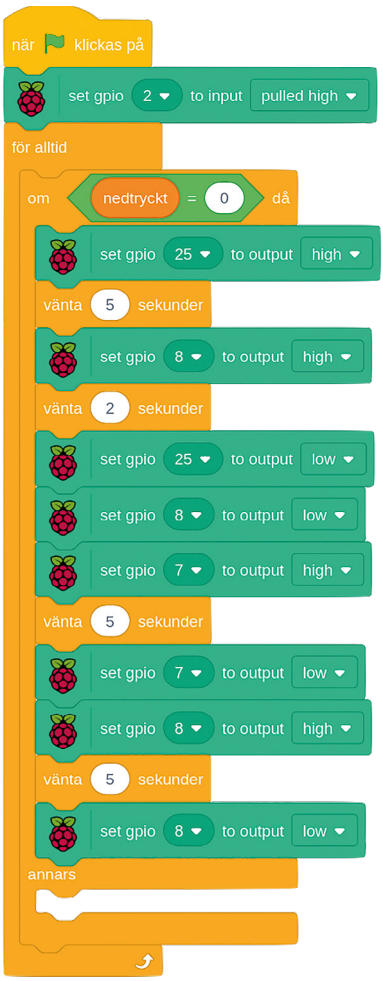



Ett riktigt övergångsställe kan inte ändra ljuset till rött så snart knappen trycks in, utan väntar istället på nästa röda lampa i sekvensen. För att bygga in det i ditt eget program ska du dra ett **when gpio is low**-block till kodområdet och välja ”2” från listrutan. Dra sedan ett **sätt nedtryckt till 1**-block under det.




Denna blockstapel håller utkik efter att knappen trycks in och sätter sedan variabeln ”tryckt” till 1. Genom att ställa in en variabel på detta sätt kan du lagra det faktum att knappen har tryckts in, även om du inte kommer att agera direkt.

Gå tillbaka till den ursprungliga blockstapeln och hitta om då-blocket. Slutligen drar du ett = -Operatorer-block till om då-blockets diamantformade mellanrum och drar sedan ett nedtryckt-rapporteringsblock till det första mellanslaget. Skriv ”0” över ”50” på blockets högra sida.



Klicka på den gröna flaggan och se hur trafikljusen går igenom sekvensen. Tryck på tryckknappsbrytaren: först ser det ut som ingenting händer, men när sekvensen har nått sitt slut – bara den gula lysdioden är tänd – kommer trafikljuset att släckas och förbli släckt tack vare din ”tryckta” variabel.

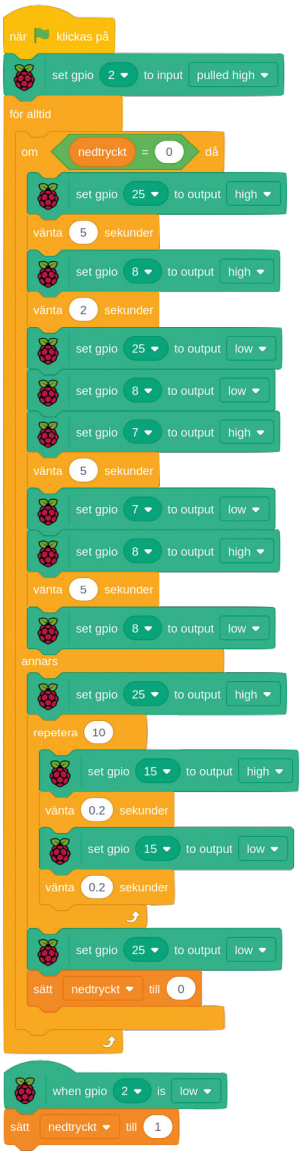
Allt som återstår är att se till att knappen till övergångsstället verkligen gör något annat än att släcka lamporna. I huvudblockstapeln ska du leta upp **annars** -blocket och dra ett **set gpio 25 to output high** -block in i det. Kom ihåg att ändra det förvalda GPIO-stiftnumret så att det matchar stiftet som den röda lysdioden är ansluten till.

Under det, fortfarande i **annars** -blocket, ska du skapa ett mönster för summern: dra ett **repetera 10** -block, fyll det sedan med blocken **set gpio 15 to output high**, **vänta 0.2 sekunder**, **set gpio 15 to output low** och **vänta 0.2 sekunder** och ändra GPIO-stiftvärdena så att de matchar stiftet för summerkomponenten.

Slutligen, under **repetera 10** -blocket men fortfarande i **annars** -blocket, ska du lägga till ett **set gpio 25 to output low** -block och ett **sätt nedtryckt till 0** -block – det sista blocket som återställer variabeln som lagrar knapptrycket, så summerns sekvens inte bara upprepas för evigt.

Klicka på den gröna flaggan och tryck sedan på brytaren på kopplingsplattan. När sekvensen har slutförts ser du det röda ljuset tändas och summern låter så att fotgängare vet att det är säkert att gå över gatan. Efter några sekunder stannar summern och trafikljussekvensen startar igen och fortsätter tills nästa gång du trycker på knappen.

Grattis: du har programmerat en egen fullt funktionell uppsättning trafikljus, komplett med övergångsställe!



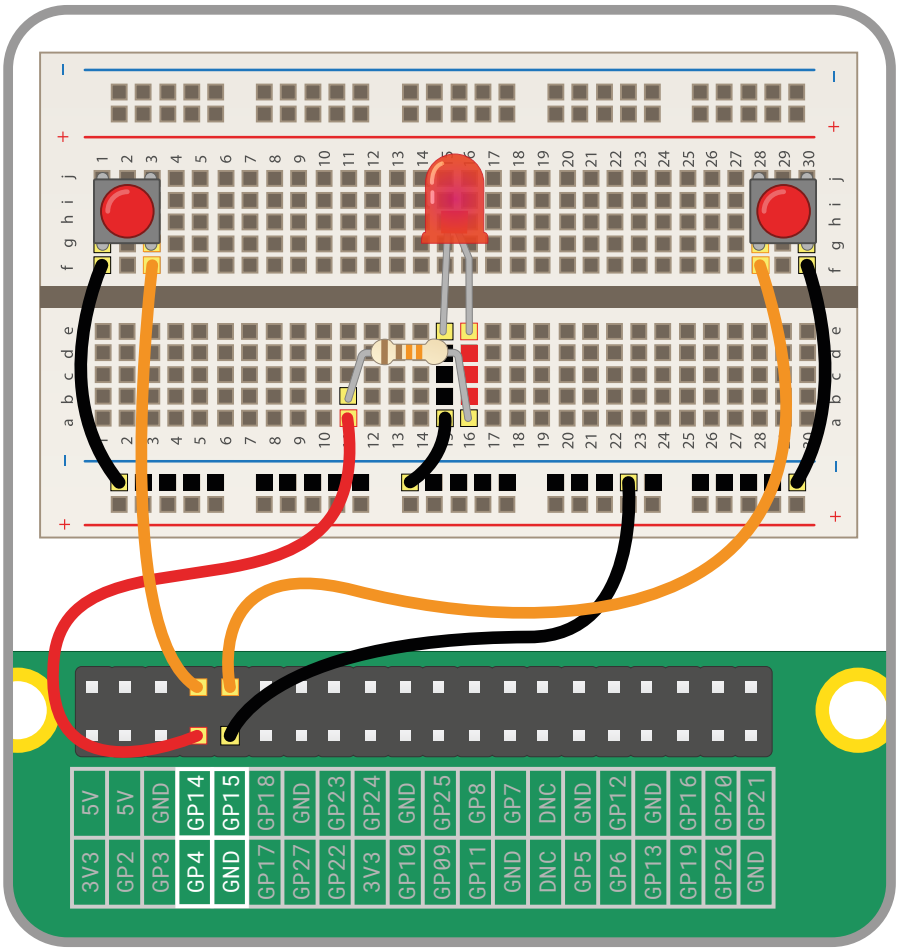

## UTMANING: KAN DU FÖRBÄTTRA DET?

Kan du ändra programmet så att fotgängaren får längre tid på sig att gå över vägen? Kan du hitta information om andra länders trafikljusmönster och programmera om lamporna så att de passar detta? Hur kan man göra för att lysdioderna ska lysa med mindre ljusstyrka?

## Python-projekt: Spel med snabb reaktion

Nu när du vet hur man använder knappar, summer och lysdioder som in- och utgångar är du redo att bygga ett exempel på verklig datortillämpning: ett spel med snabba reaktioner för två spelare, utformat för att visa vem som har snabbast reaktionstid! Till det här projektet behöver du en kopplingsplatta, en lysdiod, ett resistor på 330 Ω, två tryckknappsbrytare och några bygelkablar hane-till-hona (M2F) och hane-till-hane (M2M).

Börja med att bygga kretsen (**Bild 6-7**): anslut den första brytaren till vänster på kopplingsplattan till GPIO 14-stiftet (märkt GP14 på **Bild 6-7**), den andra brytaren på höger sida av kopplingsplattan till GPIO 15-stiftet (märkt GP15), lysdiodens längre ben till 330 Ω-resistorn som sedan ansluts till GPIO 4-stiftet (märkt GP4) på Raspberry Pi och andra benen på alla komponenterna till kopplingsplattans jordskena. Slutligen ska du ansluta jordskenan till Rapberry Pi:s jordstift (märkt GND).




▲ **Bild 6-7:** Kopplingsschema för Spel med snabb reaktion

Skapa ett nytt projekt i Thonny och spara det som **Reaktionsspel**. Du kommer att använda funktionerna **`LED`** och **`button`** från GPIO Zero-biblioteket och funktionen **`sleep`** från time-biblioteket. Istället för att importera de två GPIO Zero-funktionerna på två separata rader kan du spara tid och importera dem tillsammans med en kommasymbol (**`,`**) för att separera dem. Skriv följande i skriptområdet:

```
from gpiozero import LED, Button
from time import sleep
```

Som tidigare måste du tala om för GPIO Zero vilka stift de två knapparna och lysdioden är anslutna till. Skriv följande:

```
led = LED(4)
right_button = Button(15)
left_button = Button(14)
```

Lägg nu till instruktioner för att aktivera och stänga av lysdioden så att du kan kontrollera att den fungerar korrekt:

```
led.on()
sleep(5)
led.off()
```

Klicka på knappen Run: lysdioden kommer att tändas i fem sekunder, sedan stängs den av och programmet stoppas. I ett reaktionsspel är det dock lite förutsägbart att lysdioden släcks efter exakt 5 sekunder varje gång. Lägg till följande nedanför raden **`from time import sleep`**:

```
from random import uniform
```

Det slumpmässiga biblioteket, som namnet antyder, gör det möjligt för dig att generera slumptal (här med en enhetlig fördelning – se **rpf.io/uniform**). Leta upp raden **`sleep(5)`** och ändra den till:

```
sleep(uniform(5, 10))
```

Klicka på Run igen: den här gången lyser lysdioden i ett slumpmässigt antal sekunder mellan 5 och 10. Räkna för att se hur lång tid det tar för lysdioden att släcka och klicka sedan på knappen Run några gånger till: du ser att tiden är olika vid varje körning, vilket gör programmets resultat mindre förutsägbart.

För att förvandla knapparna till utlösare för var och en av spelarna måste du lägga till en funktion. Gå allra längst ner i programmet och skriv följande:

```
def pressed(button):
    print(str(button.pin.number) + " vann spelet")
```

Kom ihåg att Python använder indrag för att veta vilka rader som är en del av funktionen. Thonny drar automatiskt in den andra raden åt dig. Slutligen ska du lägga till följande två rader för att upptäcka spelarna som trycker på knapparna – kom ihåg att de inte får vara indragna, annars kommer Python att behandla dem som en del av funktionen.

```
right_button.when_pressed = pressed
left_button.when_pressed = pressed
```

Kör programmet och prova den här gången att trycka på en av de två knapparna så snart lysdioden slocknar. Du får se ett meddelande om att den första knappen ska tryckas in skrivas till Python-skalet längst ner i Thonny-fönstret. Tyvärr får du även meddelanden för varje gång någon av knapparna trycks in – och de använder stiftnumret istället för ett trevligt namn för knappen.

För att fixa det ska du börja med att be spelarna om deras namn. Under raden **`from random import uniform`** ska du skriva följande:

```
left_name = input("Vänster spelare heter ")
right_name = input("Höger spelare heter ")
```

Gå tillbaka till funktionen och ersätt raden **`print(str(button.pin.number) + " vann spelet")`** med:

```
    if button.pin.number == 14:
        print (left_name + " vann spelet")
    else:
        print(right_name + " vann spelet")
```

Klicka på knappen Run och skriv sedan namnen på båda spelarna i Pythons skalområde. När du trycker på knappen den här gången och kommer ihåg att göra det så fort du kan när lysdioden slocknar ser du att spelarens namn skrivs istället för stiftnumret.

För att lösa problemet med att alla knapptryckningar rapporteras som en vinst måste du lägga till en ny funktion från biblioteket sys, förkortning för *system*: **`exit`**. Under den sista raden med **`import`** ska du skriva följande:

```
from os import _exit
```

Vid slutet av funktionen, under raden **`print(right_name + " vann spelet")`**, ska du skriva följande:

```
    _exit(0)
```

Indragningen är viktig här: **`_exit(0)`** ska vara indragen med fyra mellanslag, i linje med **`else:`** två rader ovanför och **`if`** två rader ovanför detta. Denna instruktion talar om för Python att stoppa programmet efter att den första knappen har tryckts. Det innebär att spelaren vars knapp trycks in som tvåa inte får någon belöning för att förlora!

Det färdiga programmet ska nu se ut så här:

```
from gpiozero import LED, Button
from time import sleep
from random import uniform
from os import _exit

left_name = input("Vänster spelare heter ")
right_name = input("Höger spelare heter ")
led = LED(4)
right_button = Button(15)
left_button = Button(14)

led.on()
sleep(uniform(5, 10))
led.off()

def pressed(button):
    if button.pin.number == 14:
        print(left_name + " vann spelet")
    else:
        print(right_name + " vann spelet")
    _exit(0)

right_button.when_pressed = pressed
left_button.when_pressed = pressed
```

Klicka på knappen Run, ange spelarnas namn och vänta tills lysdioden slocknar så ser du namnet på den vinnande spelaren. Du kommer också att se ett meddelande från Python: **`Backend terminated or disconnected . Use 'Stop/Restart' to restart ...`**Detta betyder helt enkelt att Python fick kommandot **`_exit(0)`** och stoppade programmet, men att du måste klicka på Stop-ikonen för att avsluta helt och förbereda programmet för en ny omgång (**Bild 6-8**).

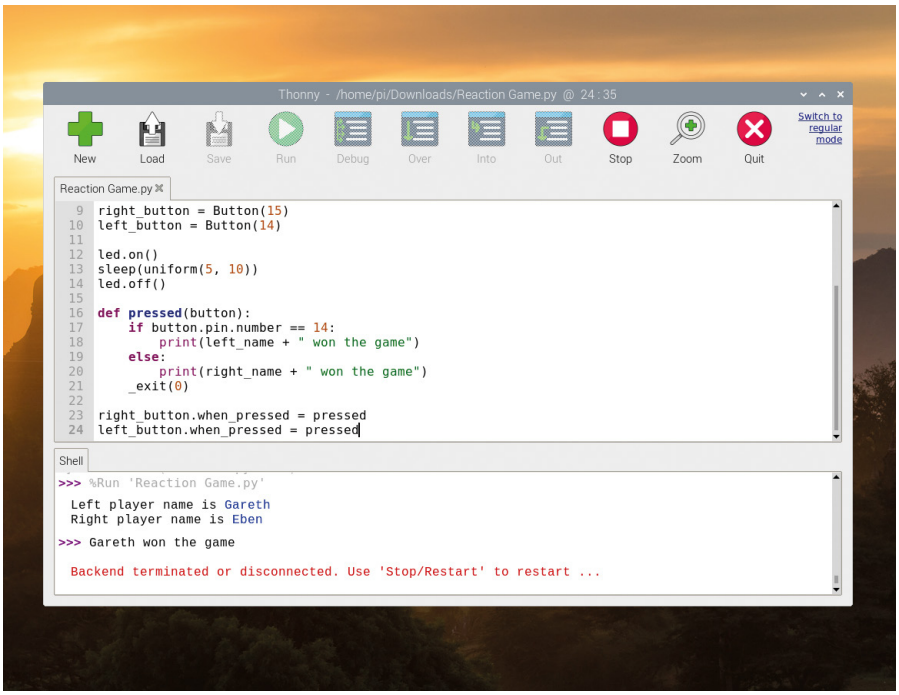
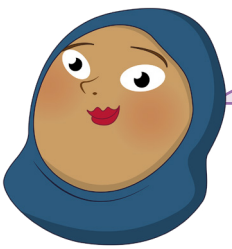



**Bild 6-8:** När vinnaren har utsetts måste du stoppa programmet

Grattis: du har skapat ett eget fysiskt spel!



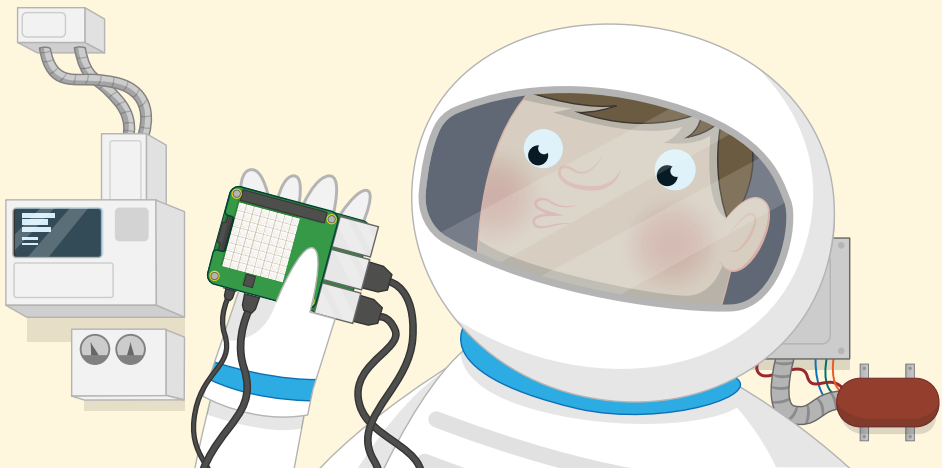
## UTMANING: FÖRBÄTTRA SPELET

Kan du lägga till en loop så att spelet körs kontinuerligt? Kom ihåg att ta bort instruktionen `_exit(0)` först! Kan du lägga till en poängräknare så att du kan se vem som vinner över flera omgångar? Vad sägs om en timer så att du kan se hur lång tid det tog för dig att reagera på att lampan slocknade?

Kapitel 7

# Fysisk databehandling med Sense HAT

**Sense HAT används på den internationella rymdstationen ISS och är ett multifunktionellt tilläggskort för Raspberry Pi som är utrustat med sensorer och en LED-matrisskärm**



Raspberry Pi levereras med stöd för en speciell typ av tilläggskort som heter *Hardware Attached on Top (HAT)*. Ett HAT kan lägga till allt från mikrofoner och lampor till elektroniska reläer och skärmar till Raspberry Pi, men det är särskilt ett HAT som är mycket speciellt: Sense HAT.

Sense HAT utformades speciellt för rymduppdraget Astro Pi. Astro Pi var ett gemensamt projekt mellan Raspberry Pi Foundation, UK Space Agency och European Space Agency, och det såg till att Raspberry Pi-kort och Sense HAT-kort transporterades upp till den internationella rymdstationen ombord på en Orgital Science Cygnus-lastraket. Dessa Sense HAT-kort – som fick smeknamnen Ed och Izzy av astronauterna – har sedan de oskadda nådde omloppsbanan högt ovanför jorden använts för att köra kod och utföra vetenskapliga experiment som skolbarn från hela Europa har bidragit med.

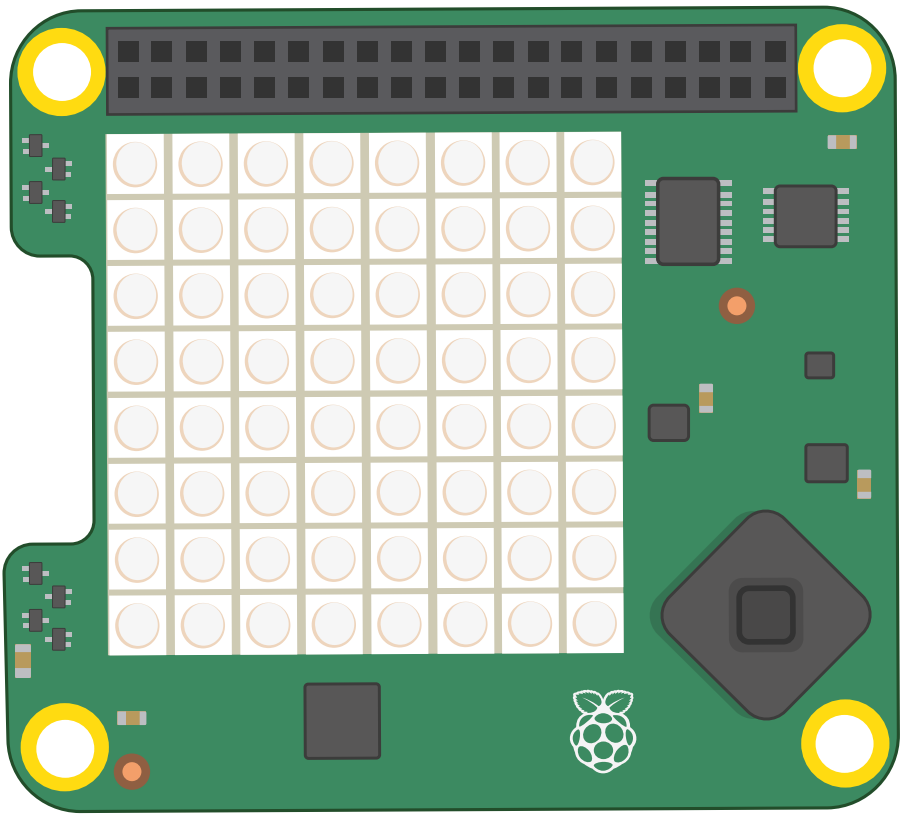
Ed och Izzy är lite för långt borta för att du ska kunna använda dem själv, men du kan hitta samma Sense HAT-maskinvara här på jorden hos alla Raspberry Pi-återförsäljare – och om du inte vill köpa ett Sense HAT just nu, kan du simulera ett i programvara!

### VERKLIG ELLER SIMULERAD

Detta kapitel fungerar bäst ihop med ett riktigt Sense HAT kopplat till en Raspberry Pi:s GPIO, men vem som helst som inte har ett kan hoppa över avsnittet ”Installera Sense HAT” och helt enkelt prova projekten i Sense HAT-emulatorn; de fungerar lika bra!

## Introduktion av Sense HAT

Sense HAT är ett kraftfullt, multifunktionellt tillägg för Raspberry Pi. Sense HAT innehåller en 8 × 8 matris med 64 röda, gröna och blå (RGB) programmerbara lysdioder som kan styras för att producera flera miljoner olika färger, samt en joystick med fem knappar och sex inbyggda sensorer.



**Gyroskopsensor:** Används för att känna av förändringar i vinkel över tid, vilket tekniskt kallas *vinkelhastighet*, genom att hålla reda på riktningen för jordens gravitationsfält – kraften som drar saker ner mot planetens mittpunkt. Enkelt uttryckt kan den gyroskopiska sensorn känna av när du roterar Sense HAT i förhållande till jordens yta och hur snabbt den roterar.

**Accelerometer:** Liknar gyroskopsensorn, men istället för att övervaka en vinkel i förhållande till jordens gravitation mäter den accelerationskraften i flera riktningar. Om avläsningar (data) från de två sensorerna kombineras kan de hjälpa dig att spåra vart en Sense HAT pekar och hur den flyttas.

**Magnetometer:** Mäter styrkan i ett magnetfält och är ytterligare en sensor som kan hjälpa till att spåra Sense HAT-kortets rörelser: genom att mäta jordens naturliga magnetfält kan magnetometern räkna ut riktningen för magnetisk norr. Samma sensor kan även användas för att upptäcka metallföremål och till och med elektriska fält. Alla dessa tre sensorer är inbyggda i ett enda chipp som är märkt ”ACCEL/GYRO/MAG” på Sense HAT-kretskortet.

**Fuktighetsgivare:** Mäter mängden vattenånga i luften, vilket kallas den *relativa fuktigheten*. Relativ luftfuktighet kan variera från 0 %, när det inte finns något vatten alls, till 100 %, när luften är helt mättad. Fuktighetsdata kan användas för att upptäcka när det finns risk för regn!

**Barometrisk trycksensor:** kallas även *barometer* och mäter lufttrycket. De flesta människor känner till barometriskt tryck från väderprognosen, men barometern har en hemlig andra användning: den kan spåra när du klättrar upp- eller nerför en kulle eller ett berg, eftersom luften blir tunnare och trycket sjunker ju längre från jordens havsnivå du kommer.

**Temperaturgivare:** Mäter hur varm eller kall den omgivande miljön är, men den påverkas även av hur varmt eller kallt Sense HAT-kortet är: om du använder ett fodral kan du märka att avläsningarna är högre än du förväntar dig. Sense HAT har ingen separat temperaturgivare. Istället använder den temperaturgivare som är inbyggda i fuktighetssensorn och den barometriska trycksensorn. Ett program kan använda en eller båda av dessa sensorer; det är upp till dig.

## SENSE HAT PÅ RASPBERRY PI 400

Sense HAT är helt kompatibelt med Raspberry Pi 400 och kan sättas in direkt i GPIO:n på baksidan. Om du gör det betyder det dock att lysdioderna kommer att vara vända bort från dig och kortet kommer att vara riktat upp och ner.

För att åtgärda detta behöver du en förlängningskabel eller ett förlängningskort till GPIO:n. Kompatibla förlängningar inkluderar Black HAT Hack3r-sortimentet från Pimoroni; du kan använda Sense HAT med Black HAT Hack3r-kortet eller helt enkelt använda den medföljande 40-stiftsbandkabeln som en förlängning. Kontrollera dock alltid tillverkarens instruktioner för att vara säker på att du ansluter kabeln och Sense HAT på rätt sätt!

## Installera Sense HAT

Om du har ett fysiskt Sense HAT ska du börja med att packa upp det och se till att du har alla delar: du bör ha själva Sense HAT-kortet, fyra metall- eller plastpelare som kallas *distanshållare* och åtta skruvar. Du kan även ha några metallstift i en svart plastremsa, som GPIO-stiften på Raspberry Pi, och om så är fallet ska du trycka den här remsan med stiftsidan upp genom undersidan av Sense HAT tills du hör ett klick.

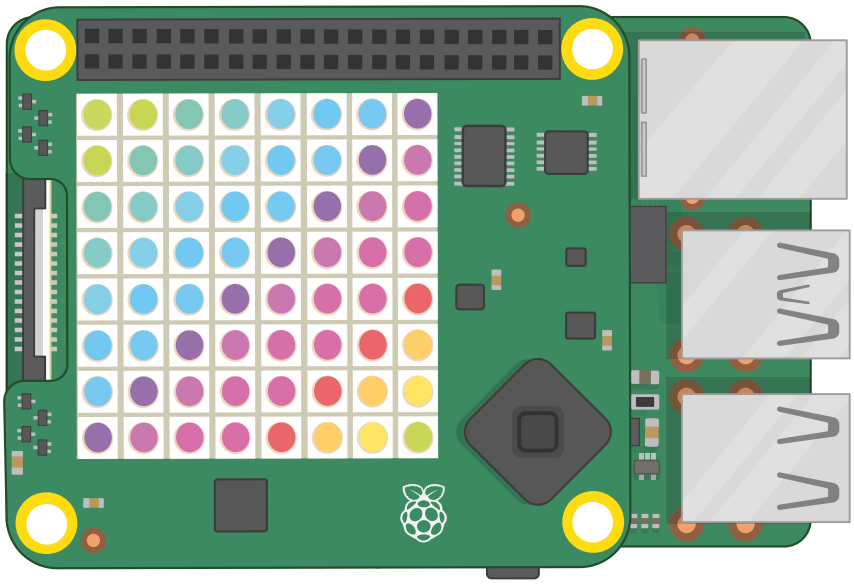
Distanshållarna är utformade för att hindra Sense HAT från att böjas när du använder joysticken. Sense HAT fungerar utan att de installeras, men om du använder dem skyddar du din Sense HAT, Raspberry Pi och GPIO:n från att skadas.

### VARNING!

Hardware attached on top-moduler (HAT) bör bara anslutas till och tas bort från GPIO:n när Raspberry Pi är avstängd och frånkopplad från strömförsörjningen. Var alltid noga med att HAT ligger platt när du installerar det och dubbelkolla att det är i linje med GPIO-stiften innan du trycker ner det.

Installera distanshållarna genom att trycka upp fyra av skruvarna från undersidan av Raspberry Pi genom de fyra monteringshålen i varje hörn, och vrid sedan på distanshållarna på skruvarna. Tryck ner Sense HAT på Raspberry Pi:s GPIO och se till att du lägger det i linje med stiften under och håller det så platt som möjligt. Skruva slutligen de fyra sista skruvarna genom monteringshålen på Sense HAT och in i distanshållarna som du installerade tidigare. Om det är installerat på rätt sätt ska Sense HAT vara plant och jämnt och ska inte böjas eller vicka när du trycker på joysticken.

Anslut strömmen till Raspberry Pi igen. Då ser du att lysdioderna på Sense HAT tänds i ett regnbågsmönster (**Bild 7-1**) och sedan stängs av igen. Sense HAT är nu installerat!

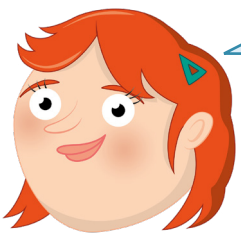


▲ **Bild 7-1:** Ett regnbågsmönster visas när strömmen slås på

Om du vill ta bort Sense HAT igen skruvar du loss de övre skruvarna och lyfter av HAT-kortet – var försiktig så att du inte böjer stiften på GPIO:n eftersom HAT-kortet sitter ganska hårt (du kan behöva bända loss det med en liten skruvmejsel). Ta sedan bort distanshållarna från Raspberry Pi.

## Hej, Sense HAT!

Som med alla programmeringsprojekt finns det ett uppenbart sätt att komma igång med Sense HAT: med ett välkomstmeddelande som rullar över LED-skärmen. Om du använder Sense HAT-emulatorn läser du in den nu genom att klicka på Raspberry Pi OS:s menyikon. Välj sedan kategorin Programmering och klicka på Sense HAT Emulator.



### PROGRAMMERINGSERFARENHET

Detta kapitel förutsätter att du har erfarenhet av Scratch 3 eller Python och Thonnys integrerade utvecklingsmiljö (IDE), beroende på om du arbetar med Scratch- eller Python-kodexemplen – eller båda! Gå till **Kapitel 4, Programmering med Scratch** eller **Kapitel 5, Programmering med Python**, om du inte redan har gjort det, och arbeta igenom projekten i det kapitlet först.

## Hälsningar från Scratch

Läs in Scratch 3 från Raspberry Pi OS-menyn. Klicka på knappen Lägg till ett tillägg längst ned till vänster i Scratch-fönstret. Klicka på Raspberry Pi Sense HAT-tillägget (**Bild 7-2**). Detta läser in blocken du behöver för att styra de olika funktionerna i Sense HAT, inklusive LED-skärmen. När du behöver dem hittar du dem i kategorin Raspberry Pi Sense HAT.

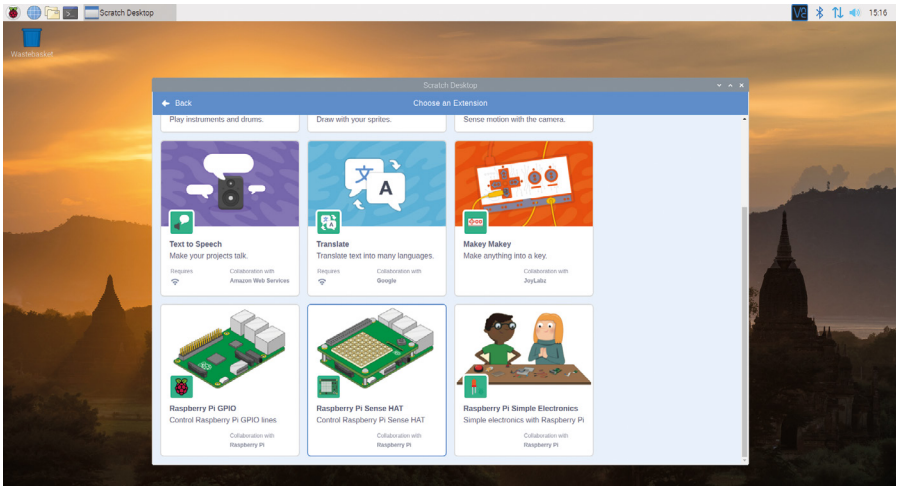



**Bild 7-2:** Lägga till Raspberry Pi Sense HAT-tillägget i Scratch 3

Börja med att dra ett när klickas på-Händelser-block till skriptområdet och dra sedan ett display text Hello!-block direkt inunder det. Redigera texten så att det står display text Hej Världen!.


Klicka på den gröna flaggan på scenområdet och titta på Sense HAT eller Sense HAT-emulatorn: meddelandet rullar långsamt över Sense HAT-kortets LED-matris och lyser upp LED-pixlarna så att bokstäverna bildas i tur och ordning (**Bild 7-3**, på nästa sida). Grattis: du har lyckats med programmet!

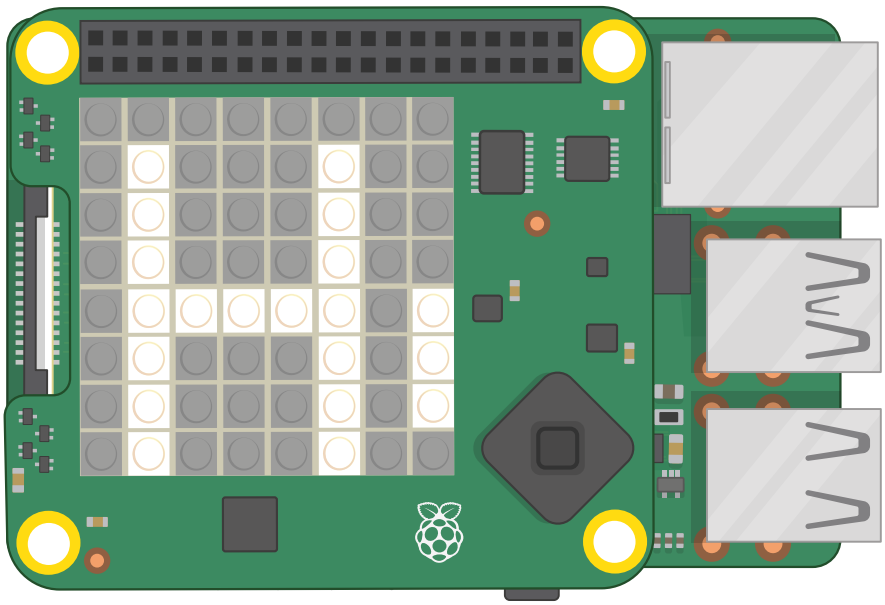


**Bild 7-3:** Meddelandet rullar över LED-matrisen

Nu när du kan rulla ett enkelt meddelande är det dags att ta en titt på hur det meddelandet visas. Du kan ändra meddelandet som ska visas men även ändra rotationen – i vilken riktning meddelandet visas på Sense HAT. Dra ett `set rotation to 0 degrees`-block från blockpaletten och infoga det under `när klickas på` och över `display text Hej Världen!` och klicka sedan på nedåtpilen bredvid 0 och ändra det till 90. Klicka på den gröna flaggan så ser du samma meddelande som tidigare, men istället för att rulla från vänster till höger kommer det att rulla nedifrån och upp (**Bild 7-4**) – du måste vrida på huvudet, eller Sense HAT, för att läsa det!

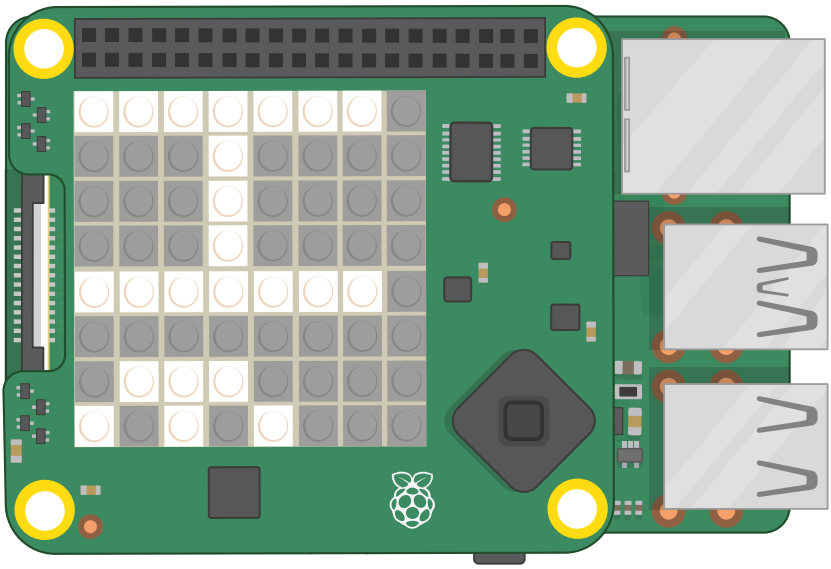


**Bild 7-4:** Den här gången rullar meddelandet vertikalt

Ändra nu tillbaka rotationen till 0 och dra sedan ett set colour-block mellan set rotation to 0 degrees och display text Hej Världen! . Klicka på färgen i slutet av blocket för att ta fram Scratch-färgväljaren och leta upp en fin gul färg. Klicka sedan på den gröna flaggan för att se hur programmet har ändrats (**Bild 7-5**).

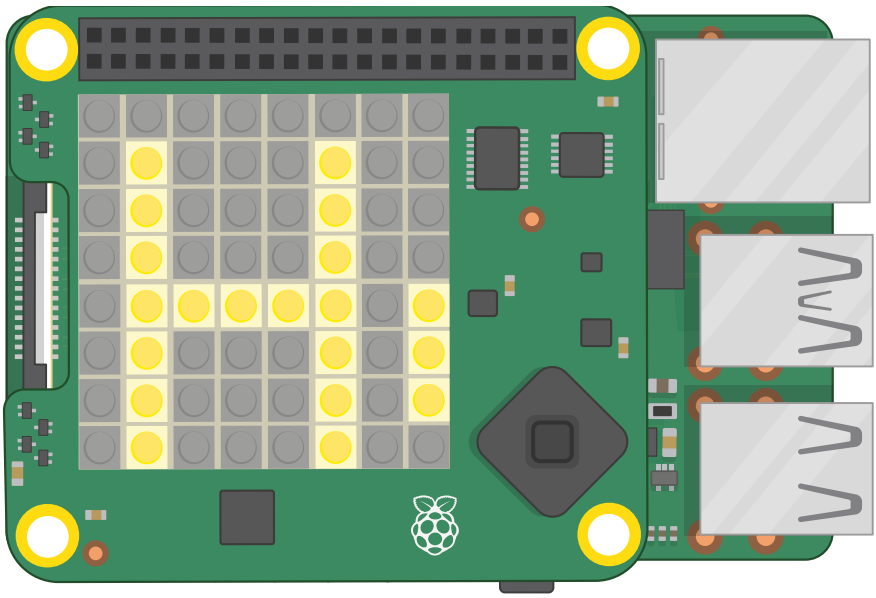


**Bild 7-5:** Ändra färg på texten

Slutligen drar du ett set background-block mellan set colour to gul och display text Hej Världen! och klickar på färgen för att få fram färgväljaren igen. Den här gången påverkar inte färgen lysdioderna som utgör meddelandet utan lysdioderna som inte gör det – den så kallade bakgrunden. Hitta en fin blå färg och klicka sedan på den gröna flaggan igen: den här gången kommer meddelandet att vara gult på en blå bakgrund. Försök att ändra dessa färger för att hitta din favoritkombination – alla färger fungerar inte bra tillsammans!

Förutom att du kan rulla hela meddelanden, kan du visa enskilda bokstäver. Dra display text-blocket från skriptområdet för att radera det och dra sedan ett display character A-block till skriptområdet istället. Klicka på den gröna flaggan så ser du skillnaden: detta block visar bara en bokstav åt gången, och bokstaven blir kvar på Sense HAT, utan att rulla eller försvinna, tills du ger den andra instruktioner. Samma färgkontrollerande block gäller för detta block som display text-blocket: försök att ändra bokstavens färg till röd (**Bild 7-6**).

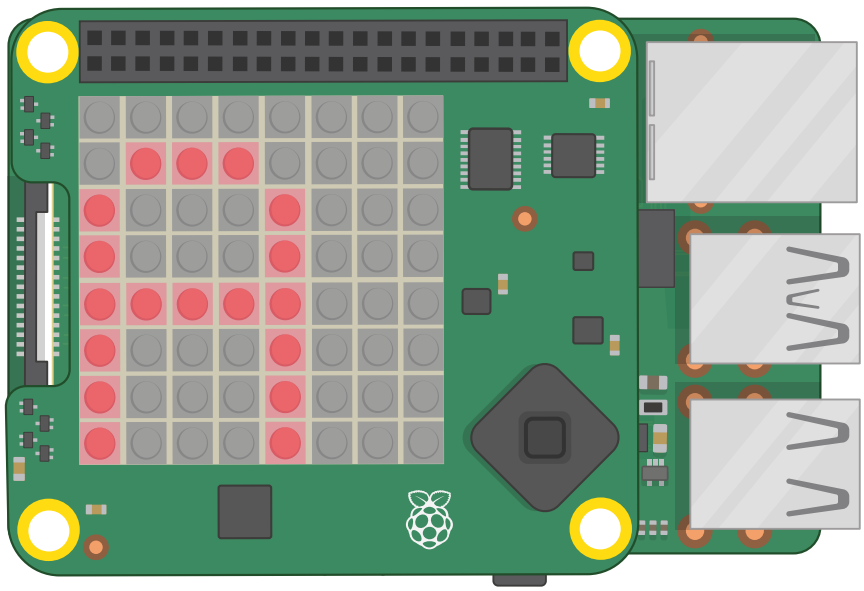
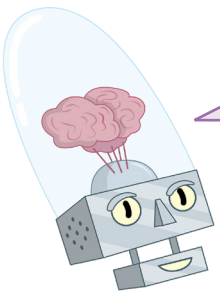


**Bild 7-6:** Visar en enskild bokstav



## UTMANING: UPPREPA MEDDELANDET

Kan du använda din kunskap om loopar för att få ett rullningsmeddelande att upprepa sig? Kan du skapa ett program som stavar ett ord bokstav för bokstav i olika färger?

## Hälsningar från Python

Läs in Thonny genom att klicka på hallonmenyikonen, välja Programmering och klicka på Thonny. Om du använder Sense HAT-emulatorn och den täcks av Thonny-fönstret kan du klicka och hålla ner musknappen på något av fönstrets titelfält – högst upp i blått – och dra det för att flytta runt det på skrivbordet tills du kan se båda fönstren.

### RADÄNDRING I PYTHON

Python-kod som är skriven för ett fysiskt Sense HAT körs på Sense HAT-emulatorn, och vice versa, med endast en ändring. Om du använder Sense HAT-emulatorn med Python måste du ändra raden från `sense_hat import SenseHat` i alla program från detta kapitel till `from sense_emu import SenseHat` istället. Om du sedan vill köra dem på ett fysiskt Sense HAT igen byter du bara tillbaka raden!

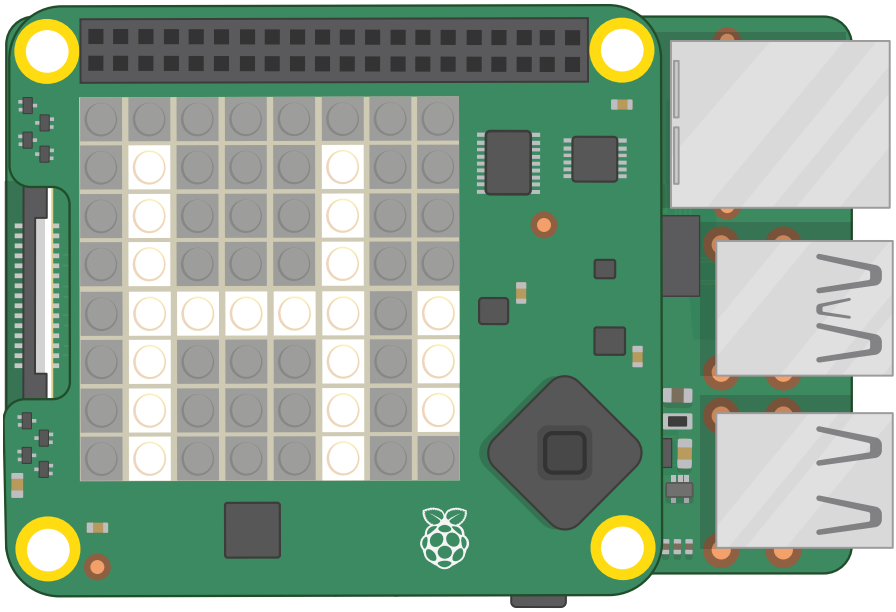
För att kunna använda Sense HAT, eller Sense HAT-emulatorn, i ett Python-program måste du importera Sense HAT-biblioteket. Skriv följande i skriptområdet, och kom ihåg att använda **`sense_emu`** (istället för **`sense_hat`**) om du använder Sense HAT-emulatorn:

```
from sense_hat import SenseHat
sense = SenseHat()
```

Sense HAT-biblioteket har en enkel funktion för att ta emot ett meddelande, formatera det så att det kan visas på LED-skärmen och rulla det smidigt. Skriv följande:

```
sense.show_message("Hej, världen!")
```

Spara programmet som **Hej Sense HAT** och klicka sedan på knappen Run. Du ser ett meddelande rulla långsamt över Sense HAT-kortets LED-matris och lysa upp LED-pixlarna så att bokstäverna bildas i tur och ordning (**Bild 7-7**, på nästa sida). Grattis: du har lyckats med programmet!



▲ **Bild 7-7:** Ett meddelande rullar över LED-matrisen

**`show_message()`**-funktionen går dock att använda på många fler sätt än så. Gå tillbaka till programmet och redigera den sista raden så att det står:

```
sense.show_message("Hej, världen!", text_colour=(255, 255, 0),
back_colour=(0, 0, 255), scroll_speed=(0.05))
```

Dessa extra instruktioner, separerade med kommatecken, kallas *parametrar*, och de styr olika aspekter av **`show_message()`**-funktionen. Den enklaste är **`scroll_speed=()`**, som ändrar hur snabbt meddelandet rullar över skärmen. Ett värde på 0.05 här rullar med ungefär två gånger den vanliga hastigheten. Ett högre tal här resulterar i en lägre hastighet.

Parametrarna **`text_colour=()`** och **`back_colour=()`** – stavat på brittisk engelska, till skillnad från de flesta Python-instruktioner – ställer in färgen på texten respektive bakgrunden. De accepterar dock inte namn på färger så du måste ange den färg du vill ha som en trio av tal. Det första talet representerar mängden rött i färgen, från 0 för inget rött alls till 255 för så mycket rött som möjligt; det andra talet är mängden grönt i färgen; och det tredje talet mängden blått. Tillsammans kallas dessa *RGB* – för rött, grönt och blått.

Klicka på Run-ikonen och titta på Sense HAT: den här gången kommer meddelandet att rulla betydligt snabbare och vara i gult på en blå bakgrund (**Bild 7-8**). Försök att ändra parametrarna för att hitta en kombination av hastighet och färg som fungerar för dig.

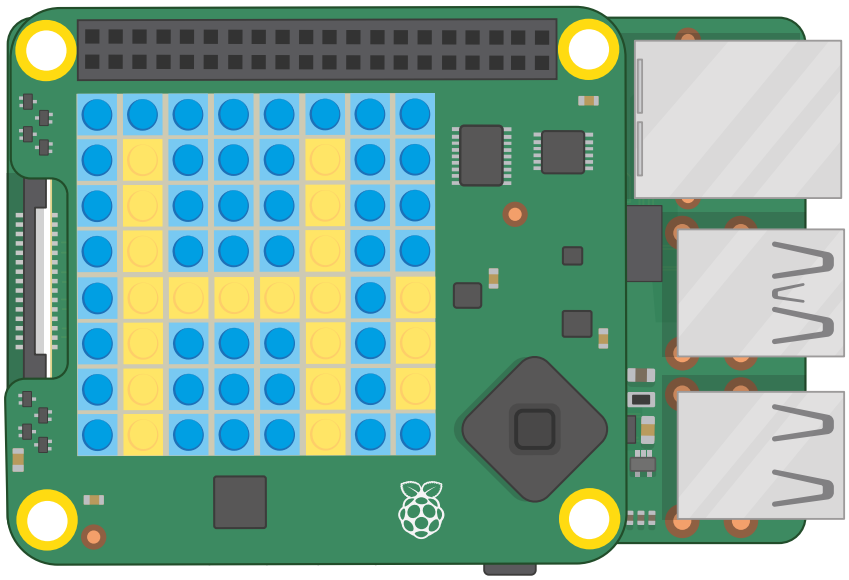


**Bild 7-8:** Ändrar färg på meddelande och bakgrund

Om du vill använda välkända namn istället för RGB-värden för att ställa in färgerna måste du skapa variabler. Ovanför `sense.show_message()`-raden lägger du till följande:

```
yellow = (255, 255, 0)
blue = (0, 0, 255)
```

Gå tillbaka till `sense.show_message()`-raden och redigera den så att det står:

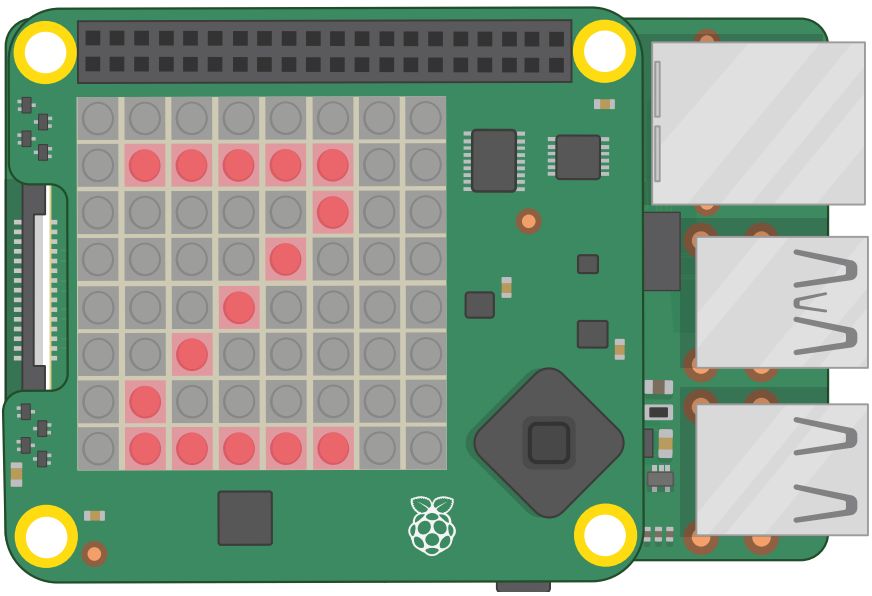
```
sense.show_message("Hej, världen!", text_colour=(yellow), back_
colour=(blue), scroll_speed=(0.05))
```

Klicka på ikonen Run igen så ser du att inget har förändrats: meddelandet är fortfarande i gult på en blå bakgrund. Den här gången har du dock använt variabelnamnen för att göra koden mer läsbar: istället för en rad med tal förklarar koden vilken färg den ställer in. Du kan specificera så många färger du vill: försök att lägga till en variabel som kallas ”röd” med värdena 255, 0 och 0; en variabel som kallas ”vit” med värdena 255, 255, 255; och en variabel som kallas ”svart” med värdena 0, 0 och 0.

Du kan både rulla hela meddelanden och visa enskilda bokstäver. Radera hela `sense.show_message()`-raden och skriv istället följande:

```
sense.show_letter("Z")
```

Klicka på Run så visas bokstaven ”Z” på Sense HAT-kortets skärm. Den här gången blir den kvar där: enskilda bokstäver, till skillnad från meddelanden, rullar inte automatiskt. Du kan även styra `sense.show_letter()` med samma färgparametrar som `sense.show_message()`: försök att ändra bokstavens färg till röd (**Bild 7-9**).



▲ **Bild 7-9:** Visar en enskild bokstav

**UTMANING:** UPPREPA MEDDELANDET

Kan du använda din kunskap om loopar för att få ett rullningsmeddelande att upprepa sig? Kan du skapa ett program som stavar ett ord bokstav för bokstav i olika färger? Hur snabbt kan du få ett meddelande att rulla?

## Nästa steg: Rita med ljus

Sense HAT-kortets LED-skärm är inte bara till för meddelanden: du kan även visa bilder. Varje LED kan behandlas som en enda pixel – förkortning för *picture element* – i en bild du väljer, så att du kan liva upp programmen med bilder och till och med animationer.

Om du vill skapa bilder måste du dock kunna ändra enskilda lysdioder. För att göra det måste du förstå hur Sense HAT-kortets LED-matris är utformad så att du kan skriva ett program som slår på eller stänger av rätt lysdioder.

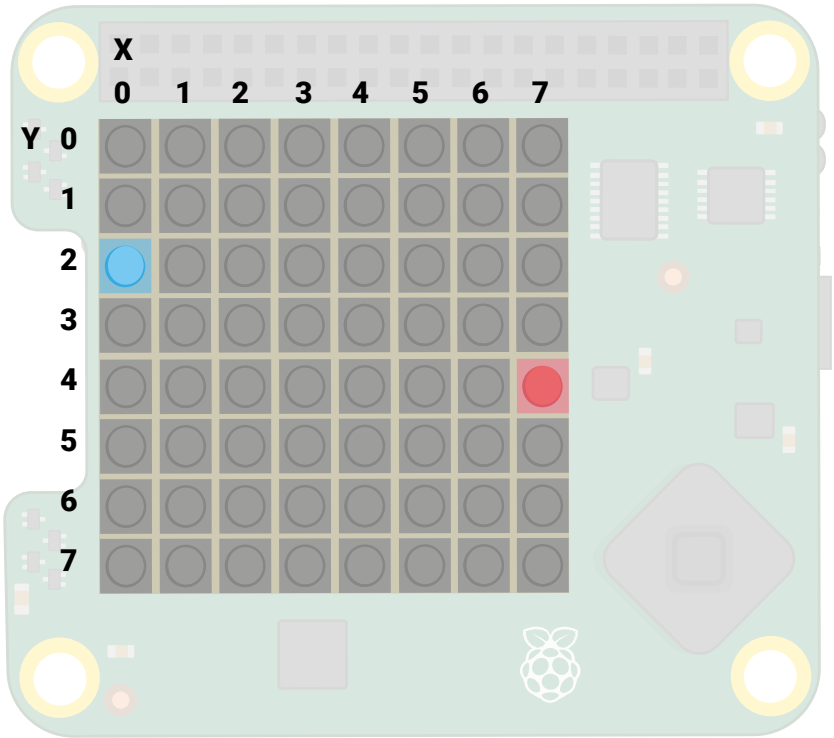



**Bild 7-10:** LED-matrisens koordinatsystem

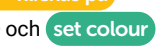
Det finns åtta lysdioder i varje rad på skärmen och åtta i varje kolumn (**Bild 7-10**). När du räknar lysdioderna bör du dock – som i de flesta programmeringsspråk – börja vid 0 och sluta vid 7. Den första lysdioden är i det övre vänstra hörnet och den sista är längst ner till höger. Med hjälp av siffrorna från raderna och kolumnerna kan du leta upp *koordinaterna* för vilken lysdiod som helst på matrisen. Den blå lysdioden i den avbildade matrisen finns vid koordinaterna 0, 2; den röda lysdioden finns vid koordinaterna 7, 4. X-axeln, vågrät på matrisen, kommer först och följs av Y-axeln, lodrät på matrisen.

När du planerar vilka bilder du vill rita på Sense HAT kan det hjälpa att först rita dem för hand, på rutat papper, eller så kan du planera i ett kalkylblad som LibreOffice Calc.

## Bilder i Scratch

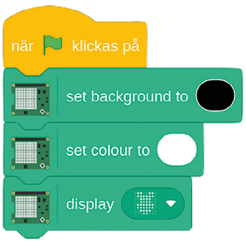
Starta ett nytt projekt i Scratch och spara ditt befintliga projekt om du vill behålla det. Om du har arbetat igenom projekten i det här kapitlet kommer Scratch 3 att hålla Raspberry Pi Sense HAT-tillägget inläst; om du har stängt och öppnat Scratch 3 sedan ditt senaste projekt behöver du ladda tillägget med knappen Lägg till ett tillägg. Dra ett 

-Händelser-block till kodområdet och dra sedan blocken 

och 

direkt inunder det. Redigera båda för att ställa in bakgrundsfärgen till svart och färgen till vit:

ställ in svart genom att skjuta reglagen Brightness och Saturation till 0; ställ in vit genom att skjuta Brightness till 100 och Saturation till 0. Du måste göra detta i början av varje Sense HAT-program, för annars använder Scratch helt enkelt de färger du valde sist – även om du valde dem i ett annat program. Slutligen drar du ett **display hallon**-block till längst ner i programmet.




Klicka på den gröna flaggan: du ser Sense HAT-kortets lysdioder lysa upp ett hallon (**Bild 7-11**).

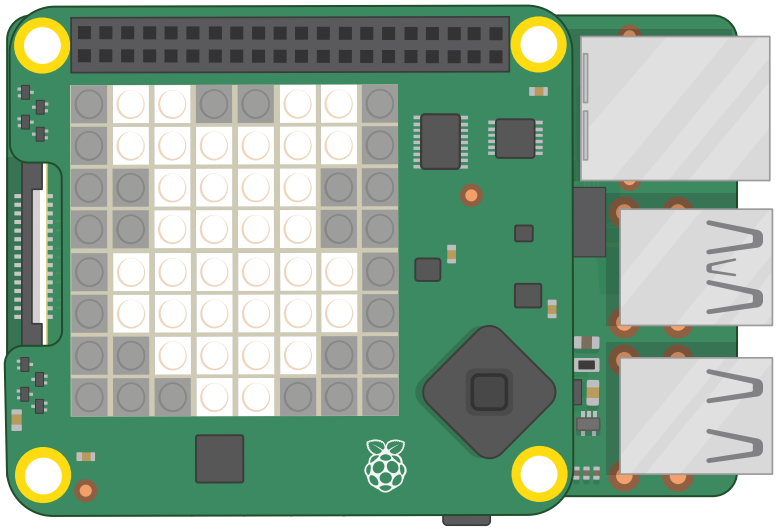
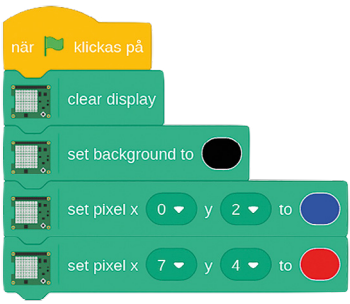


**Bild 7-11:** Titta inte direkt på lysdioderna när de är bländande vita

Du är inte heller begränsad till den förinställda hallonformen. Klicka på nedåtpilen bredvid hallonet till aktiverat ritläge: du kan klicka på vilken lysdioid som helst i mönstret för att slå på eller av den, medan de två knapparna längst ner ställer in alla lysdioder till på eller av. Försök nu att rita ett eget mönster och klicka sedan på den gröna pilen för att se det på Sense HAT. Försök även ändra färgen och bakgrundsfärgen med hjälp av blocken ovan.

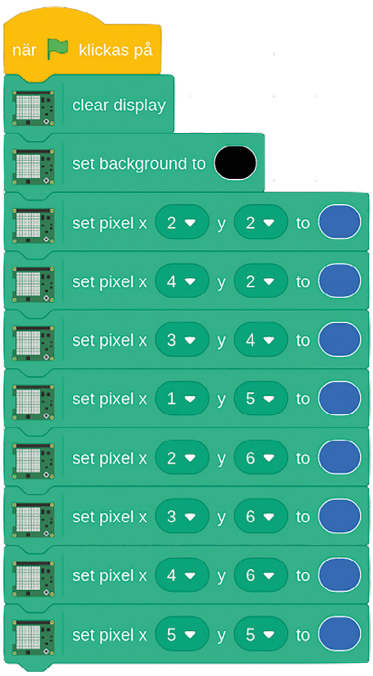
När du är klar drar du de tre blocken till blockpaletten för att radera dem och placerar ett **clear display**-block under **när klickas på**. Klicka sedan på den gröna flaggan så släcks alla lysdioder.

För att skapa en bild måste du kunna styra enskilda pixlar och ge dem olika färger. Du kan göra detta genom att länka redigerade **display hallon**-block med **set colour** -block eller så kan du hantera varje pixel individuellt. Om du vill skapa en egen version av LED-matrisexemplet som visas i början av detta avsnitt, med två specifikt utvalda lysdioder som lyser i rött och blått, ska du lämna **clear display**-blocket högst upp i programmet och dra in ett **set background**-block under det. Ändra **set background**-blocket till svart och dra sedan in två **set pixel x 0 y 0**-block under det. Slutligen redigerar du dessa block enligt följande:

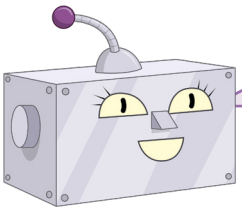



Klicka på den gröna flaggan så ser du att lysdioderna tänds för att matcha matrisbilden (**Bild 7-10**) på sidan 165. Grattis: du kan styra enskilda lysdioder!

Redigera den befintligaa uppsättningen pixelblocken som följer och dra fler längst ner tills du har följande program:



Innan du klickar på den gröna flaggan kan du försöka gissa vilken bild som kommer att visas baserat på de LED-matriskoordinater du har använt. Kör sedan programmet och se om du har rätt!



### UTMANING: NYA DESIGNER

Kan du designa fler bilder? Försök att få tag på lite rutpapper och använd det för att först planera bilden för hand. Kan du rita en bild och få färgerna att ändras?

## Bilder i Python

Starta ett nytt program i Thonny och spara det som Sense HAT Bild. Skriv sedan följande i skriptområdet – kom ihåg att använda **`sense_emu`** (istället för **`sense_hat`**) om du använder Sense HAT-emulatorn:

```
from sense_hat import SenseHat
sense = SenseHat()
```

Kom ihåg att du behöver båda dessa rader i programmet för att kunna använda Sense HAT. Skriv sedan:

```
sense.clear(255, 255, 255)
```

Klicka på ikonen Run utan att titta direkt på Sense HAT-kortets lysdioder: du ska se att alla blir bländande vita (**Bild 7-12**) – vilket är anledningen till att du inte bör titta direkt på dem när du kör programmet!

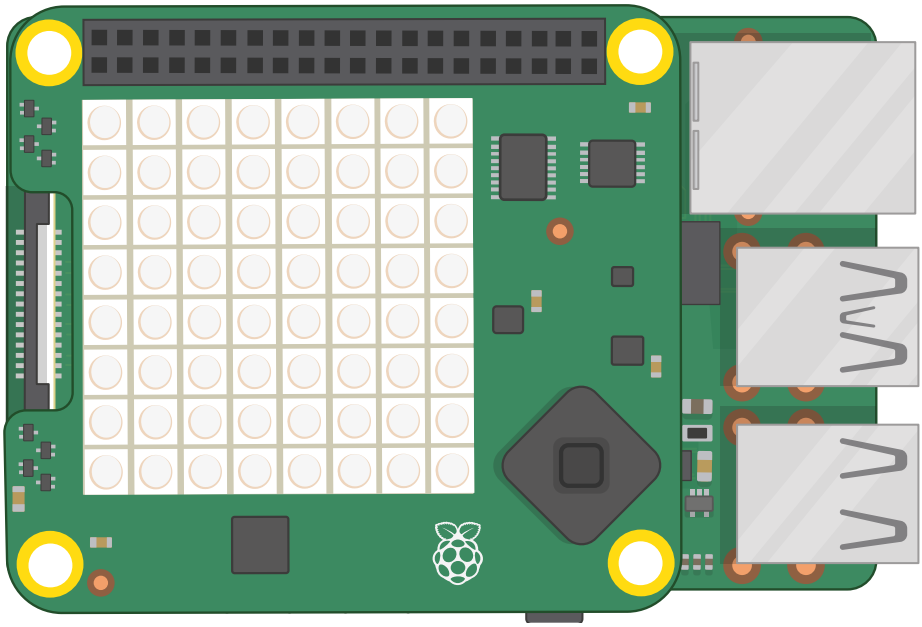


**Bild 7-12:** Titta inte direkt på matrisen när den lyser i bländande vitt

`sense.clear()` är utformat för att rensa lysdioderna från tidigare programmering, men accepterar RGB-färgparametrar – vilket innebär att du kan ändra skärmen till vilken färg du vill. Försök ändra raden till:

```
sense.clear(0, 255, 0)
```

Klicka på Run så blir Sense HAT ljusgrön (**Bild 7-13**, på nästa sida). Experimentera med olika färger eller lägg till de variabler för färgnamn som du skapade för Hej, världen-programmet för att det ska bli mer lättläst.

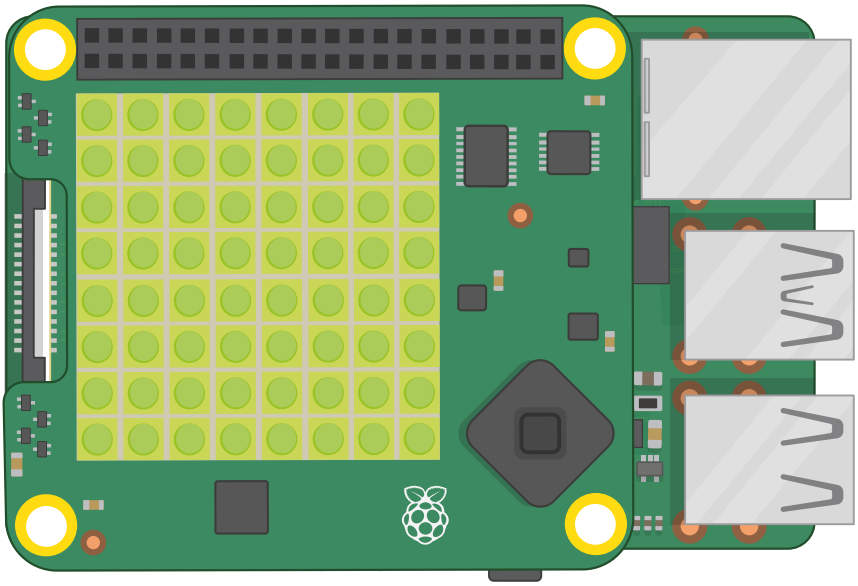


**Bild 7-13:** LED-matrisen lyser i ljusgrönt

För att rensa lysdioderna måste du använda RGB-värdena för svart: 0 röd, 0 blå och 0 grön. Det finns dock ett enklare sätt. Redigera raden i programmet till:

```
sense.clear()
```

Sense HAT kommer att bli mörkt; **`sense.clear()`**-funktionen fungerar som så att om det inte finns något mellan parenteserna ska alla lysdioder ställas in till svart – dvs. stängas av (**Bild 7-14**). När du behöver rensa lysdioderna i programmen fullständigt är det den här funktionen du ska använda.

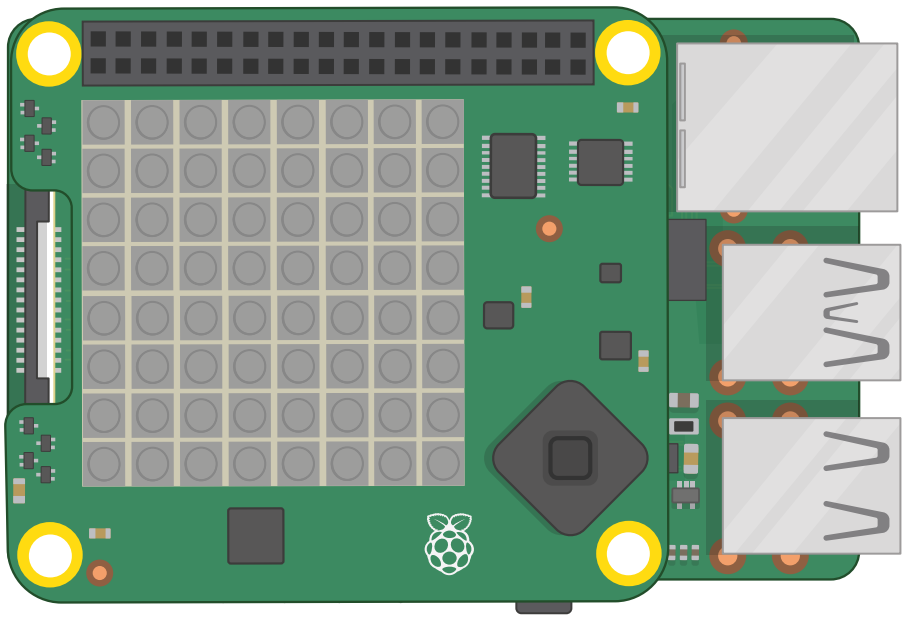


**Bild 7-14:** Använd `sense.clear`-funktionen för att stänga av alla lysdioder

Om du vill skapa en egen version av LED-matrisen som visades tidigare i kapitlet, med två specifikt utvalda lysdioder som lyser i rött och blått, ska du lägga till följande rader i programmet efter **`sense.clear()`** :

```
sense.set_pixel(0, 2, (0, 0, 255))
sense.set_pixel(7, 4, (255, 0, 0))
```

De första två siffrorna är pixelns position på matrisen, X-axel (vågrät) följd av Y-axel (lodrät). Det andra paret, i sin egen uppsättning parenteser, är RGB-värdena för pixelfärg. Klicka på knappen Run så ser du resultatet: två lysdioder på Sense HAT kommer att lysa upp, precis som i **Bild 7-10** på sida 165.

Radera dessa två rader och skriv följande:

```
sense.set_pixel(2, 2, (0, 0, 255))
sense.set_pixel(4, 2, (0, 0, 255))
sense.set_pixel(3, 4, (100, 0, 0))
sense.set_pixel(1, 5, (255, 0, 0))
sense.set_pixel(2, 6, (255, 0, 0))
sense.set_pixel(3, 6, (255, 0, 0))
sense.set_pixel(4, 6, (255, 0, 0))
sense.set_pixel(5, 5, (255, 0, 0))
```

Innan du klickar på Run kan du titta på koordinaterna och jämföra dem med matrisen: kan du gissa vilken bild dessa instruktioner kommer att rita? Klicka på Run för att ta reda på om du har rätt!

Det tar dock tid att rita en detaljerad bild med enskilda **`set_pixel()`**-funktioner. För att få det att gå snabbare kan du ändra flera pixlar samtidigt. Radera alla **`set_pixel()`**-rader och skriv följande:

```
g = (0, 255, 0)
b = (0, 0, 0)

creeper_pixels = [
    g, g, g, g, g, g, g, g,
    g, g, g, g, g, g, g, g,
    g, b, b, g, g, b, b, g,
    g, b, b, g, g, b, b, g,
    g, g, g, b, b, g, g, g,
    g, g, b, b, b, b, g, g,
    g, g, b, b, b, b, g, g,
    g, g, b, g, g, b, g, g
]

sense.set_pixels(creeper_pixels)
```

Det är mycket där, men börja med att klicka på Run för att se om du känner igen en viss liten creeper. De två första raderna skapar två variabler som innehåller färger: grön och svart. För att koden för bilden ska vara enklare att skriva och läsa, är variablerna enstaka bokstäver: ”g” för grönt och ”b” för svart.

Nästa block med kod skapar en variabel som innehåller färgvärden för alla 64 pixlar på LED-matrisen, separerade med kommatecken och inneslutna mellan hakparenteser. Istället för siffror använder det dock de färgvariabler som du skapade tidigare: titta noga, och kom ihåg att ”g” är för grönt och ”b” är för svart, så kan du redan se bilden som kommer att visas (**Bild 7-15**).

Slutligen använder **`sense.set_pixels(creeper_pixels)`** den variabeln för att med hjälp av **`sense.set_pixels()`**-funktionen rita på hela matrisen på en gång. Mycket lättare än att försöka rita pixel-för-pixel!

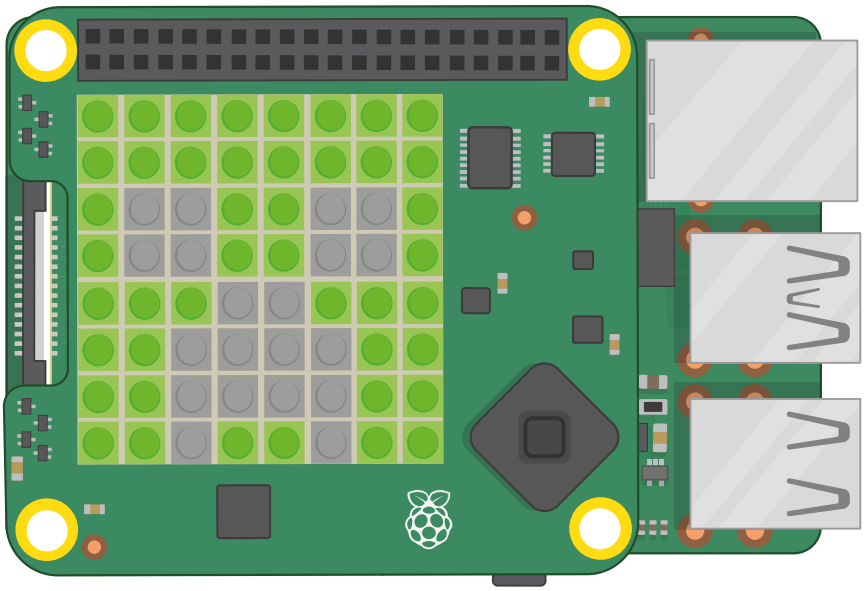


**Bild 7-15:** Visar en bild på matrisen

Du kan också rotera och vända bilder, antingen som ett sätt att visa bilder med rätt sida upp när du vänder på Sense HAT eller som ett sätt att skapa enkla animationer från en enda asymmetrisk bild.

Börja med att redigera **`creeper_pixels`**-variabeln för att stänga hans vänstra öga genom att ersätta de fyra ”b”-pixlarna, först de första två på tredje raden och sedan de första två på fjärde raden, med ”g”:

```
creeper_pixels = [
    g, g, g, g, g, g, g, g,
    g, g, g, g, g, g, g, g,
    g, g, g, g, g, b, b, g,
    g, g, g, g, g, b, b, g,
    g, g, g, b, b, g, g, g,
    g, g, b, b, b, b, g, g,
    g, g, b, b, b, b, g, g,
    g, g, b, g, g, b, g, g
]
```

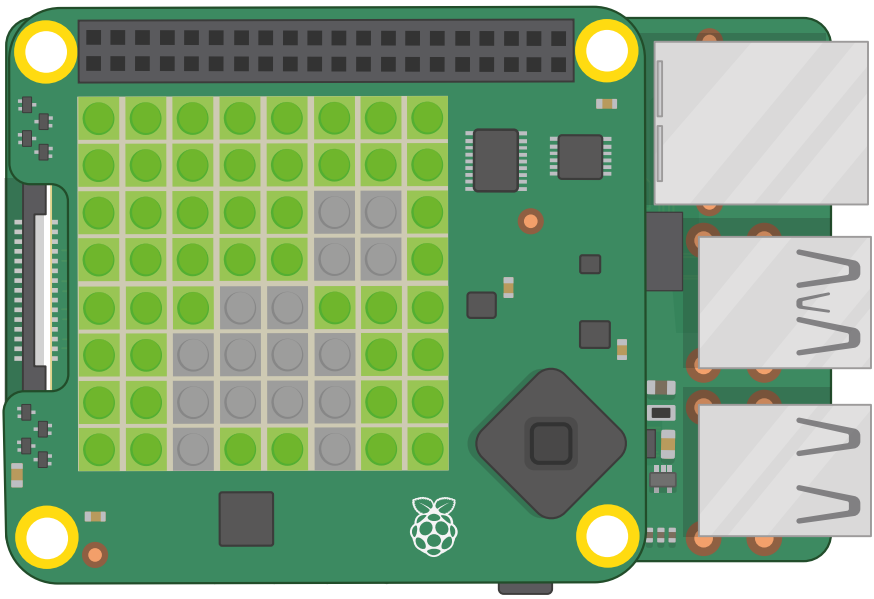
Klicka på Run så ser du att det vänstra ögat på creepern stängs (**Bild 7-16**, på nästa sida). För att skapa en animation går du högst upp i programmet och lägger till raden:

```
from time import sleep
```

Gå sedan längst ner och skriv:

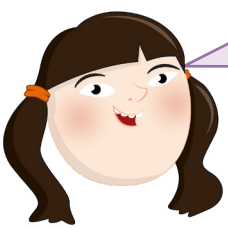
```
while True:
    sleep(1)
    sense.flip_h()
```

Klicka på Run och titta på creepern när den stänger och öppnar ögonen, ett i taget!



▲ **Bild 7-16:** Visar en enkel animation med två bilder

**`flip_h()`**-funktionen vänder en bild på den horisontella axeln, vågrätt; om du vill vända en bild på den vertikala axeln ersätter du istället **`sense.flip_h()`** med **`sense.flip_v()`**. Du kan även rotera en bild 0, 90, 180 eller 270 grader med hjälp av **`sense.set_rotation(90)`** genom att ändra numret beroende på hur många grader du vill rotera bilden. Försök använda det här för att få creepern att snurra runt istället för att blinka!



## UTMANING: NYA DESIGNER

Kan du designa fler bilder och animationer? Försök att få tag på lite rutpapper och använd det för att först planera din bild för hand för att göra det lättare att skriva variabeln. Kan du rita en bild och få färgerna att ändras? Tips: du kan ändra variablerna när du redan har använt dem en gång.

## Känna av världen runt dig

Sense HAT-kortets verkliga styrka ligger i de olika sensorer som det har. Dessa låter dig läsa av allt från temperatur till acceleration och använda detta på olika sätt i programmen.

### EMULERA SENSORERNA

Om du använder Sense HAT-emulatorn måste du aktivera simulering av tröghets- och miljösensorer: i emulatorn klickar du på Edit och sedan på Preferences och markerar dem. I samma meny väljer du ”180°..360°|0°..180°” under ”Orientation Scale” för att se till att siffrorna i emulatorn matchar siffrorna som rapporterats av Scratch och Python och klickar sedan på stängningsknappen.

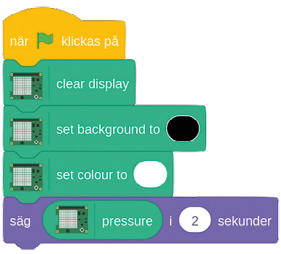
## Miljöavkänning

Den barometriska trycksensorn, fuktighetsgivaren och temperaturgivaren är alla miljösensorer; de mäter omgivningen runt Sense HAT.

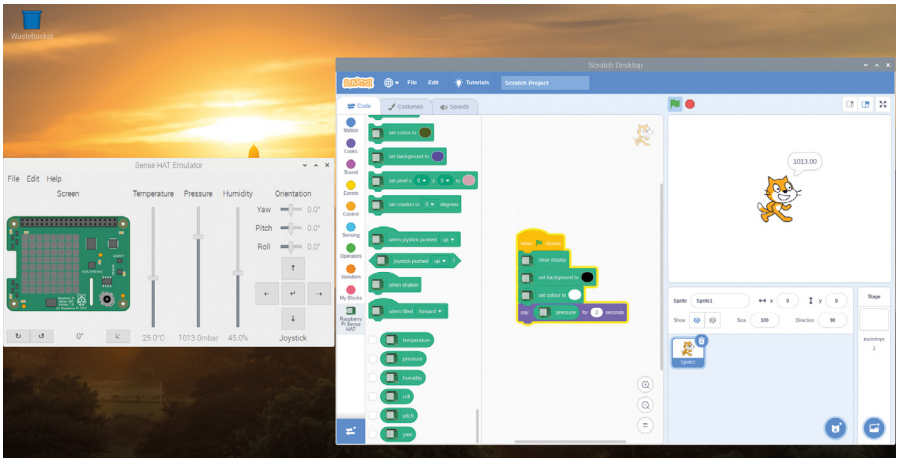
## Miljöavkänning i Scratch

Starta ett nytt program i Scratch (spara ditt gamla om du vill) och lägg till Raspberry Pi Sense HAT-tillägget om det inte redan har lästs in. Dra ett `när klickas på`-Händelser-block till kodområdet och dra sedan in ett `clear display`-block under det samt ett `set background to svart`-block inunder det. Lägg sedan till ett `set colour to vit`-block – använd reglagen Brightness och Saturation för att välja rätt färg. Det är alltid en bra idé att göra detta när du startar programmen, eftersom det kommer att se till att Sense HAT inte visar något från ett gammalt program samtidigt som du säkert vet vilka färger du använder.

Dra ett `säg Hej! i 2 sekunder`-Utseende-block direkt inunder de befintliga blocken. För att ta en avläsning från trycksensorn letar du upp `pressure`-blocket i kategorin Raspberry Pi Sense HAT och drar det över ordet ”Hej!” i `säg Hej! i 2 sekunder`-blocket.




Klicka på den gröna flaggan så kommer Scratch-katten att meddela den aktuella avläsningen från trycksensorn i *millibar*. Meddelandet försvinner efter två sekunder. Försök att blåsa på Sense HAT (eller flytta Pressure-reglaget upp i emulatorn) och klicka på den gröna flaggan för att köra programmet igen; du bör se en högre avläsning den här gången (**Bild 7-17**, på nästa sida).

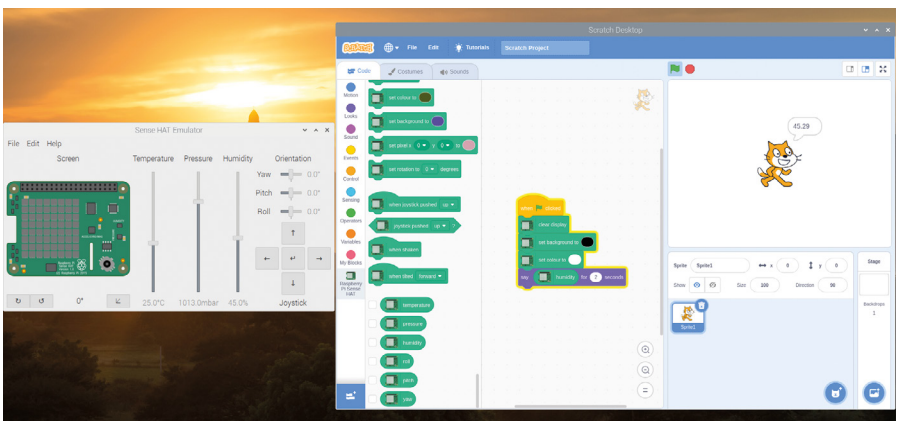


▲ **Bild 7-17:** Visa tryckgivarens avläsning

## ÄNDRA VÄRDEN

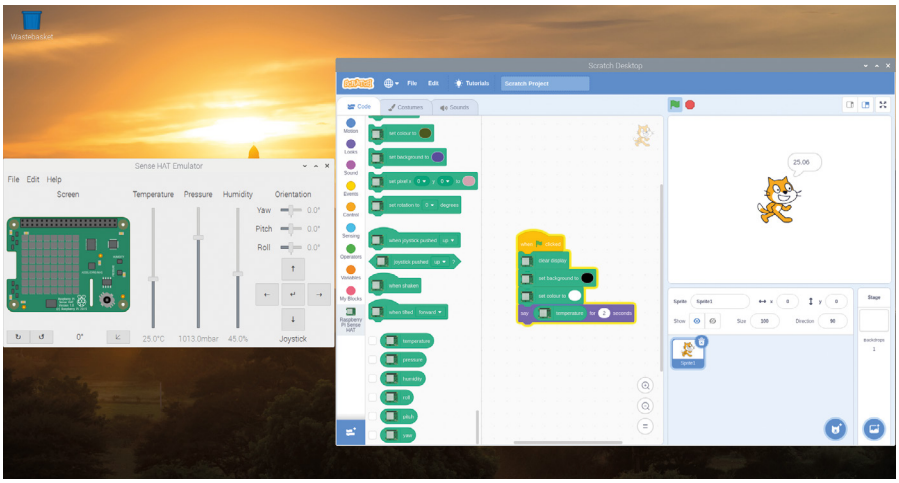
Om du använder Sense HAT-emulatorn kan du ändra värdena som rapporterats av var och en av de emulerade sensorerna med hjälp av reglagen och knapparna. Försök att skjuta trycksensorns inställning ner mot botten och klicka sedan på den gröna flaggan igen.

För att byta till fuktighetsgivaren tar du bort **pressure**-blocket och ersätter det med **humidity**. Kör programmet igen så ser du aktuell relativ luftfuktighet i rummet. Du kan försöka köra den igen medan du blåser på Sense HAT (eller flyttar emulatorns Humidity-reglage uppåt) för att ändra avläsningen (**Bild 7-18**) – din andedräkt är förvånansvärt fuktig!

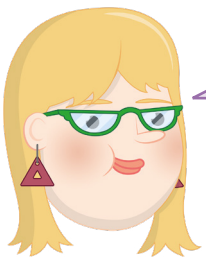


▲ **Bild 7-18:** Visar avläsningen från fuktighetsgivaren

Med temperaturgivaren är det här så enkelt som att ta bort humidity-blocket och ersätta det med temperature och sedan köra programmet igen. Du ser en temperatur i grader Celsius (**Bild 7-19**). Det är dock inte säkert att det är den exakta temperaturen i rummet: Raspberry Pi genererar värme under hela den tid den körs och det värmer även upp Sense HAT och dess sensorer.




▲ **Bild 7-19:** Visar temperaturgivarens avläsning



## UTMANING: RULLNING OCH LOOP

Kan du ändra programmet så att det läser av var och en av sensorerna i tur och ordning och sedan rullar dem över LED-matrisen istället för att skriva ut dem till scenområdet? Kan du göra så att programmet går i en loop, så att det ständigt skriver ut de nuvarande miljöförhållandena?

## Miljöavkänning i Python

Skapa ett nytt program i Thonny och spara det som **Sense HAT-sensorer** för att börja ta avläsningar från sensorerna. Skriv följande i skriptområdet, som du alltid ska göra när du använder Sense HAT – och kom ihåg att använda `sense_emu` om du använder emulatorn:

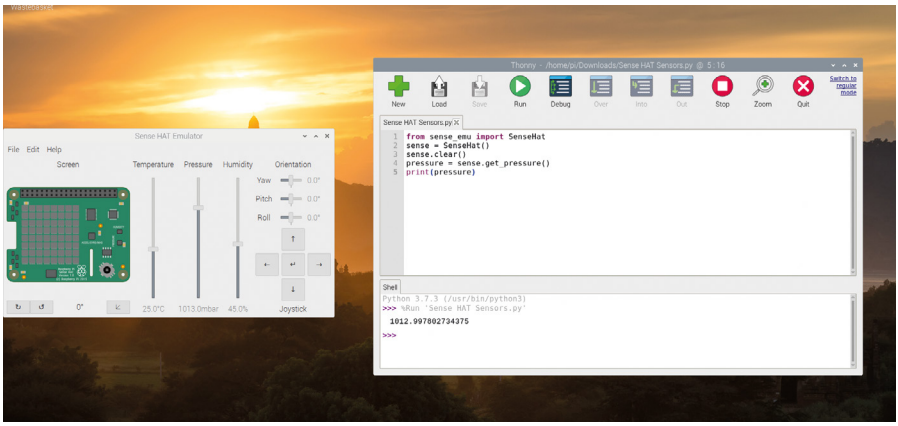
```
from sense_hat import SenseHat
sense = SenseHat()
sense.clear()
```

Det är alltid en bra idé att inkludera `sense.clear()` när du startar programmen, ifall Sense HAT-kortets skärm fortfarande visar något från det senaste programmet som kördes.

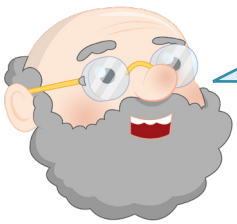
För att ta en avläsning från trycksensorn skriver du:

```
pressure = sense.get_pressure()
print(pressure)
```

Klicka på Run så ser du att ett nummer skrivs till Python-skalet längst ner i Thonny-fönstret. Det här är den lufttrycksavläsning som påvisades av den barometriska trycksensorn i *millibar* (**Bild 7-20**). Försök att blåsa på Sense HAT (eller flytta Pressure-reglaget upp i emulatorn) medan du klickar på Run-knappen igen; numret bör vara högre den här gången.




▲ **Bild 7-20:** Skriver ut en tryckavläsning från Sense HAT



### ÄNDRA VÄRDEN

Om du använder Sense HAT-emulatorn kan du ändra värdena som rapporterats av var och en av de emulerade sensorerna med hjälp av reglagen och knapparna. Försök att skjuta trycksensorns inställning ner mot botten och klicka sedan på Run igen.

För att byta till fuktighetsgivaren tar du bort de två sista raderna med kod och ersätter dem med:

```
humidity = sense.get_humidity()
print(humidity)
```

Klicka på Run så ser du att ett annat nummer skrivs till Python-skalet: den här gången är det aktuell relativ fuktighet i rummet i procent. Du kan återigen blåsa på Sense HAT (eller flytta emulatorns Humidity-reglage uppåt) så ser du att procenten blir högre när du kör programmet igen (**Bild 7-21**) – din andedräkt är förvånansvärt fuktig!

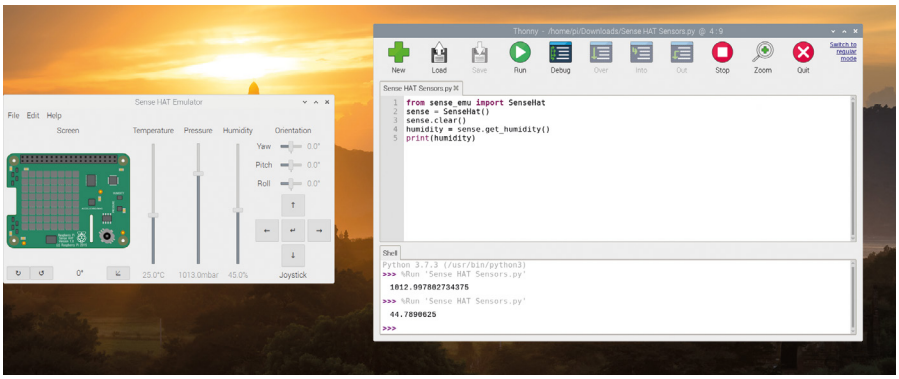



**Bild 7-21:** Visar fuktighetsgivarens avläsning

För temperaturgivaren tar du bort de två sista raderna i programmet och ersätter dem med:

```
temp = sense.get_temperature()
print(temp)
```

Klicka på Run igen så ser du en temperatur i grader Celsius (**Bild 7-22**). Det är dock inte säkert att det är den exakta temperaturen i rummet: Raspberry Pi genererar värme under hela den tid den körs och det värmer även upp Sense HAT och dess sensorer.

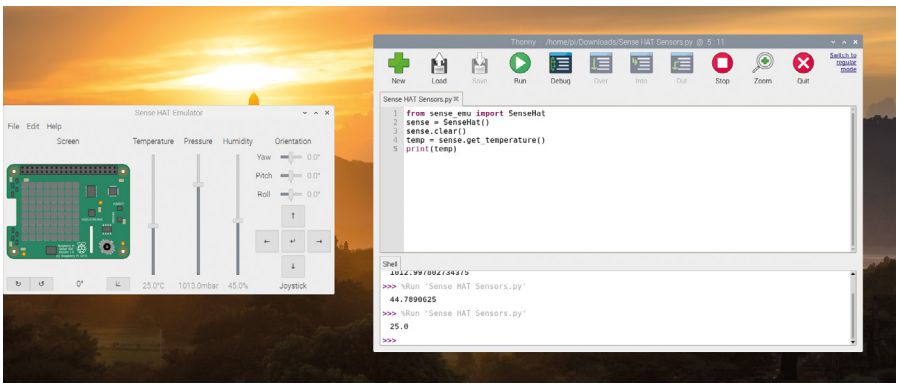
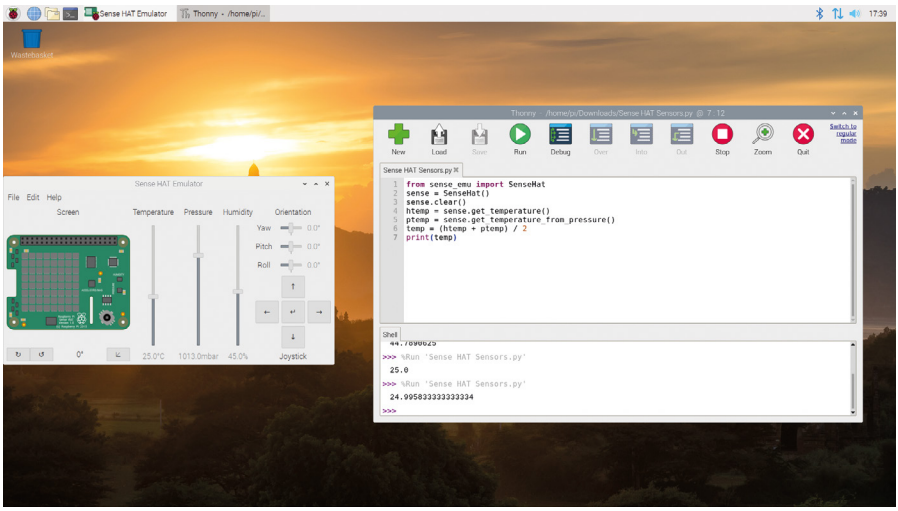



**Bild 7-22:** Visar aktuell temperaturavläsning

Sense HAT rapporterar normalt temperaturen baserat på en avläsning från temperaturgivaren som är inbyggd i fuktighetsgivaren. Om du vill använda avläsningen från trycksensorn istället ska du använda **`sense.get_temperature_from_pressure()`**. Det är också möjligt att kombinera de två avläsningarna för att få ett genomsnitt, som kan vara mer exakt än att använda varje sensor enskilt. Radera de två sista raderna i programmet och skriv:

```
htemp = sense.get_temperature()
ptemp = sense.get_temperature_from_pressure()
temp = (htemp + ptemp) / 2
print(temp)
```

Klicka på Run-ikonen så ser du att ett nummer skrivs till Python-konsolen (**Bild 7-23**). Den här gången baseras det på avläsningar från båda sensorerna, som du har lagt ihop och delat med två – antalet avläsningar – för att få ett genomsnitt av båda. Om du använder emulatorn kommer alla tre metoderna – fuktighet, tryck och genomsnitt – att visa samma nummer.




▲ **Bild 7-23:** En temperatur baserad på avläsningarna från båda sensorerna

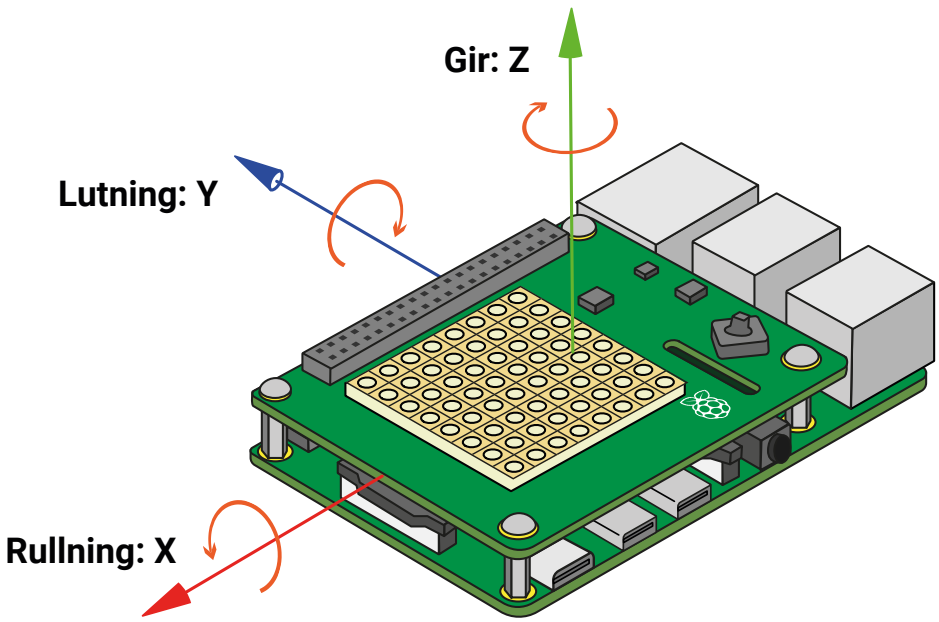
## **UTMANING:** RULLNING OCH LOOP

Kan du ändra programmet så att det läser av var och en av sensorerna i tur och ordning och sedan rullar dem över LED-matrisen istället för att skriva ut dem till skalet? Kan du göra så att programmet går i en loop, så att det ständigt skriver ut de nuvarande miljöförhållandena?

## Tröghetsavkänning

Den gyroskopiska sensorn, accelerometern och magnetometern kombineras för att bilda en så kallad *tröghetsmätningsenhet (IMU)*. Även om dessa sensorer tekniskt sett tar mätningar från den omgivande miljön precis som miljösensorerna – till exempel mäter magnetometern magnetfältets styrka – används de vanligtvis för uppgifter om rörelsen av själva Sense HAT-kortet. En IMU är summan av flera sensorer; vissa programmeringsspråk låter dig ta avläsningar enskilt från varje sensor, medan andra endast ger dig en kombinerad avläsning.

Innan du kan förstå IMU måste du dock förstå hur saker rör sig. Sense HAT, och den Raspberry Pi som den är fäst vid, kan röra sig längs tre rumsliga axlar: sida-till-sida på X-axeln; framåt och bakåt på Y-axeln; och upp och ner på Z-axeln (**Bild 7-24**). Det kan även rotera längs de här tre axlarna, men då ändras deras namn: när det roterar på X-axeln kallas det *rullning*, när det roterar på Y-axeln kallas det *lutning* och när det roterar på Z-axeln kallas det *gir*. När du roterar Sense HAT längs den korta axeln justerar du dess lutning; när du roterar det längs dess långa axel är det rullning; när du snurrar det runt medan du håller det platt på bordet justerar du dess gir. Tänk på dem som ett flygplan: när det lyfter, ökar det lutningen för att klättra; när det gör ett segervarv snurrar det bokstavligen längs sin mittaxel; när det använder rodret för att svänga som en bil skulle göra, utan att rulla, är det en gir.

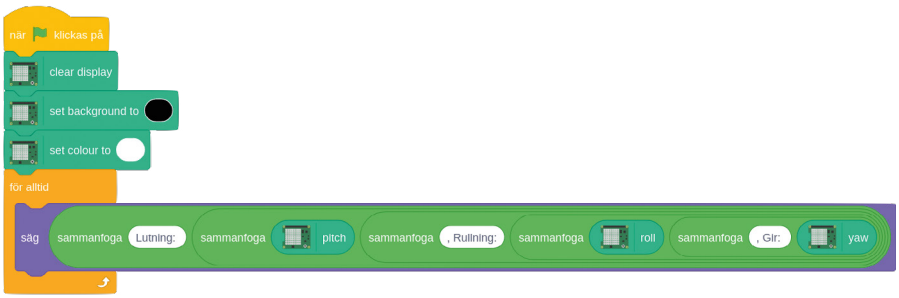



▲ **Bild 7-24:** De rumsliga axlarna i Sense HAT-kortets IMU

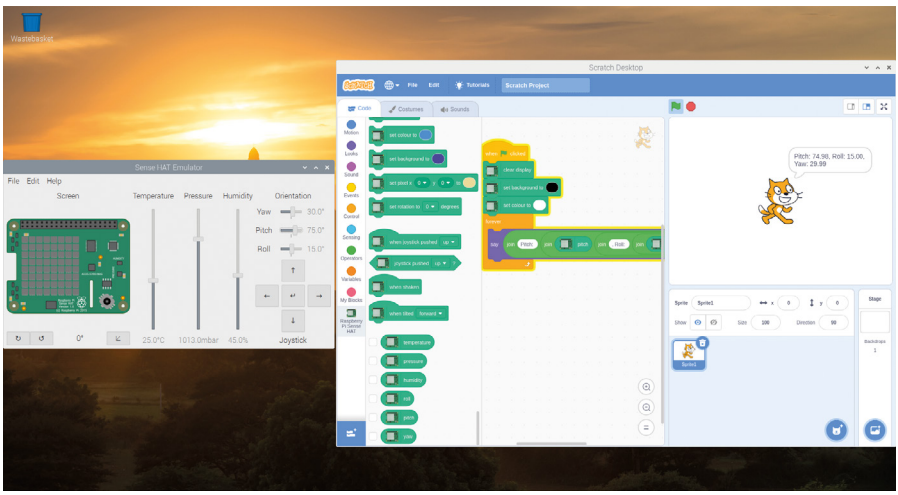
## Tröghetsavkänning i Scratch

Starta ett nytt program i Scratch och läs in Raspberry Pi Sense HAT-tillägget om det inte redan har lästs in. Starta programmet på samma sätt som förut: dra ett när klickas på -Händelser-block till kodområdet och dra sedan in ett clear display-block under det samt dra och redigera ett set background to svart- och set colour to vit-block.

Dra sedan ett för alltid-block till botten av dina befintliga block och fyll det med ett säg Hej!-block. För att visa en avläsning för var och en av de tre axlarna på IMU – lutning, rullning och gir – måste du lägga till sammanfoga-Operatorer-block plus motsvarande Sense HAT-block. Kom ihåg att inkludera mellanslag och komman så att resultatet är lätt att läsa.




Klicka på den gröna flaggan för att köra programmet och försök att flytta runt Sense HAT och Raspberry Pi – var försiktig så att du inte lossar några kablar! När du lutar Sense HAT via de tre axlarna kommer du se att värdena för lutning, rullning och gir förändras därefter (**Bild 7-25**).




▲ **Bild 7-25:** Visar värden för lutning, rullning och gir

## Tröghetsavkänning i Python

Starta ett nytt program i Thonny och spara det som **Sense HAT-rörelse**. Fyll i de vanliga startraderna och kom ihåg att använda `sense_emu` om du använder Sense HAT-emulatorn:

```
from sense_hat import SenseHat
sense = SenseHat()
sense.clear()
```

För att kunna använda information från IMU för att räkna ut den aktuella riktningen för Sense HAT på de tre axlarna skriver du följande:

```
orientation = sense.get_orientation()
pitch = orientation["pitch"]
roll = orientation["roll"]
yaw = orientation["yaw"]
print("lutning {0} rullning {1} gir {2}".format(pitch, roll, yaw))
```

Klicka på Run så visas avläsningar för Sense HAT-kortets riktning uppdelad över de tre axlarna (**Bild 7-26**). Försök att rotera Sense HAT och klicka på Run igen. Du bör då se att siffrorna ändras för att återspegla den nya riktningen.

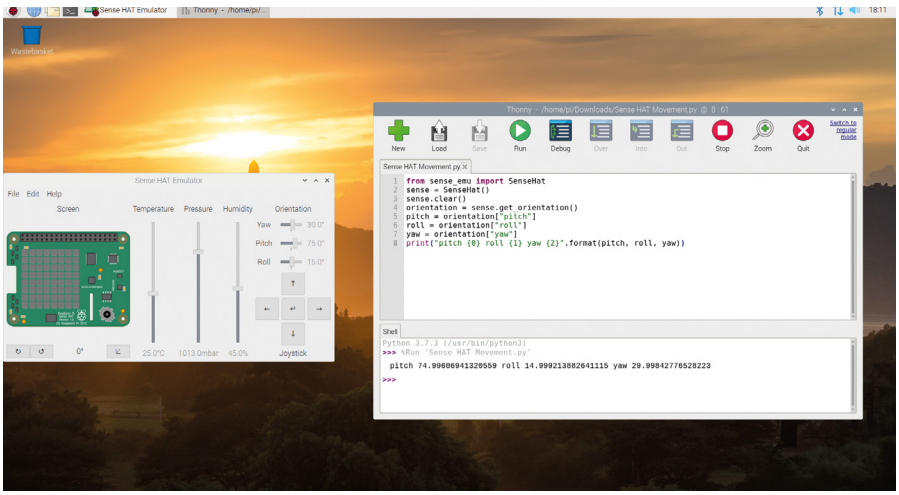



**Bild 7-26:** Visar Sense HAT-kortets värden för lutning, rullning och gir

IMU kan dock göra mer än att mäta riktning: den kan även upptäcka rörelse. För att få exakta avläsningar för rörelse måste IMU avläsas regelbundet i en loop: till skillnad från när du avläser riktning ger en enda avläsning inte någon användbar information när det gäller att upptäcka rörelse. Radera allt efter `sense.clear()` och skriv sedan följande kod:

```
while True:
    acceleration = sense.get_accelerometer_raw()
    x = acceleration["x"]
    y = acceleration["y"]
    z = acceleration["z"]
```

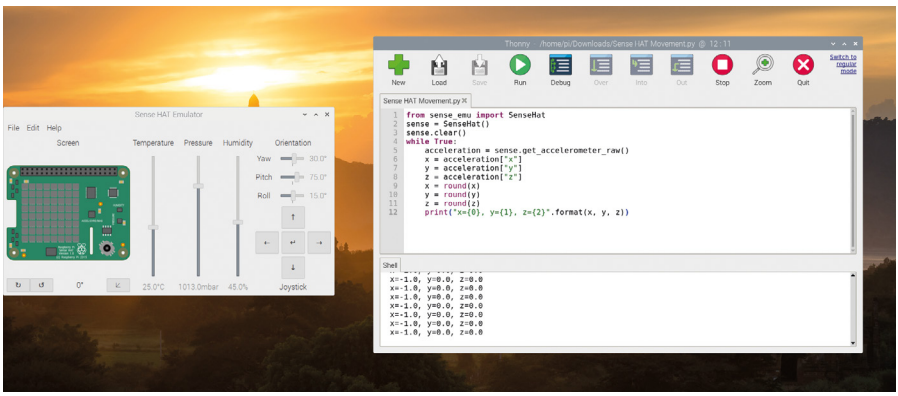
Du har nu variabler som innehåller de aktuella accelerometeravläsningarna för de tre rumsliga axlarna: X, eller vänster och höger; Y, eller framåt och bakåt; och Z, eller upp eller ner. Siffrorna från accelerometersensorn kan vara svåra att läsa av, så skriv följande för att göra dem lättare att förstå genom att avrunda dem till närmaste heltal:

```
    x = round(x)
    y = round(y)
    z = round(z)
```

Slutligen skriver du ut de tre värdena genom att skriva följande rad:

```
    print("x={0}, y={1}, z={2}".format(x, y, z))
```

Klicka på Run så ser du att värden från accelerometern skrivs till Pythons skalsområde (**Bild 7-27**). Till skillnad från de tidigare programmen kommer dessa att skrivas ut kontinuerligt. Om du vill stoppa utskriften klickar du på den röda stoppknappen som stoppar programmet.




▲ **Bild 7-27:** Accelerometeravläsningar avrundade till närmaste heltal

Du kanske har märkt att accelerometern meddelar dig att en av axlarna – Z-axeln, om Raspberry Pi ligger platt på bordet – har ett accelerationsvärde på 1,0 gravitationskonstant (1 G), men Sense HAT rör sig inte. Det beror på att den upptäcker jordens gravitationskraft – den kraft som drar Sense HAT ner mot jordens mittpunkt och som är anledningen till att föremål som ramlar ner från skrivbordet hamnar på golvet.

Samtidigt som programmet körs ska du försöka att försiktigt plocka upp Sense HAT och Raspberry Pi och rotera dem – men se till att inte lossa någon av kablarna! När Raspberry Pi:s nätverk och USB-portar pekar mot golvet ser du att värdena ändras så att Z-axeln visar 0 G och X-axeln nu visar 1 G. Vrid Rasberry Pi igen så att HDMI- och strömportarna pekar mot golvet så är det nu Y-axeln som visar 1 G. Om du gör det motsatta, så att HDMI-porten pekar mot taket, ser du istället -1 G på Y-axeln.

Med kunskapen om att jordens tyngdkraft är ungefär 1 G, och din kunskap om de rumsliga axlarna, kan du använda avläsningar från accelerometern till att ta reda på vilket håll som är ner – och på samma sätt vilket håll som är upp. Du kan även använda den för att upptäcka rörelse: försök att försiktigt skaka Sense HAT och Raspberry Pi och kolla samtidigt på siffrorna: ju hårdare du skakar, desto större är accelerationen.

När du använder **`sense.get_accelerometer_raw()`** säger du till Sense HAT att stänga av de andra två sensorerna i IMU – den gyroskopiska sensorn och magnetometern – och endast visa data från accelerometern. Naturligtvis kan du även göra samma sak med de andra sensorerna.

Leta upp raden **`acceleration = sense.get_accelerometer_raw()`** och ändra den till:

```
orientation = sense.get_gyroscope_raw()
```

Ändra ordet **`acceleration`** på alla tre raderna under den till **`orientation`**. Klicka på Run så ser du riktningen på Sense HAT för alla tre axlarna, avrundade till närmaste heltal. Till skillnad från den senaste gången du kontrollerade riktningen kommer datan den här gången endast att komma från gyroskopet utan att använda accelerometern eller magnetometern. Det här kan vara användbart om du till exempel vill veta riktningen för en rörlig Sense HAT på baksidan av en robot utan att rörelsen förvirrar saker, eller om du använder Sense HAT nära ett starkt magnetfält

Stoppa programmet genom att klicka på den röda stoppknappen. För att kunna använda magnetometern tar du bort allt från programmet bortsett från de första fyra raderna och skriver sedan följande under raden **`while True`**:

```
north = sense.get_compass()
print(north)
```

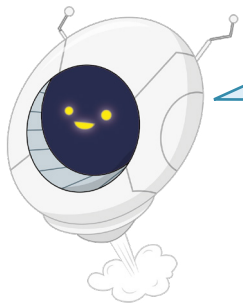
Kör programmet så ser du att riktningen för magnetisk norr skrivs till Pythons skalområde upprepade gånger. Rotera Sense HAT försiktigt så ser du att rubriken ändras allteftersom riktningen för Sense HAT i förhållande till norr skiftar: du har byggt en kompass! Om du har en magnet – det går bra med en kylskåpsmagnet – kan du försöka att flytta den runt Sense HAT för att se hur det påverkar magnetometerns avläsningar.

**UTMANING:** AUTOROTERA

Kan du skriva ett program som roterar en bild beroende på positionen för Sense HAT med hjälp av det du har lärt dig om LED-matrisen och tröghetsmätningsenhetens sensorer?

## Joystickkontroll

Sense HAT-kortets joystick, som finns i det nedre högra hörnet, må vara liten, men den är förvånansvärt kraftfull: förutom att den kan känna igen inmatningar i fyra riktningar – upp, ner, vänster och höger – har den även en femte inmatning som nås genom att man trycker ner den ovanifrån som en tryckknappsströmbrytare.



**VARNING!**

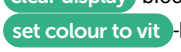
Sense HAT-kortets joystick bör endast användas om du har passat in distanshållarna enligt beskrivningen i början av det här kapitlet. Utan distanshållarna kan Sense HAT-kortet böjas när du trycker neråt på joysticken och skada både Sense HAT och Raspberry Pi:s GPIO.

## Joystickkontroll i Scratch

Starta ett nytt program i Scratch med Raspberry Pi Sense HAT-tillägget inläst. Precis som förut drar du ett 

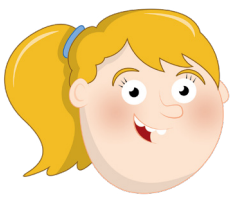
-Händelser-block till skriptområdet och drar sedan ett 

-block inunder det samt drar och redigerar ett 

- och ett 

-block.

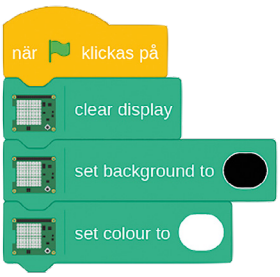
I Scratch kartläggs Sense HAT-kortets joystick till piltangenterna på tangentbordet: att trycka joysticken uppåt motsvarar att trycka på uppåtpilen, att trycka den neråt är detsamma som att trycka på nedåtpilen, att trycka den åt vänster är detsamma som att trycka på vänsterpilen och att trycka joysticken åt höger är detsamma som att trycka på högerpilen. Om du trycker joysticken inåt som en tryckknappsströmbrytare, däremot, motsvarar det att trycka på **ENTER**-tangenten.

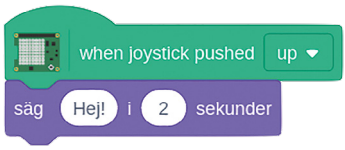


**VARNING!**

Joystickkontrollen finns endast tillgänglig på fysiska Sense HAT. När du använder Sense HAT-emulatorn använder du motsvarande tangenter på tangentbordet för att simulera joysticktryck istället.

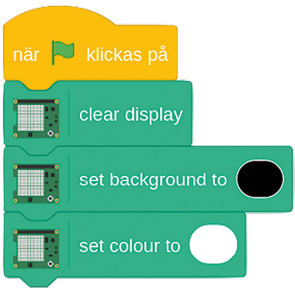
Dra ett **when joystick pushed up**-block till kodområdet. Sedan, för att ge det något att göra, drar du ett **säg Hej! i 2 sekunder**-block under det.

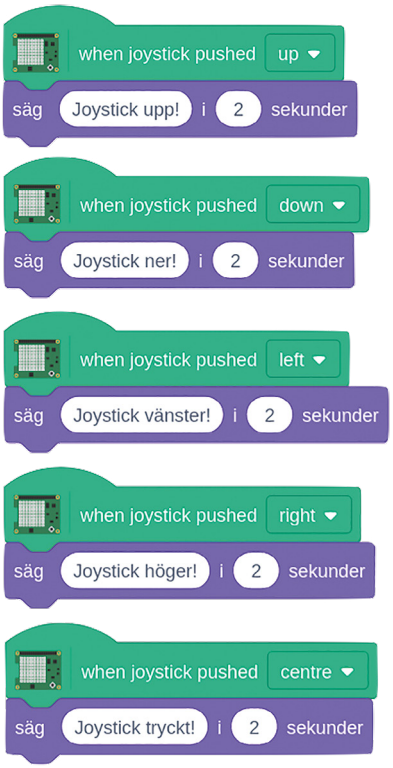


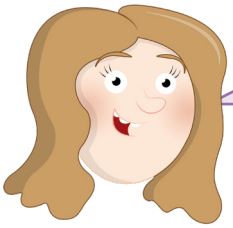

Tryck joysticken uppåt så ser du Scratch-katten säga ett glatt ”Hej!”.

Därefter redigerar du **säg Hej! i 2 sekunder**-blocket till ett **säg joystick upp! i 2 sekunder**-block och fortsätter att lägga till Händelser- och Utseende-block tills du har något för vart och ett av de fem sätt som man kan trycka på joysticken.



Försök att trycka joysticken i olika riktningar för att se dina meddelanden visas!



### SISTA UTMANINGEN

Kan du använda Sense HAT-kortets joystick för att styra en Scratch-sprajt på scenområdet? Kan du göra det på ett sådant sätt att om denna sprajt hämtar en annan sprajt, som representerar ett objekt, visar Sense HAT-kortets lysdioder ett glatt meddelande?

## Joystickkontroll i Python

Starta ett nytt program i Thonny och spara det som Sense Hat Joystick. Börja med de vanliga tre raderna som ställer in Sense HAT och rensar LED-matrisen:

```
from sense_hat import SenseHat
sense = SenseHat()
sense.clear()
```

Sätt sedan upp en oändlig loop:

```
while True:
```

Beordra sedan Python att lyssna efter inmatningar från Sense HAT-joysticken med följande rad, som Thonny automatiskt kommer att dra in åt dig:

```
    for event in sense.stick.get_events():
```

Lägg slutligen till följande rad – som Thonny återigen kommer att dra in åt dig – för att något faktiskt ska hända när ett joysticktryck upptäcks:

```
        print(event.direction, event.action)
```

Klicka på Run och försök att trycka joysticken i olika riktningar. Du ser den riktning som du valt att skriva till Pythons skalområde: upp, ner, vänster, höger och mitten för när du har tryckt ner joysticken som en tryckknappsströmbrytare.

Du ser även att du får två händelser varje gång du trycker en gång på joysticken: en händelse, **`pressed`**, för när du först trycker en riktning; den andra händelsen, **`released`**, för när joysticken går tillbaka till mitten. Du kan använda det här i programmen: tänk på en karaktär i ett spel, som du kan få att börja röra sig när joysticken trycks i en riktning och stanna så snart den släpps.

Du kan även använda joysticken för att utlösa funktioner, snarare än att vara begränsad till att använda en bestämd loop. Radera allt under **`sense.clear()`** och skriv följande:

```
def red():
    sense.clear(255, 0, 0)

def blue():
    sense.clear(0, 0, 255)

def green():
    sense.clear(0, 255, 0)

def yellow():
    sense.clear(255, 255, 0)
```

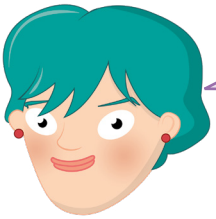
Dessa funktioner ändrar hela LED-matrisen på Sense HAT till en enda färg: röd, blå, grön eller gul – vilket visar att programmet är extremt enkelt att hantera! Du måste berätta för Python vilken funktion som passar med vilken joystickinmatning för att de faktiskt ska utlösas. Skriv följande rader:

```
sense.stick.direction_up = red
sense.stick.direction_down = blue
sense.stick.direction_left = green
sense.stick.direction_right = yellow
sense.stick.direction_middle = sense.clear
```

Slutligen behöver programmet en oändlig loop – känd som den *huvudsakliga* loopen – för att fortsätta köras och därmed fortsätta leta efter joystickinmatningar, snarare än att bara köra igenom koden du har skrivit en gång och sedan sluta. Skriv följande två rader:

```
while True:
    pass
```

Klicka på Run och försök att flytta joysticken: du kommer se att lysdioderna lyser i en härlig färg! Om du vill stänga av lysdioderna trycker du på joysticken som en tryckknapp: **`middle`**-riktningen är inställd på att använda **`sense.clear()`**-funktionen för att stänga av dem alla. Grattis: du kan ta emot inmatningar från joysticken!



## SISTA UTMANINGEN

Kan du använda det du har lärt dig till att rita en bild på skärmen och få den att rotera i den riktning du trycker joysticken? Kan du göra så att mitteninmatningen växlar mellan mer än en bild?

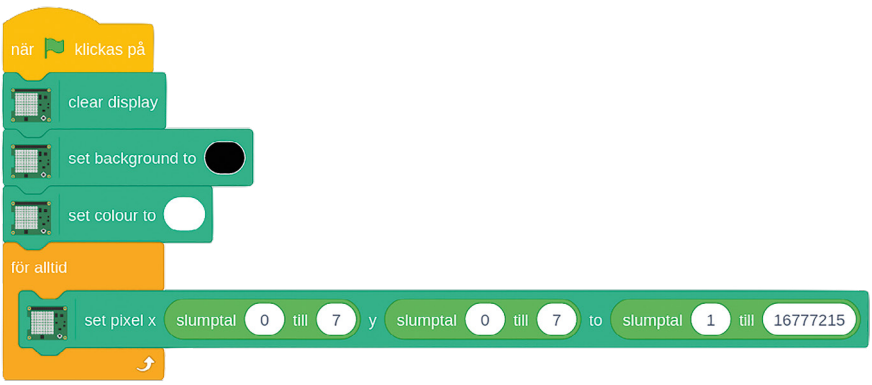
## Scratch-projekt: Sprakande Sense HAT

Nu när du vet hur du hanterar Sense HAT är det dags att lägga samman allt du har lärt dig för att bygga en värmekänslig sprakande enhet – en enhet som är som lyckligast när den är kall och som gradvis saktar ner ju varmare den blir.

Starta ett nytt Scratch-projekt och lägg till Raspberry Pi Sense HAT-tillägget om det inte redan har lästs in. Börja som alltid med fyra block: när klickas på , clear display , set background to svart och set colour to vit . Kom ihåg att du måste ändra färgerna från standardinställningarna.

Börja med att skapa en enkel men konstnärlig sprakande enhet. Dra ett för alltid -block till kodområdet och fyll det med ett set pixel x 0 y 0 to colour -block. Istället för att använda bestämda siffror fyller du i vart och ett av avsnitten x, y och färg i det blocket med ett slumptal 1 till 10 -Operatorer-block.

Värdena 1 till 10 är inte särskilt användbara här så du måste göra lite ändringar. De två första siffrorna i set pixel -blocket är X- och Y-koordinaterna för pixeln på LED-matrisen, vilket betyder att de ska vara en siffra mellan 0 och 7 – så ändra de två första blocken så att de visar slumptal 0 till 7 . Nästa avsnitt är färgen som pixeln ska ställas in till. När du använder färgväljaren visas den färg du väljer direkt i skriptområdet. Internt representeras dock färgerna av ett tal och du kan använda talet direkt. Redigera det senast valda slumpmässiga blocket så att det står slumptal 0 till 16777215 .




Klicka på den gröna flaggan så ser du att lysdioderna på Sense HAT börjar lysa i slumpmässiga färger (**Bild 7-28**). Grattis: du har skapat en elektronisk sprakande enhet!

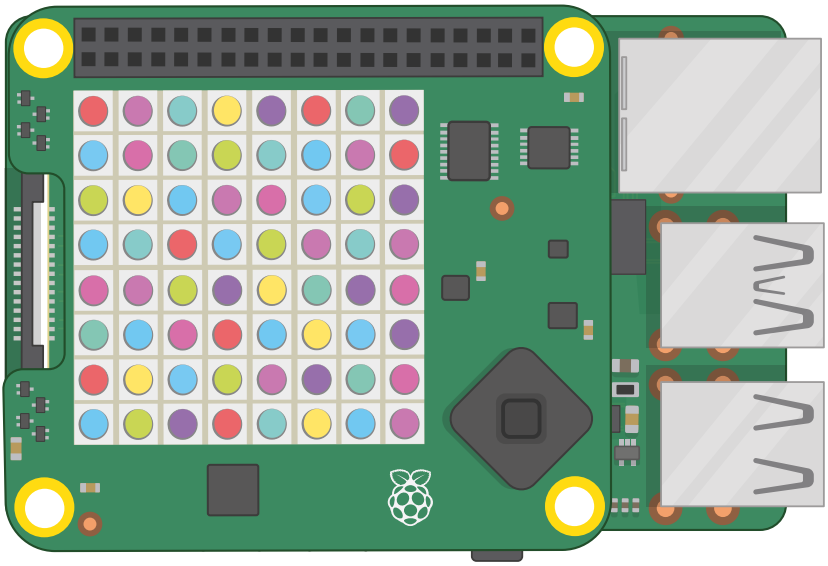
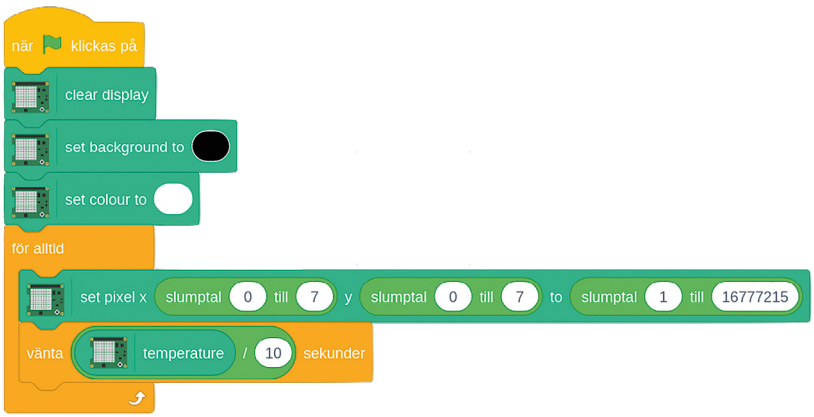


**Bild 7-28:** Pixlarna tänds i slumpmässiga färger

Den sprakande enheten är inte särskilt interaktiv. För att ändra på det börjar du med att dra ett **vänta 1 sekunder**-block så att det ligger under **set pixel**-blocket men i **för alltid**-blocket. Dra ett **( ) / ( )**-Operatorer-block över 1 och skriv sedan 10 i dess andra utrymme. Dra slutligen ett **temperature**-block över det första utrymmet i det delade Operatorer-blocket.




Klicka på den gröna flaggan så märker du – om du inte bor någonstans där det väldigt kallt – att den sprakande enheten är betydligt långsammare än tidigare. Det beror på att du har skapat en temperaturberoende fördröjning: programmet väntar nu *den aktuella temperaturen delat med 10* antal sekunder före varje loop. Om temperaturen i rummet är 20 °C väntar programmet 2 sekunder innan det startar loopen; om temperaturen är 10 °C väntar det 1 sekund; om det är under 10 °C väntar det mindre än en sekund.

Om Sense HAT avläser en negativ temperatur – under 0 °C, vattnets fryspunkt – försöker den vänta mindre än 0 sekunder, men eftersom det är omöjligt – i alla fall utan att uppfinna tidsresor – kommer du att se samma effekt som om den väntade i 0 sekunder. Grattis: nu kan du försöka integrera de olika funktionerna i Sense HAT i dina egna program!

## Python-projekt: Sense HAT-tricorder

Nu när du vet hur du hanterar Sense HAT är det dags att lägga samman allt du har lärt dig för att bygga en tricorder – en enhet som är välbekant för fans till en viss science-fiction-serie för att kunna rapportera för flera sensorer som är inbyggda i den.

Starta ett nytt projekt i Thonny och spara det som **Tricorder**. Börja sedan med de vanliga raderna som du behöver för att göra ett Sense HAT-program:

```
from sense_hat import SenseHat
sense = SenseHat()
sense.clear()
```

Därefter behöver du fastställa funktioner för var och en av de olika sensorerna i Sense HAT. Börja med tröghetsmätarenheten genom att skriva:

```
def orientation():
    orientation = sense.get_orientation()
    pitch = orientation["pitch"]
    roll = orientation["roll"]
    yaw = orientation["yaw"]
```

Eftersom du ska rulla resultaten från sensorn över lysdioderna är det vettigt att avrunda dem så att du inte väntar på dussintals decimaler. Avrunda dem dock till en decimal, istället för heltal, genom att skriva följande:

```
    pitch = round(pitch, 1)
    roll = round(roll, 1)
    yaw = round(yaw, 1)
```

Slutligen måste du säga åt Python att rulla resultaten till lysdioderna, så att tricordern fungerar som en handhållen enhet utan att behöva anslutas till en bildskärm eller tv:

```
    sense.show_message("Lutning {0}, Rullning {1}, Gir {2}".
format(pitch, roll, yaw))
```

Nu när du har en fullständig funktion för att läsa av och visa riktningen från IMU måste du skapa liknande funktioner för var och en av de andra sensorerna. Börja med temperaturgivaren:

```
def temperature():
    temp = sense.get_temperature()
    temp = round(temp, 1)
    sense.show_message("Temperatur: %s grader Celsius" % temp)
```

Titta noga på raden som skriver ut resultatet till lysdioderna: **%s** kallas en platshållare och ersätts med innehållet i variabeln **temp**. Med hjälp av detta kan du snyggt formatera resultatet med en etikett, ”Temperatur:” och en måttenhet, ”grader Celsius”, vilket gör programmet mycket mer lättanvänt.

Därefter fastställer du en funktion för fuktighetsgivaren:

```
def humidity():
    humidity = sense.get_humidity()
    humidity = round(humidity, 1)
    sense.show_message("Fuktighet: %s procent" % humidity)
```

Därefter trycksensorn:

```
def pressure():
    pressure = sense.get_pressure()
    pressure = round(pressure, 1)
    sense.show_message("Tryck: %s millibar" % pressure)
```

Och slutligen kompassavläsningen från magnetometern:

```
def compass():
    for i in range(0, 10):
        north = sense.get_compass()
    north = round(north, 1)
    sense.show_message("Norr: %s grader" % north)
```

Den korta **for**-loopen i denna funktion tar tio avläsningar från magnetometern för att säkerställa att den har tillräckligt med data för att ge dig ett exakt resultat. Om du upptäcker att det rapporterade värdet fortsätter att skifta kan du försöka att utöka det till 20, 30 eller till och med 100 loopar för att förbättra noggrannheten ytterligare.

Programmet har nu fem funktioner, som var och en tar en avläsning från en av sensorerna i Sense HAT och rullar dem över lysdioderna. Det behöver dock ett sätt att välja vilken sensor du vill använda, och joysticken är perfekt för det.

Skriv följande:

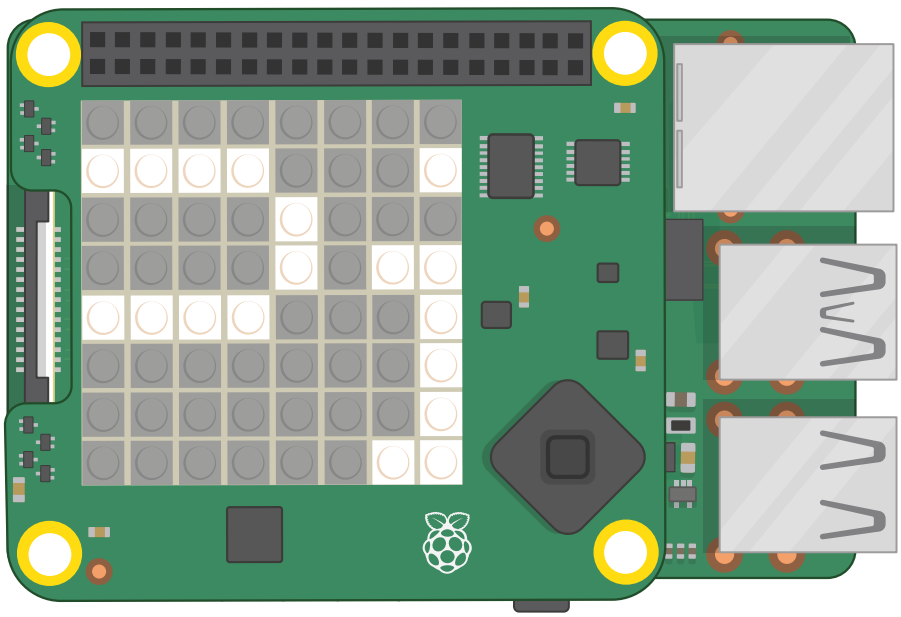
```
sense.stick.direction_up = orientation
sense.stick.direction_right = temperature
sense.stick.direction_down = compass
sense.stick.direction_left = humidity
sense.stick.direction_middle = pressure
```

Dessa rader tilldelar en sensor till var och en av de fem möjliga riktningarna på joysticken: upp avläser från orienteringssensorn; ner avläser från magnetometern; vänster avläser från fuktighetsgivaren; höger avläser från temperaturgivaren; och om du trycker på spaken i mitten kommer avläsningen från trycksensorn.

Slutligen behöver du en huvudsaklig loop så att programmet fortsätter att lyssna efter joysticktryck och inte omedelbart avslutas. Allra längst ner i programmet skriver du följande:

```
while True:
    pass
```

Klicka på Run och försök att flytta joysticken för att ta en avläsning från en av sensorerna (**Bild 7-29**). När det har slutat rulla resultatet trycker du i en annan riktning. Grattis: du har byggt en handhållen tricorder som skulle göra Förenade planeters federation stolt!



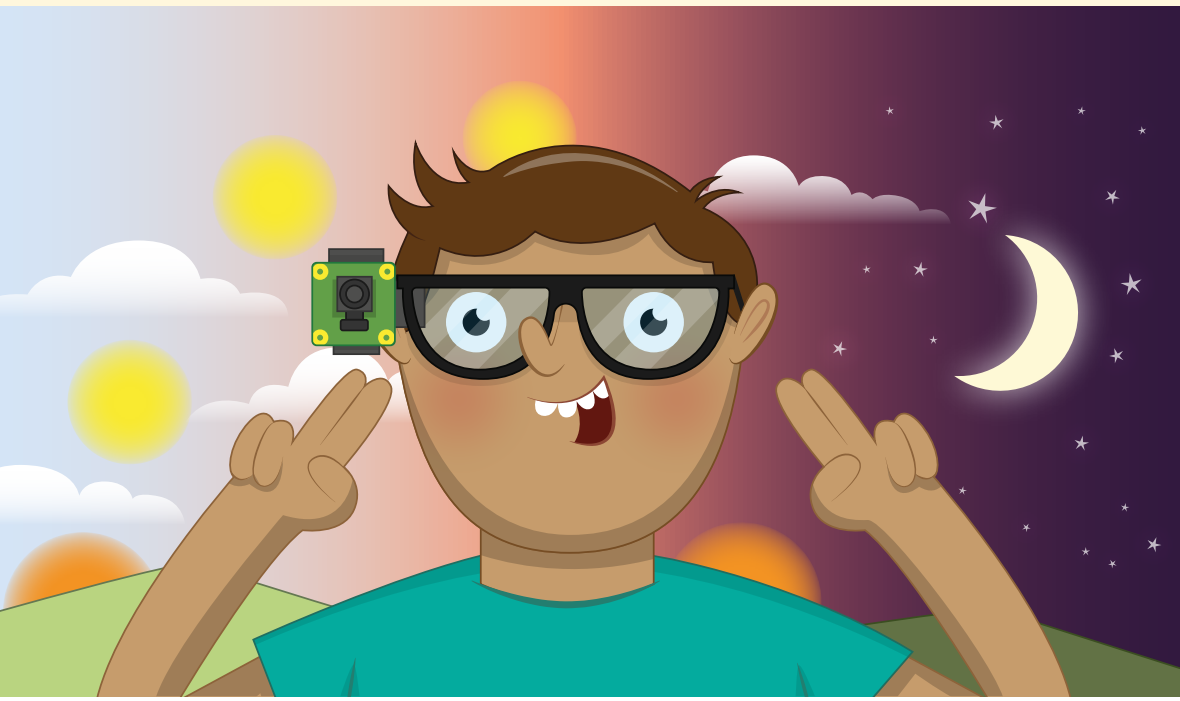
▲ **Bild 7-29:** Varje avläsning rullar över skärmen

För att hitta fler Sense HAT-projekt följer du länkarna i **Bilaga D, Ytterligare information**.

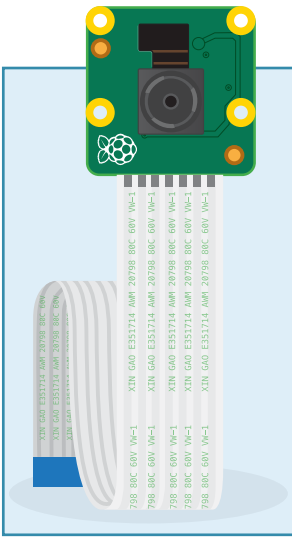
Kapitel 8

# Raspberry Pi Camera Module

**Genom att ansluta en Camera Module eller HQ Camera till Raspberry Pi kan du ta högupplösta bilder och spela in videor samt skapa fantastiska projekt med datorseende**



Om du någonsin har velat bygga något som kan se – känt inom det robottekniska området som *datorseende* – är Raspberry Pi Camera Module (tillval), eller den nya High Quality Camera, en utmärkt utgångspunkt. Camera Module/HQ Camera är ett litet fyrkantigt kretskort med en tunn bandkabel som ansluts till Camera Serial Interface-porten (CSI) på Raspberry Pi (finns inte på Raspberry Pi 400). Den ger stillbilder med hög upplösning och rörliga videosignaler som kan användas som de är eller integreras i dina egna program.



## KAMERATYPER!

Tre typer av Raspberry Pi-kameror finns tillgängliga: den standardmässiga Camera Module, ”NoIR”-versionen och High Quality (HQ) Camera. Om du vill ta vanliga bilder och videor i väl upplysta miljöer bör du använda den standardmässiga Camera Module. Om du vill använda speciella objektiv och letar efter bästa bildkvalitet bör du använda HQ Camera Module. NoIR Camera Module – som kallas så eftersom den inte har något infrarött- eller IR-filter – är utformad för användning med infraröda ljuskällor för att ta bilder och videor i totalt mörker. Om du bygger ett rede, en säkerhetskamera eller ett annat projekt som involverar mörkerseende vill du ha NoIR-versionen – men kom ihåg att samtidigt köpa en infraröd ljuskälla!

Den standardmässiga Camera Module och NoIR Camera Module från Raspberry Pi är baserade på en Sony IMX219-bildsensor. Detta är en *8 megapixels sensor*, vilket betyder att den kan ta bilder med upp till 8 miljoner pixlar i dem. Det gör den genom att ta bilder som är upp till 3 280 pixlar breda och 2 464 pixlar höga. Förutom att ta stillbilder kan Camera Module spela in videofilmer med full HD-upplösning med en hastighet på 30 bilder per sekund (30 fps). För att få en mjukare rörelse eller till och med för att skapa en slowmotion-effekt, kan kameran ställas in på att ta bilder med högre bildhastighet genom att upplösningen sänks: 60 fps för videobilder med 720 p och upp till 90 fps för bilder med 480 p (VGA).

High Quality Camera använder en Sony IMX477-sensor på 12,3 megapixel, som också är större än den i den standardmässiga Camera Module och NoIR Camera Module – vilket innebär att den kan samla in mer ljus och därmed ta bilder av högre kvalitet. Till skillnad från Camera Modules innehåller HQ Camera dock inte ett objektiv, utan vilket det inte kan ta några bilder eller videor. Du kan använda valfritt objektiv med C- eller CS-fattning; andra objektivmonteringar kan användas med en lämplig adapter för C- eller CS-fattning. För information om hur du monterar ett objektiv, se **Bilaga F: High Quality Camera**.

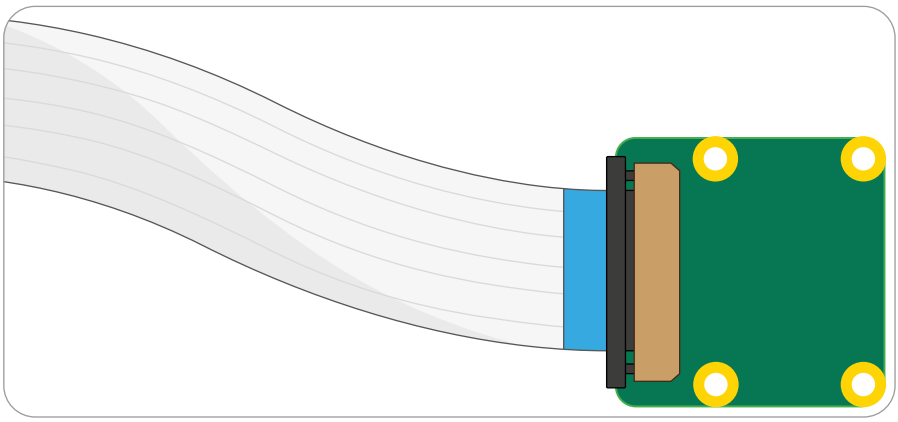
## RASPBERRY PI 400

Tyvärr är Raspberry Pi Camera Modules inte kompatibla med Raspberry Pi 400. Du kan använda USB-webbkameror som ett alternativ, men du kommer inte att kunna använda programverktyg som är specifika för Raspberry Pi Camera Module och som ingår i Raspberry Pi OS.

## Installera kameran

Camera Module eller HQ Camera bör, liksom alla maskinvarutillägg, endast anslutas till eller kopplas bort från Raspberry Pi när strömmen är avstängd och strömkabeln urkopplad. Om Raspberry Pi är påslagen väljer du Shutdown från hallonmenyn, väntar på att den stängs av och kopplar bort den.

I de flesta fall är den medföljande bandkabeln redan ansluten till Camera Module eller HQ Camera. Om den inte är det vänder du kamerakortet upp och ner så att sensorn är inunder och letar efter ett platt anslutningsdon i plast. Haka försiktigt i naglarna runt de utstickande kanterna och dra utåt tills anslutningsdonet är delvis utdraget. Skjut bandkabeln, med silverkanterna nedåt och den blå plasten uppåt, under klaffen som du precis drog ut och tryck sedan försiktigt tillbaka klaffen på plats med ett klick (**Bild 8-1**). Det spelar ingen roll vilken ände av kabeln du använder. Om kabeln är ordentligt installerad kommer den att vara rak och lossnar inte om du drar försiktigt i den. Om den lossnar drar du ut klaffen och försöker igen.



▲ **Bild 8-1:** Anslut bandkabeln till Camera Module

Installera kabelns andra ände på samma sätt. Leta upp kameraporten (eller CSI-porten) på Raspberry Pi och dra klaffen försiktigt uppåt. Om Raspberry Pi är installerad i ett fodral kan det vara lättare att ta bort det först. Placera Raspberry Pi så att HDMI-porten är vänd mot dig och skjut in bandkabeln så att silverkanterna är till vänster om dig och den blå plasten till höger (**Bild 8-2**). Tryck sedan försiktigt tillbaka klaffen på plats. Om kabeln är ordentligt installerad kommer den att vara rak och lossnar inte om du drar försiktigt i den. Om den lossnar drar du ut klaffen och försöker igen.

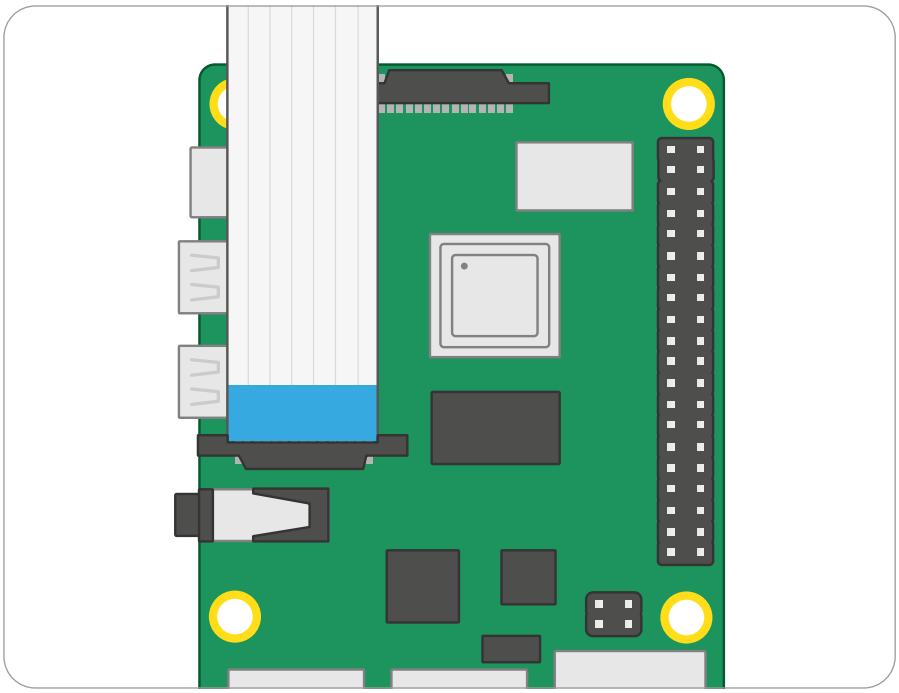
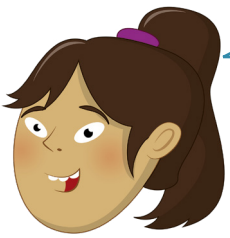


**Bild 8-2:** Anslut bandkabeln till kamera/CSI-porten på Raspberry Pi

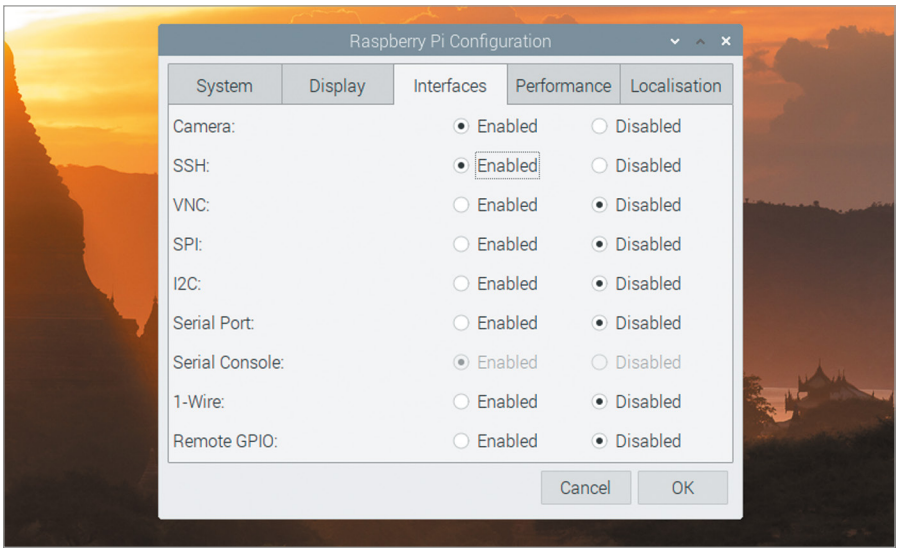
Camera Module levereras med en liten bit blå plast som täcker linsen för att skydda den från repor under tillverkning, leverans och installation. Leta upp den lilla plastfliken och dra försiktigt av den från linsen för att göra kameran redo att användas.



### JUSTERA FOKUS

Camera Module levereras vanligtvis med ett litet plasthjul som används för att justera objektivets fokus. Även om det fabriksinställda fokuset vanligtvis är perfekt kan du skjuta ratten över objektivet och vrida den försiktigt för att justera fokus manuellt om du använder kameran för mycket närbildsarbete. För information om att fokusera HQ Camera, se **Bilaga F**.

Anslut strömförsörjningen till Raspberry Pi och låt den ladda Raspberry Pi OS. Innan du kan använda kameran måste du meddela Raspberry Pi att en kamera är ansluten: öppna hallonikonmenyn, välj kategorin Inställningar och klicka på Raspberry Pi Configuration. När verktyget har lästs in klickar du på fliken Interfaces, letar upp posten Camera i listan och klickar på den runda alternativknappen till vänster om ”Enabled” för att slå på den (**Bild 8-3**, på nästa sida). Klicka på OK så kommer verktyget att uppmana dig att starta om din Raspberry Pi. Gör det, och sedan är kamera redo att användas!

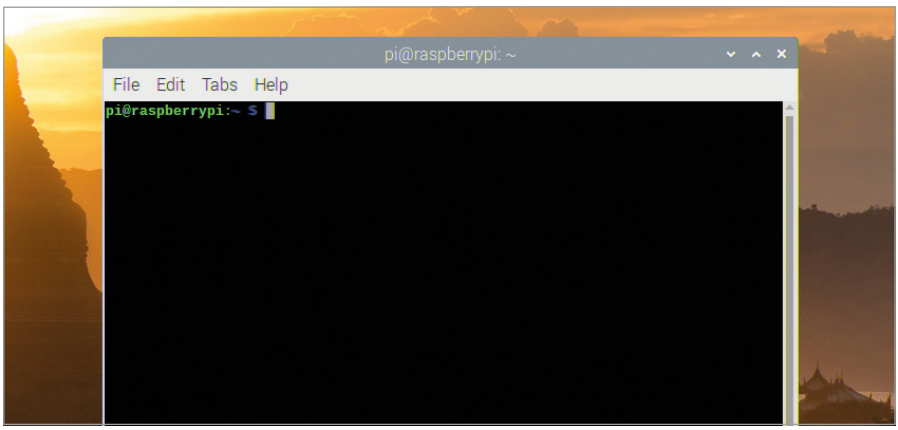



▲ **Bild 8-3:** Du måste aktivera kameran i Raspberry Pi Configuration

## Testa kameran

Du kan använda verktyget **raspistill** för att bekräfta att Camera Module eller HQ Camera är ordentligt installerad och att du har aktiverat gränssnittet i verktyget Raspberry Pi Configuration. Detta, tillsammans med **raspivid** för videor, är utformat för att ta bilder från kameran med hjälp av Raspberry Pi:s *kommandoradsgränssnitt (CLI)*.

Till skillnad från de program du hittills har använt hittar du inte raspistill i menyn. Du klickar istället på hallonikonmenyn för att läsa in menyn, väljer kategorin Tillbehör och klickar på LXTerminal. Ett svart fönster med grön och blå text visas (**Bild 8-4**): detta är *terminalen* som låter dig få tillgång till kommandoradsgränssnittet.




▲ **Bild 8-4:** Öppna ett fönster i LXTerminal för att ange kommandon

För att testa kameran skriver du följande i LXTerminal:

```
raspistill -o test.jpg
```

När du trycker på **ENTER** visas en stor bild av vad kameran ser på skärmen (**Bild 8-5**). Detta kallas *liveförhandsgranskningen* och, om du inte meddelar raspistill annat, kommer den att visas i 5 sekunder. När dessa 5 sekunder är slut kommer kameran att ta en enda stillbild och spara den i hemmappen under namnet **test.jpg**. Om du vill ta en till bild skriver du samma kommando igen – men se till att ändra filnamnet för utmatning, efter **`-o`**, så att du inte sparar över den första bilden!

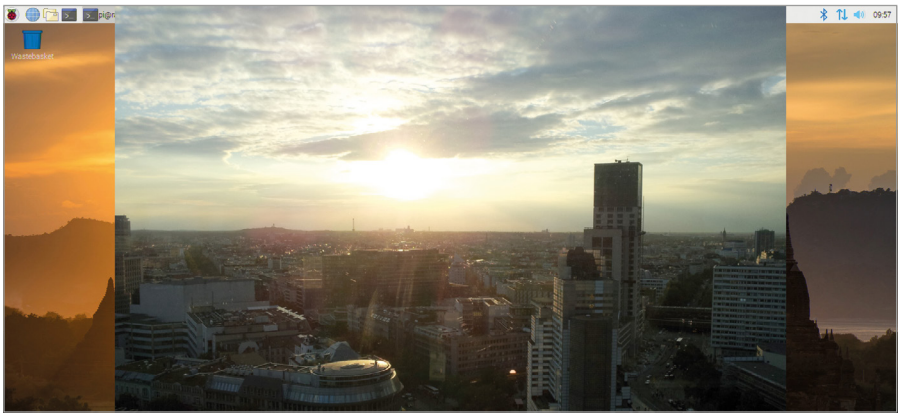


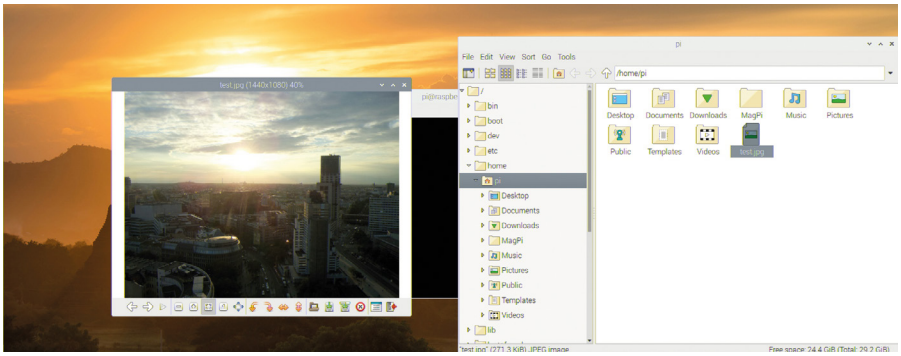
**Bild 8-5:** Liveförhandsgranskningen från kameran

Om liveförhandsgranskningen visades upp och ner måste du meddela raspistill att kameran är vänd. Camera Module är utformad så att bandkabeln ska komma ut ur den nedersta kanten. Om den kommer ut på sidorna eller från överdelen, som på vissa andra tredjeparts tillbehör för kamerafattningar, kan du rotera bilden 90, 180 eller 270 grader med **`-rot`**-kopplaren. För en kamera som är monterad så att kabeln kommer ut från överdelen, använder du bara följande kommando:

```
raspistill -rot 180 -o test.jpg
```

Om bandkabeln kommer ut från den högra kanten använder du ett rotationsvärde på 90 grader. Om den kommer ut från den vänstra kanten använder du 270 grader. Om den ursprungliga bilden togs i fel vinkel kan du prova en annan och använda **`-rot`**-kopplaren för att korrigera den.

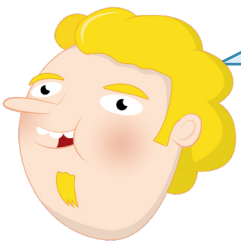
Om du vill se bilden öppnar du Filhanteraren från kategorin Tillbehör i hallonmenyn: bilden du har tagit, kallad **test.jpg**, kommer att ligga i **home/pi**-mappen. Leta upp den i listan över filer och dubbelklicka sedan på den för att läsa in den i en bildvisare (**Bild 8-6**). Du kan även bifoga bilden till e-postmeddelanden, ladda upp den på webbplatser via webbläsaren eller dra den till en extern lagringsenhet.




▲ **Bild 8-6:** Öppna den tagna bilden

## Introduktion av picamera

Det mest flexibla sättet att styra Camera Module eller HQ Camera är med hjälp av Python, via det praktiska picamera-biblioteket. Det ger dig full kontroll över kamerans förmåga till förhandsgranskning, bilder och videoinspelning och låter dig integrera dem i dina egna program – och till och med kombinera dem med program som använder GPIO-modulen via GPIO Zero-biblioteket!



### PYTHON-PROGRAMMERING

Projekten i detta kapitel förutsätter att du har erfarenhet av Python-programmeringsspråket, Thonny IDE och Raspberry Pi:s GPIO-stift. Gå till **Kapitel 5, Programmering med Python** och **Kapitel 6, Fysisk databehandling med Scratch och Python**, om du inte redan har gjort det, och arbeta först igenom projekten i dessa kapitel!

Stäng terminalen, om den fortfarande är öppen, genom att klicka på X längst upp till höger i fönstret och läs sedan in Thonny från kategorin Programmering i hallonmenyn. Spara det nya projektet som **Kamera** och börja sedan importera biblioteken som programmet behöver genom att skriva följande i skriptområdet:

```
from picamera import PiCamera
from time import sleep
camera = PiCamera()
```

Den sista raden gör att du kan styra Camera Module eller HQ Camera med hjälp av **camera**-funktionen. Skriv följande för att starta:

```
camera.start_preview()
sleep(10)
camera.stop_preview()
```

Klicka på Run så försvinner skrivbordet. Istället ser du en förhandsgranskning i helskärm av allt som kameran kan se (**Bild 8-7**). Försök att flytta runt den eller vinka med handen framför linsen så ser du att bilden på skärmen ändras så att den matchar. Efter 10 sekunder stängs förhandsgranskningen och programmet avslutas – men till skillnad från förhandsgranskningen från raspistill kommer ingen bild att sparas efteråt.

**Bild 8-7:** En liveförhandsgranskning av kameravyn i helskärm

Om förhandsgranskningen visades upp och ner kan du rotera bilden för att få den åt rätt håll igen. Precis under raden **camera = PiCamera()** skriver du:

```
camera.rotation = 180
```

Om förhandsgranskningen visades upp och ner kommer den raden att få det att se rätt ut igen. Precis som med raspistill kan du med hjälp av **camera.rotation** rotera bilden 90, 180 eller 270 grader, beroende på om kabeln kommer ut från höger sida, överdelen eller vänster sida av Camera Module. Kom ihåg att använda **camera.rotation** i början av alla program du skriver, för att undvika att ta bilder eller videor som är upp och ner!

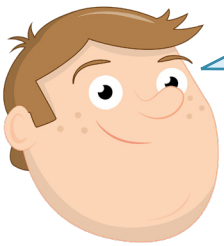
## Ta stillbilder

För att ta en bild, snarare än att endast visa en liveförhandsgranskning, måste programmet ändras. Börja med att minska fördröjningen för förhandsgranskningen: leta upp raden **`sleep(10)`** och ändra den så att det står:

```
sleep(5)
```

Precis under den raden lägger du till följande:

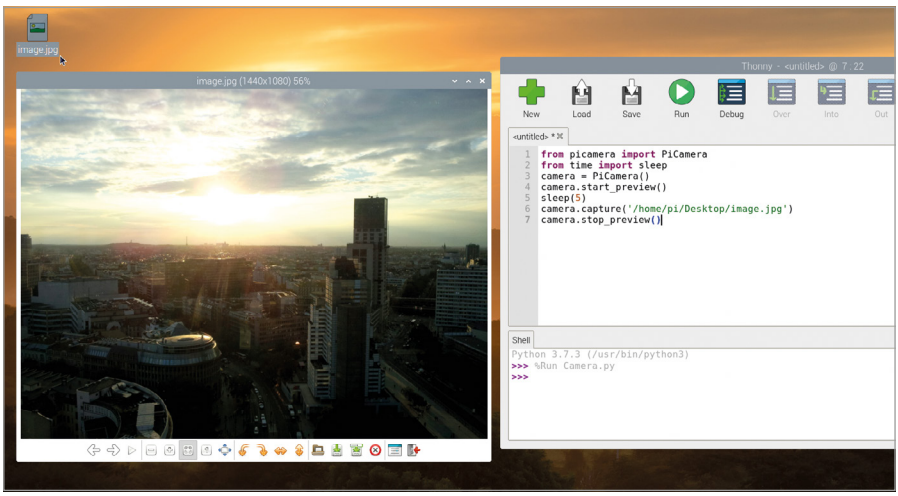
```
camera.capture('/home/pi/Desktop/image.jpg')
```



### DAGS ATT JUSTERA

När kameran är i förhandsgranskningsläge analyserar den videon för att se om den behöver justeras för att få bästa kvalitet. Du kommer att se detta om du befinner dig i en mycket mörk eller mycket ljus miljö: förhandsgranskningen är först omöjlig att se och sedan blir den snabbt tydligare. För att ge kameran tid att justera ska du alltid lägga till en förhandsgranskningsperiod på minst 2 sekunder i programmet innan du tar en bild.

**`camera.capture`**-funktionen meddelar Python att det ska spara en stillbild, och det behöver inte bara veta vad bilden ska kallas utan även i vilken mapp den ska sparas. I det här exemplet sparar du den till skrivbordet – hitta den genom att titta strax under Papperskorgen. Om Thonny-fönstret är i vägen klickar du bara på och drar titelfältet för att flytta det. Dubbelklicka på filen för att se bilden du har tagit (**Bild 8-8**). Grattis: du har programmerat en kamera!




▲ **Bild 8-8:** Öppna den tagna bilden

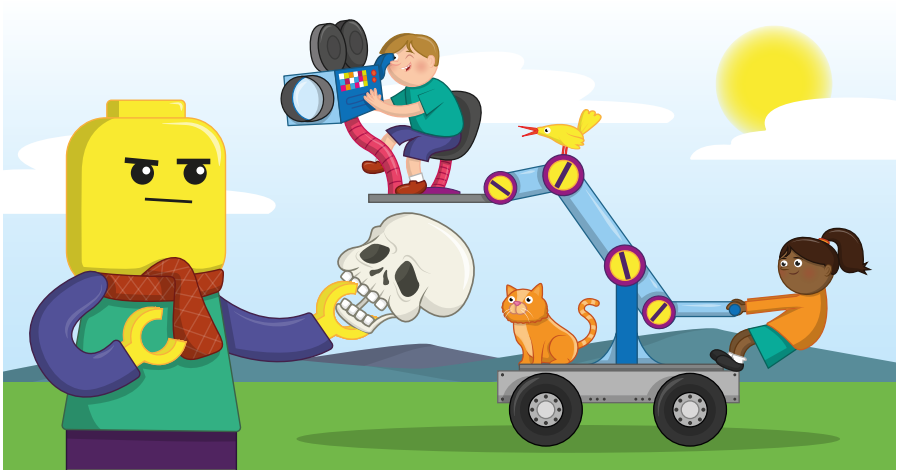
### Spela in rörlig video

Förutom att ta stillbilder kan du spela in videoklipp. Radera allt mellan raderna **`camera.start_preview()`** och **`camera.stop_preview()`** och skriv sedan följande under **`camera.start_preview():`**

```
camera.start_recording('/home/pi/Desktop/video.h264')
sleep(10)
camera.stop_recording()
```

Kamerans förhandsgranskning visas som tidigare, men den här gången spelas den även in i en fil på skrivbordet. Vänta i de 10 sekunder som du har meddelat Python att vara i viloläge – du kan alltid dansa en liten dans framför kameran för att göra videon intressant – och sedan, när förhandsgranskningen har stängts, hittar du videofilen på skrivbordet.

För att spela upp videon dubbelklickar du bara på filen **video.h264** på skrivbordet. Videon börjar spela – och om du dansade ser du att dansen spelas upp för dig! När videon är slut avslutas uppspelningsprogrammet med ett meddelande i LXTerminal. Grattis: du kan nu spela in videoklipp med Raspberry Pi Camera Module eller HQ Camera!



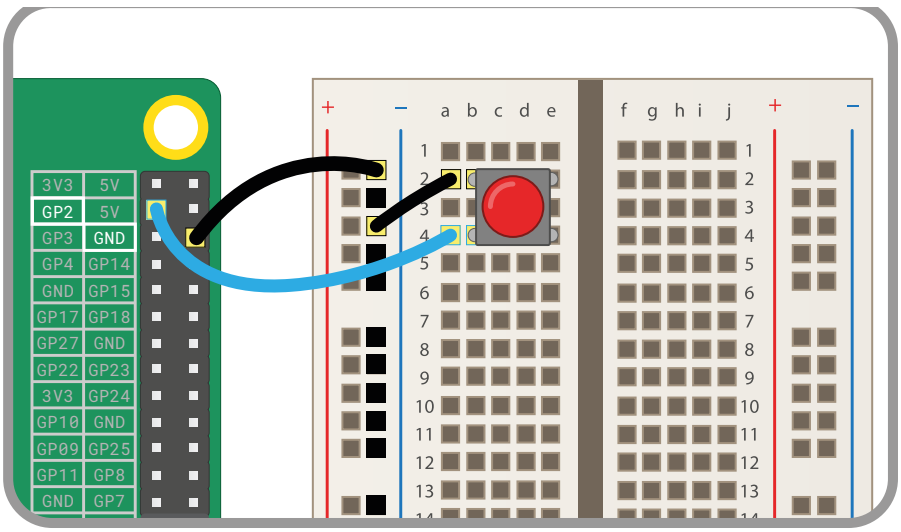
## Stop-motion-animering med tryckknapp

Med hjälp av det du har lärt dig i det här kapitlet, och din kunskap om hur du ansluter maskinvara till Raspberry Pi:s GPIO-rubrik från **Kapitel 6**, **Fysisk databehandling**, är det dags att bygga något speciellt: en egen stop-motion-animeringsstudio.

Stop-motion-animering görs genom att man tar massor av bilder av stillobjekt, såsom modellbilar eller actionfigurer, och flyttar objekten en aning mellan varje bild. Även om objekten aldrig rör sig på någon av bilderna så ser det ut som om de rör sig så snabbt eller så långsamt som du vill om du visar dem en efter en tillräckligt snabbt!

För det här projektet behöver du en tryckknappsströmbrytare, ett kopplingsdäck, en hane-till-hane-kopplingstråd (M2M) och ett par hane-till-hona-kopplingstrådar (M2F). Om du inte har ett kopplingsdäck kan du ansluta kopplaren med hjälp av hona-till-hona-kopplingstrådar (F2F) istället, men det kommer att göra det svårare att trycka. Om du behöver läsa på om någon av dessa komponenter igen ska du se **Kapitel 6, Fysisk databehandling med Scratch och Python**. Du behöver även objekt att animera: dessa kan vara allt från en lerklump till en leksaksbil eller en actionfigur.

Börja med att skapa kretsen: lägg till tryckknappen på kopplingsdäcket och anslut sedan kopplingsdäckets jordskena till ett jordstift på Raspberry Pi (markerat GND på **Bild 8-9**) med en hane-till-hona-kopplingstråd. Använd en hane-till-hane-kopplingstråd för att ansluta ett ben på kopplaren till jordskenan på kopplingsdäcket och sedan en hane-till-hona-kopplingstråd för att ansluta det andra benet på kopplaren till GPIO-stift 2 (markerat GP2 på **Bild 8-9**).




▲ **Bild 8-9:** Kopplingsschema för att ansluta en tryckknapp till GPIO-stiften

Skapa ett nytt projekt och spara det som **Stop-motion**. Börja med att importera och ställa in de bibliotek du behöver för att använda kameran och GPIO-porten:

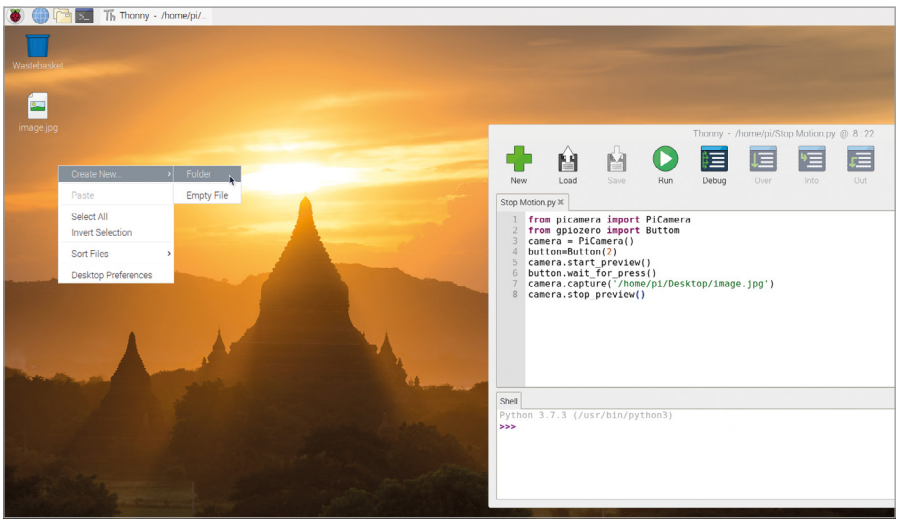
```
from picamera import PiCamera
from gpiozero import Button
camera = PiCamera()
button = Button(2)
```

Skriv sedan följande:

```
camera.start_preview()
button.wait_for_press()
camera.capture('/home/pi/Desktop/image.jpg')
camera.stop_preview()
```

Klicka på Run så visas en förhandsgranskning av det som kameran är riktad mot. Förhandsgranskningen stannar kvar på skärmen tills du trycker på tryckknappsströmbrytaren: tryck på den nu så stängs förhandsgranskningen när programmet sparar en bild till skrivbordet. Leta upp bilden, som kallas **image.jpg**, och dubbelklicka för att öppna den och bekräfta att programmet fungerar.

Stop-motion-animering innebär att du skapar massor av stillbilder för att ge intryck av rörelse när de alla sätts samman. Om du har alla dessa enskilda bilder på skrivbordet blir det rörigt, så du behöver en mapp att lagra allihop i. Högerklicka var som helst på skrivbordet där det inte redan finns en fil eller en ikon och välj sedan Create New och Folder (**Bild 8-10**). Döp mappen till **animation**, med endast små bokstäver, och klicka sedan på OK-knappen.




▲ **Bild 8-10:** Skapa en ny mapp för de tagna bilderna

Det är inte så bra om du behöver starta om programmet varje gång du tar en bild till animeringen, så du måste ändra det så att det körs i en loop. Till skillnad från de tidigare looparna du har skapat behöver den här ett sätt att stängas med stil – annars kan du inte längre se skrivbordet om du stoppar programmet medan kamerans förhandsgranskning visas! För att kunna göra detta måste du använda två speciella instruktioner: `try` och `except`.

Börja med att radera allt efter **camera.start_preview(),** och skriv sedan:

```
frame = 1
```

Detta skapar en ny variabel, **frame**, som programmet kommer att använda för att lagra det aktuella bildnumret. Du kommer att använda denna variabel inom kort för att säkerställa att du sparar en ny fil varje gång; utan den kommer du bara att spara över din första bild varje gång du trycker på knappen!

Ställ sedan in loopen genom att skriva:

```
while True:
    try:
```

Den nya instruktionen **try** meddelar Python att det ska köra den kod som finns inuti – vilket kommer att vara koden för att ta bilder. Skriv:

```
button.wait_for_press()
camera.capture('/home/pi/Desktop/animation/frame%03d.jpg' % frame)
frame += 1
```

Det finns ett par smarta knep i dessa tre rader med kod. Det första finns i filnamnet: med **%03d** meddelar du Python att ta ett tal och lägga till så många nollor framför som det krävs för att göra det tre siffror långt. Således blir ”1” till ”001”, ”2” blir ”002” och ”10” blir ”010”. Du behöver detta i programmet för att filerna ska hamna i rätt ordning och för att säkerställa att du inte skriver över en fil som du redan har sparat.

**% frame** i slutet av raden meddelar Python att det ska använda bildvariabelns tal i filnamnet. Den sista raden – **frame += 1** – ökar bildvariabeln med 1 för att säkerställa att varje fil blir unik. Första gången du trycker på knappen kommer **frame** att öka från 1 till 2, nästa gång från 2 till 3 och så vidare.

För tillfället kommer koden dock inte att avslutas ordentligt när du har slutat ta bilder. För att det ska ske behöver du ett **except** för **try**. Skriv följande, och kom ihåg att ta bort en nivå av indrag på första raden så att Python vet att det inte ingår i **try**-avsnittet:

```
    except KeyboardInterrupt:
        camera.stop_preview()
        break
```

Det färdiga programmet ska nu se ut så här:

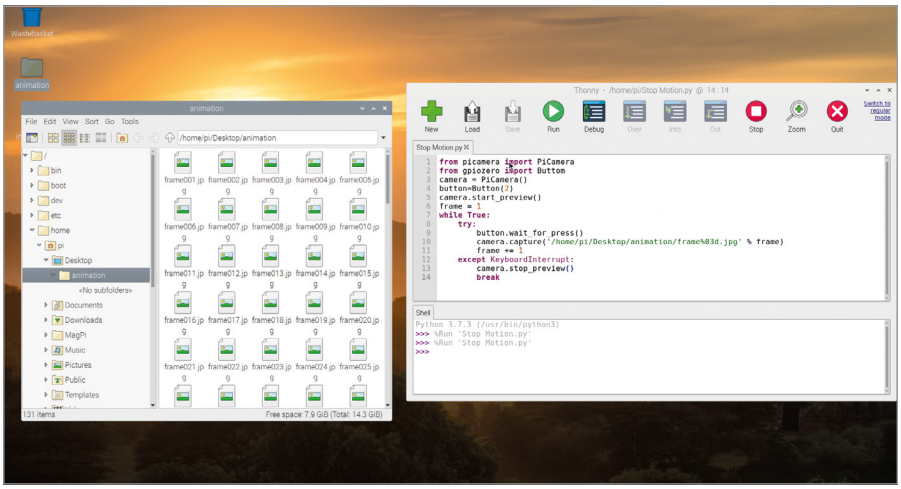
```
from picamera import PiCamera
from time import sleep
from gpiozero import Button
camera = PiCamera()
button = Button(2)
camera.start_preview()
frame = 1
while True:
    try:
        button.wait_for_press()
        camera.capture('/home/pi/Desktop/animation/frame%03d.jpg'
% frame)
        frame += 1
    except KeyboardInterrupt:
        camera.stop_preview()
        break
```

Försök att klicka på Run, men istället för att trycka på knappen, trycker du på **CTRL**- och **C**-tangenterna på tangentbordet. Du behöver inte trycka på båda knapparna samtidigt: håll bara ner **CTRL**, tryck och släpp **C** och släpp sedan **CTRL**. Dessa två tangenter fungerar som ett avbrott och ber Python att avsluta vad det håller på med. Utan raden `except KeyboardInterrupt:` skulle Python omedelbart avslutas och låta kamerans förhandsgranskning blockera skärmen. Med den raden på plats kör Python dock den kod som finns inuti – i detta fall kod som säger att det ska stoppa kamerans förhandsgranskning och avsluta med en ren skärm.

Nu är du redo att börja ta bilder till stop-motion-animeringen! Placera Camera Module eller HQ Camera där den kan se objekten du ska animera och se till att den inte rör sig – om kameran rör sig förstörs effekten. Placera objekten i startpositionerna och klicka sedan på Run för att starta programmet. Kontrollera att allt ser bra ut i förhandsgranskningen och tryck på tryckknappen för att ta den första bilden.

Flytta objekten en aning – ju mindre du flyttar dem mellan bilder, desto mjukare blir den färdiga animeringen – och tryck på tryckknappen igen för att ta en till bild. Fortsätt göra detta tills animeringen är klar. Ju fler bilder du tar, desto längre blir animeringen.

När du är klar trycker du på **CTRL+C** för att stänga programmet och dubbelklickar sedan på **animation**-mappen på skrivbordet för att se bilderna du har tagit (**Bild 8-11**, på nästa sida). Dubbelklicka på valfri bild för att öppna den och se den i närmare detalj!




▲ **Bild 8-11:** De tagna bilderna i mappen

För närvarande har du dock bara en mapp full med stillbilder. För att skapa en animering måste du förvandla dem till en video. För att göra detta klickar du på hallonikonen för att läsa in menyn, väljer Tillbehör och klickar på LXTerminal. Detta öppnar ett *kommandoradsgränssnitt*, vilket diskuteras i närmare detalj i **Bilaga C**, som låter dig skriva kommandon till Raspberry Pi. När LXTerminal har lästs in börjar du med att byta till den mapp du skapade genom att skriva:

```
cd Desktop/animation
```

Det är viktigt att ”D” i ”Desktop” är en versal. Raspberry Pi OS är så kallat *skiftlägeskänsligt*, vilket betyder att ett kommando eller mappnamn inte fungerar om du inte skriver det exakt som det ursprungligen skrevs! När du har bytt mapp skriver du följande:

```
ffmpeg -i ram% 03d.jpg -r 10 animation.h264
```

Detta använder ett program som kallas **ffmpeg** för att ta stillbilderna i mappen och konvertera dem till en video som kallas **animation.h264**. (Obs! Om ffmpeg inte finns tillgängligt kan du installera det med `sudo apt-get install ffmpeg`.) Beroende på hur många stillbilder du tog kan processen ta några minuter. Du vet att den är klar när du ser att LXTerminal-prompten visas igen.

För att spela upp videon letar du upp filen **animation.h264** i **animation**-mappen och dubbelklickar på den för att öppna den. Alternativt kan du spela upp den från LXTerminal genom att skriva följande:

```
omxplayer animation.h264
```

När videon har lästs in ser du att stop-motion-animeringen kommer till liv. Grattis: du har förvandlat Raspberry Pi till en kraftfull animeringsstudio!

Om animeringen rör sig för snabbt eller för långsamt kan du ändra **`-r 10`**-delen i kommandot **`ffmpeg`** till ett lägre eller högre tal: detta är bildhastigheten, eller hur många stillbilder som utgör en sekund av videon. Ett lägre tal gör att animeringen går långsammare men ser mindre smidig ut. Ett högre tal ser mjukare ut men kommer att få animeringen att gå snabbare.

Om du vill spara videon ska du dra och släppa den från skrivbordet till Videos-mappen, annars kommer du att skriva över filen nästa gång du kör programmet!

## Avancerade kamerainställningar

Om du behöver mer kontroll över Raspberry Pi Camera Module eller HQ Camera kan du använda Pythons picamera-bibliotek för att få tillgång olika inställningar. Dessa inställningar, tillsammans med dess standardvärden, beskrivs nedan för inkludering i dina egna program.

**`camera.awb_mode = 'auto'`**

Denna ställer in kamerans automatiska vitbalansläge och kan ställas in till något av följande lägen: **`off`**, **`auto`**, **`sunlight`**, **`cloudy`**, **`shade`**, **`tungsten`**, **`fluorescent`**, **`incandescent`**, **`flash`** eller **`horizon`**. Om du tycker att bilderna och videorna ser lite blå eller gula ut kan du försöka med ett annat läge.

**`camera.brightness = 50`**

Denna ställer in kamerabildens ljusstyrka, från mörkast vid 0 till ljusast vid 100.

**`camera.color_effects = None`**

Denna ändrar den färgeffekt som för närvarande används av kameran. Normalt bör denna inställning lämnas som den är, men om du anger två tal kan du ändra hur kameran registrerar färg: försök med **`(128, 128)`** för att skapa en svartvit bild.

**`camera.contrast = 0`**

Denna ställer in bildens kontrast. Ett högre tal får saker att se mer dramatiska och starka ut och ett lägre tal får saker att se blekare ut. Du kan använda valfritt tal mellan -100 för minimal kontrast och 100 för maximal kontrast.

**`camera.crop = (0.0, 0.0, 1.0, 1.0)`**

Denna gör att du kan beskära bilden, klippa ut delar från sidorna och överst för att bara ta den del av bilden du behöver. Talen representerar X-koordinat, Y-koordinat, bredd och höjd och tar som standard hela bilden. Försök att minska de två sista talen – 0,5 och 0,5 är en bra utgångspunkt – för att se vilken effekt denna inställning har.

**`camera.exposure_compensation = 0`**

Denna ställer in kamerans *exponeringskompensation*, så att du manuellt kan styra hur mycket ljus som fångas för varje bild. Till skillnad från när du ändrar inställningen för ljusstyrka styr detta faktiskt själva kameran. Giltiga värden varierar från -25 för en mycket mörk bild till 25 för en mycket ljus bild.

**`camera.awb_mode = 'auto'`**

Denna ställer in *exponeringsläget* eller den logik som Camera Module/HQ Camera använder för att bestämma hur en bild ska exponeras. Möjliga lägen är: **`off`**, **`auto`**, **`night`**, **`backlight`**, **`spotlight`**, **`sports`**, **`snow`**, **`beach`**, **`verylong`**, **`fixedfps`**, **`antishake`** och **`fireworks`**.

**`camera.framerate = 30`**

Denna ställer in antalet bilder som tagits för att skapa en video per sekund eller *bildhastigheten*. En högre bildhastighet skapar en mjukare video men tar upp mer lagringsutrymme. Högre bildhastigheter kräver en lägre upplösning som du kan ställa in via **`camera.resolution`**.

**`camera.hflip = False`**

Denna vänder kamerabilden över den horisontella axeln, eller Y-axeln, när den är inställd på **`True`**.

**`camera.image_effect = 'none'`**

Denna tillämpar en av ett antal bildeffekter på videoströmmen, som kommer att synas i förhandsgranskningen samt de sparade bilderna och videoklippen. Möjliga effekter är: **`blur`**, **`cartoon`**, **`colorbalance`**, **`colorpoint`**, **`colorswap`**, **`deinterlace1`**, **`deinterlace2`**, **`denoise`**, **`emboss`**, **`film`**, **`gpen`**, **`hatch`**, **`negative`**, **`none`**, **`oilpaint`**, **`pastel`**, **`posterise`**, **`saturation`**, **`sketch`**, **`solarize`**, **`washedout`** och **`watercolor`**.

**`camera.ISO = 0`**

Denna ändrar kamerans ISO-inställning, vilket påverkar hur känslig den är för ljus. Som standard justerar kameran detta automatiskt beroende på tillgängligt ljus. Du kan ställa in ISO själv med hjälp av något av följande värden: 100, 200, 320, 400, 500, 640, 800. Ju högre ISO, desto bättre kommer kameran att fungera i miljöer med svagt ljus, men desto kornigare blir bilden eller videon.

**`camera.meter_mode = 'average'`**

Denna styr hur kameran fastställer mängden tillgängligt ljus när den ställer in exponeringen. Standardinställningen beräknar genomsnittet av den tillgängliga mängden ljus genom hela bilden. Andra möjliga lägen är **`backlit`**, **`matrix`** och **`spot`**.

`camera.resolution = (1920, 1080)`
Denna ställer in upplösningen för den tagna bilden eller videon, representerad av två tal för bredd och höjd. Lägre upplösningar tar upp mindre lagringsutrymme och låter dig använda en högre bildhastighet. Högre upplösningar har bättre kvalitet men tar upp mer lagringsutrymme.

`camera.rotation = 0`
Denna styr bildens rotation, från 0 grader till 90, 180 och 270 grader. Använd denna om du inte kan placera kameran så att bandkabeln kommer ut ur botten.

`camera.saturation = 0`
Denna styr mättnaden på bilden eller hur levande färgerna är. Möjliga värden varierar från -100 till 100.

`camera.sharpness = 0`
Denna styr bildens skärpa. Möjliga värden varierar från -100 till 100.

`camera.shutter_speed = 0`
Denna styr hur snabbt slutaren öppnas och stängs när du tar bilder och videor. Du kan ställa in slutarhastigheten manuellt i mikrosekunder. Långsammare slutarhastigheter fungerar bättre i svagt ljus och snabbare slutarhastigheter fungerar bättre i starkare ljus. Denna bör normalt lämnas som automatisk, standardinställningen.

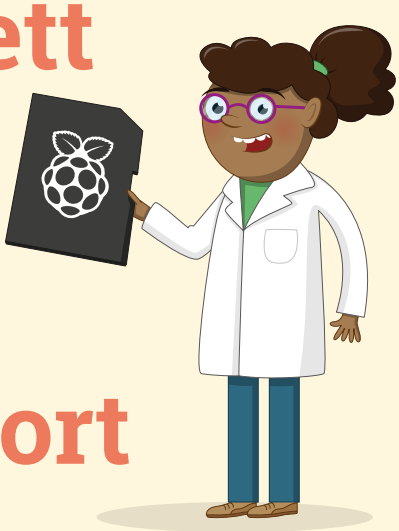
`camera.vflip = False`
Denna vänder kamerabilden över den vertikala axeln, eller X-axeln, när den är inställd på `True`.

`camera.video_stabilization = False`
När den är inställd på `True` aktiveras videostabilisering. Du behöver bara denna om Camera Module eller HQ Camera rör sig medan du spelar in, till exempel om den är ansluten till en robot eller bärs, för att minska skakningen i den inspelade videon.

Mer information om dessa inställningar, liksom ytterligare inställningar som inte dokumenteras här, finns på **picamera.readthedocs.io**.

Bilaga A

# Installera ett operativ-system till ett microSD-kort



Du kan köpa microSD-kort med förinstallerat NOOBS (New Out of the Box Software) från alla bra Raspberry Pi-återförsäljare, så att du enkelt kan installera Raspberry Pi OS (tidigare kallad Raspbian) på Raspberry Pi. Alternativt kan du använda följande instruktioner för att använda Raspberry Pi Imager för att installera ett operativsystem manuellt till ett eget tomt (eller återanvänt) microSD-kort.

> **VARNING!**
>
> Om du har köpt ett microSD-kort som redan har NOOBS förinstallerat behöver du inte göra något annat än att ansluta det till Raspberry Pi. Dessa instruktioner är för tomma microSD-kort eller för tidigare använda kort som du vill installera ett nytt operativsystem på. Om du tillämpar dessa instruktioner på ett microSD-kort med filer på kommer du att förlora dessa filer, så se till att du säkerhetskopierar först!

## Ladda ner Raspberry Pi Imager

Raspberry Pi OS är baserat på Debian och är det officiella operativsystemet för Raspberry Pi. Det enklaste sättet att installera Raspberry Pi OS på ett microSD-kort för Raspberry Pi är att använda Raspberry Pi Imager-verktyget, som går att ladda ner från **rpf.io/downloads**. Observera att den här metoden ersätter installationen av operativsystemet via NOOBS, även om det fortfarande finns tillgängligt från samma nedladdningssida.

Raspberry Pi Imager-applikationen finns tillgänglig för Windows-, macOS- och Ubuntu Linux-datorer, så välj den aktuella versionen för ditt system.

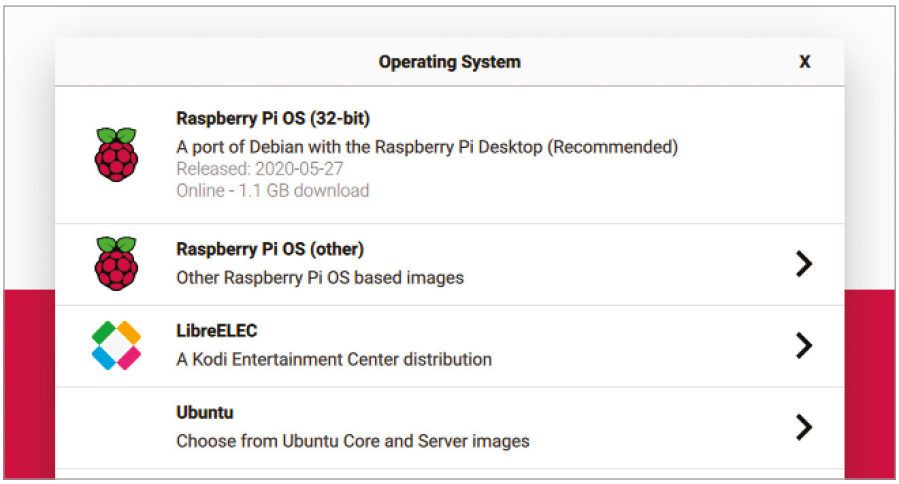
På macOS dubbelklickar du på den nedladdade DMG-filen. Du kan behöva ändra inställningen för säkerhet och integritet för att tillåta appar som laddas ner från ”App Store och identifierade utvecklare” att köra filen. Sedan kan du dra Raspberry Pi Imager-ikonen till Applications-mappen.

På en Windows-dator dubbelklickar du på den nedladdade EXE-filen. Välj ”Yes”-knappen när du blir ombedd så att filen kan köras. Klicka sedan på knappen ”Install” för att starta installationen.

## Skriv operativsystemet till microSD-kortet

Anslut microSD-kortet till din PC eller Mac-dator: du behöver en USB-adapter för microSD-kort såvida inte datorn har en inbyggd kortläsare. Observera att kortet inte behöver förformateras.

Starta Raspberry Pi Imager-applikationen. Klicka på knappen ”Choose OS” för att välja vilket operativsystem du vill installera. Det översta alternativet är det standardmässiga Raspberry Pi OS – om du föredrar den nedbantade Lite-versionen eller Full-versionen (med all rekommenderad programvara förinstallerad) väljer du ”Raspberry Pi OS (other)”. Det finns också alternativ för att installera LibreELEC (välj version för din Raspberry Pi-modell) och Ubuntu Core eller Server.



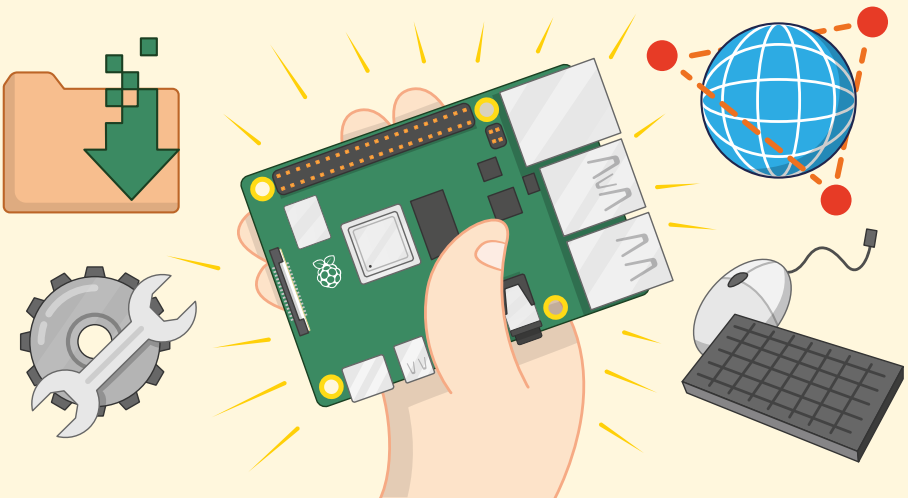

Observera: Om du vill installera ett annat operativsystem, till exempel Lakka, laddar du bara ner dess bildfil från den aktuella webbplatsen och väljer sedan alternativet ”Use Custom” i Raspberry Pi Imager.

När ett operativsystem är markerat klickar du på knappen ”Choose SD card” och väljer ditt microSD-kort (vanligtvis finns det bara ett alternativ).

Slutligen klickar du på ”Write”-knappen och väntar medan verktyget skriver det valda operativsystemet till kortet och sedan verifierar det. När du är klar kan du ta bort microSD-kortet. Du kan sedan sätta in kortet i Raspberry Pi och starta upp den i operativsystemet du just installerat.

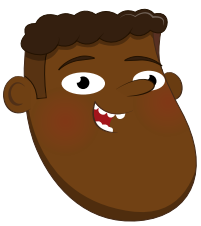
## Bilaga B

# Installera och avinstallera programvara



Raspberry Pi OS levereras med ett urval av populära programvarupaket som är handplockade av Raspberry Pi Foundation, men dessa är inte de enda paket som fungerar på en Raspberry Pi. Med hjälp av följande instruktioner kan du bläddra bland ytterligare programvara, installera den och avinstallera den igen och på så vis utöka prestandan i Raspberry Pi.

Instruktionerna i denna bilaga är ett tillägg till instruktionerna i **Kapitel 3, Använda Raspberry Pi**, som förklarar hur du använder verktyget Recommended Software. Om du inte har läst det kapitlet redan, gör det innan du använder metoderna som beskrivs i denna bilaga.

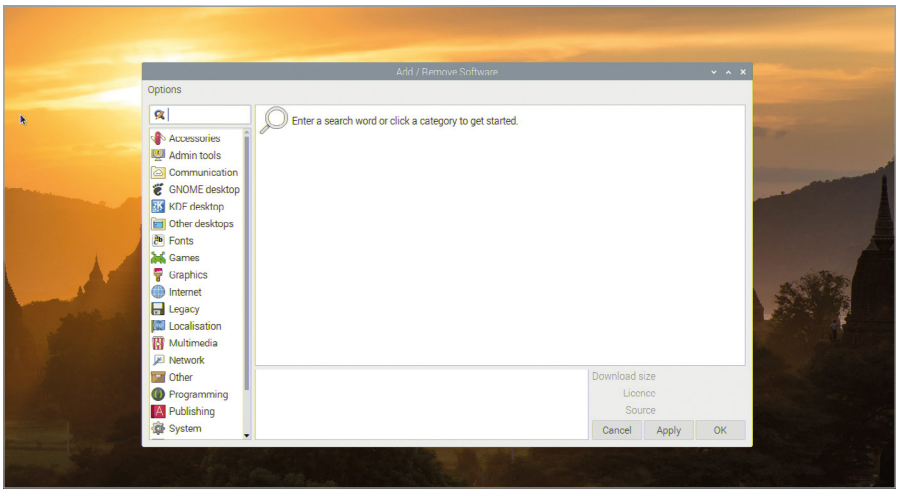


### KORTKAPACITET

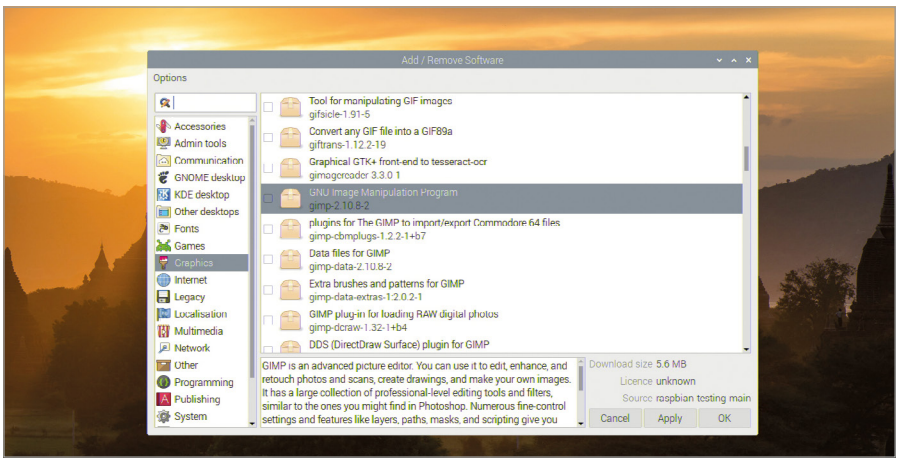
Om du lägger till mer programvara på Raspberry Pi tar det upp plats på microSD-kortet. Ett kort på 16 GB eller större låter dig installera mer programvara. För att kontrollera om kortet du tänker använda är kompatibelt med Raspberry Pi, besök **rpf.io/sdcardlist**.

## Bläddra bland tillgänglig programvara

För att se och söka i listan över tillgängliga programvarupaket för Raspberry Pi OS, med så kallade *programvaruförråd*, klickar du på hallonikonen för att läsa in menyn. Välj kategorin Inställningar och klicka sedan på Add/Remove Software. Efter några sekunder visas verktygets fönster.



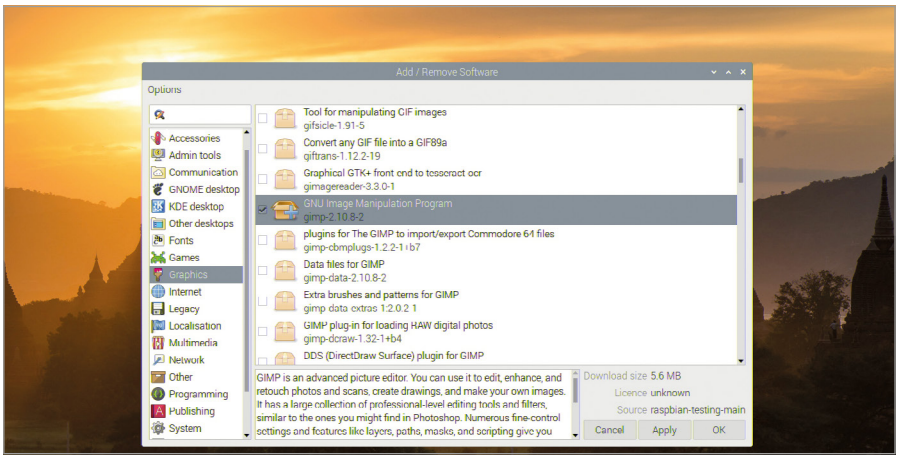

Den vänstra sidan av fönstret Add/Remove Software innehåller en lista med kategorier – samma kategorier som du hittar i huvudmenyn när du klickar på hallonikonen. Om du klickar på en av dem visas en lista över programvara som är tillgänglig i den kategorin. Du kan också ange ett sökord i rutan längst upp till vänster i fönstret, till exempel ”textredigerare” eller ”spel”, och se en lista med matchande programvarupaket från alla kategorier. Om du klickar på valfritt paket visas ytterligare information i utrymmet längst ner i fönstret.



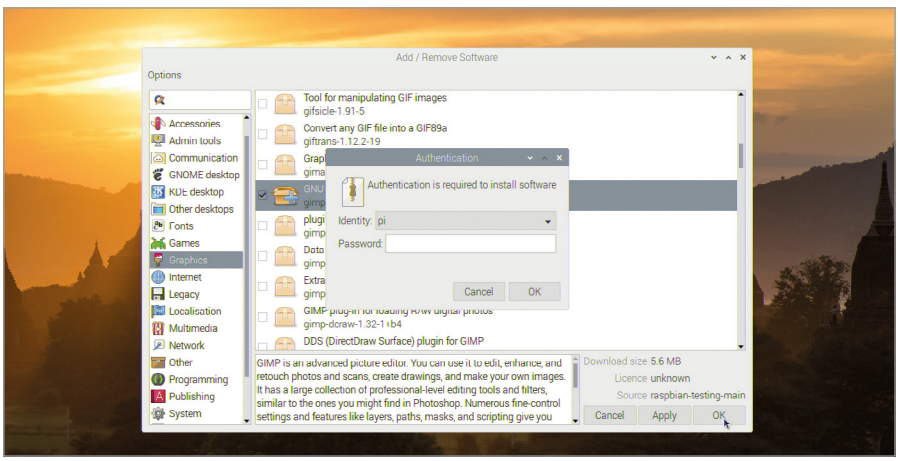
Om kategorin du har valt har många tillgängliga programvarupaket kan det ta lite tid för verktyget Add/Remove Software att gå igenom listan.

## Installera programvara

Markera rutan bredvid det paket du vill installera genom att klicka på den. Du kan installera mer än ett paket samtidigt: fortsätt bara klicka för att lägga till fler paket. Ikonen bredvid paketet ändras till en öppen ruta med en ”+”-symbol för att bekräfta att den kommer att installeras.

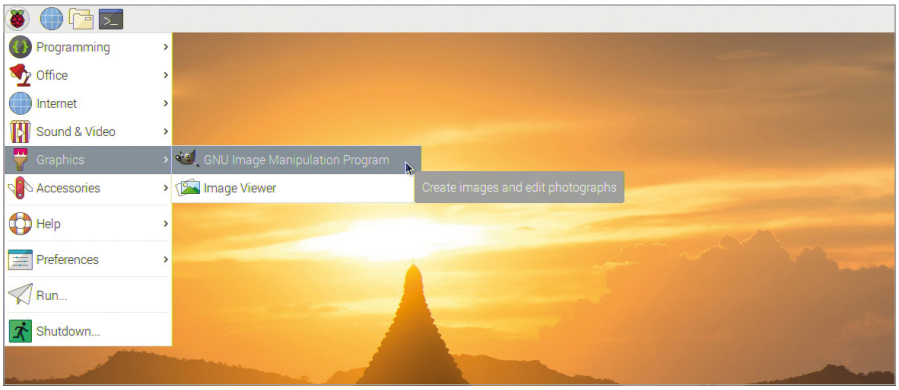



När du är nöjd med dina val klickar du antingen på OK- eller Apply-knappen. Den enda skillnaden är att OK stänger verktyget Add/Remove Software när din programvara är installerad medan Apply-knappen lämnar den öppen. Du blir ombedd att ange ditt lösenord för att bekräfta din identitet – du vill trots allt inte att någon ska kunna lägga till eller ta bort programvara från din Raspberry Pi!



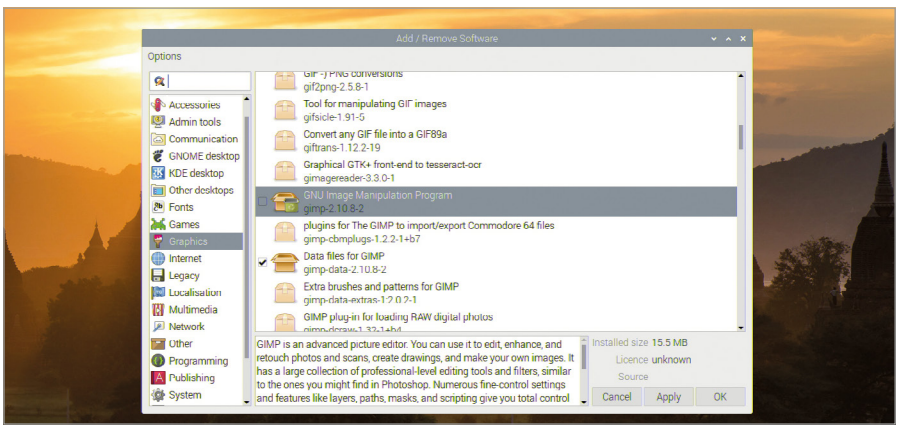
Du kanske upptäcker att när du installerar ett paket installeras andra paket samtidigt. Dessa kallas *beroenden* och är paket som programvaran du valde att installera behöver för att fungera – till exempel ljudeffektpaket för ett spel eller en databas som hör till en webbserver.

När programvaran är installerad bör du kunna hitta den genom att klicka på hallonikonen för att läsa in menyn och leta upp programvarupaketets kategori. Tänk på att menykategorin inte alltid är densamma som kategorin från verktyget Add/Remove Software, medan en del programvaror inte har en menypost överhuvudtaget. Denna programvara kallas *programvara med kommandorad* och måste köras i LXTerminal. För mer information om kommandoraden och LXTerminal, läs **Bilaga C, Kommandoradsgränssnittet**.




## Avinstallera programvara

Om du vill ta bort eller *avinstallera* ett paket letar du upp det i listan över paket – sökfunktionen är användbar här – och avmarkerar rutan intill genom att klicka på den. Du kan avinstallera mer än ett paket samtidigt: fortsätt bara klicka för att ta bort fler paket. Ikonen bredvid paketet ändras till en öppen ruta intill en liten papperskorg för att bekräfta att den kommer att avinstalleras.



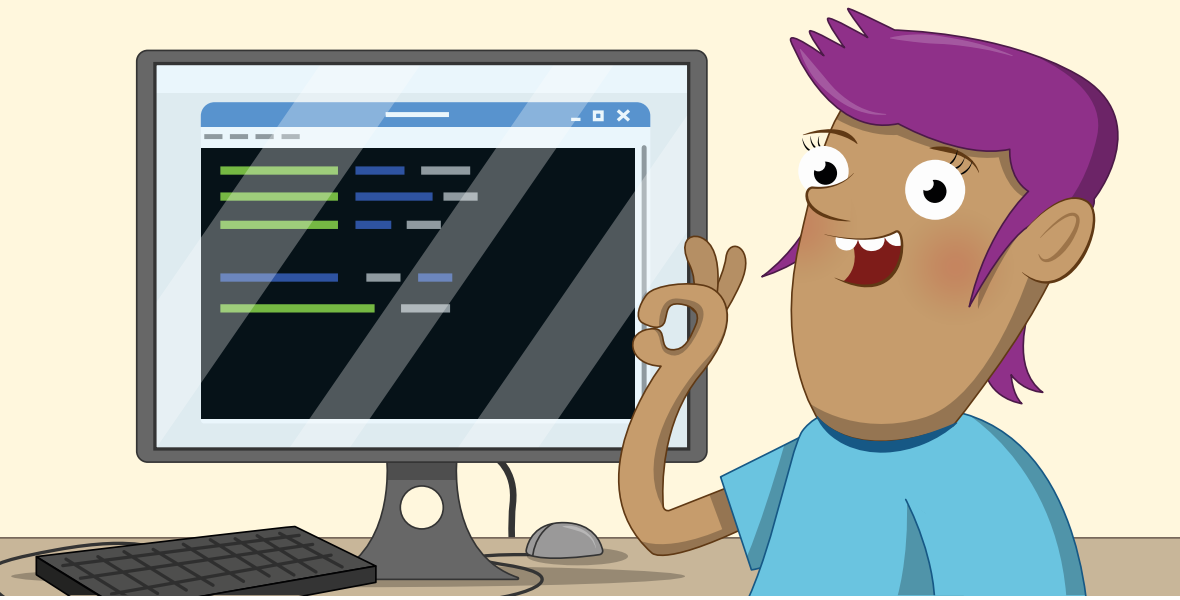
Precis som tidigare kan du klicka på OK- eller Apply-knappen för att börja avinstallera de valda programvarupaketen. Du blir ombedd att bekräfta ditt lösenord, såvida du inte har gjort det under de senaste minuterna, och du kan också bli ombedd att bekräfta att du även vill ta bort eventuella beroenden som hör till programvarupaketet. När avinstallationen är klar försvinner programvaran från hallonikonmenyn, men filer som du skapade med programvaran – till exempel bilder för ett grafikpaket eller sparade filer för ett spel – tas inte bort.

### VARNING!

All programvara som installerats på Raspberry Pi OS visas i Add/Remove Software, inklusive programvara som krävs för att Raspberry Pi ska fungera. Det är möjligt att ta bort så många paket att skrivbordet inte längre läses in. För att undvika detta bör du inte avinstallera saker om du inte är säker på att du inte längre behöver dem. Om du redan har gjort det installerar du om Raspberry Pi OS enligt instruktionerna i **Kapitel 2, Kom igång med Raspberry Pi** eller installerar om operativsystemet med hjälp av **Bilaga A**.

## Bilaga C

# Kommandorad-gränssnittet



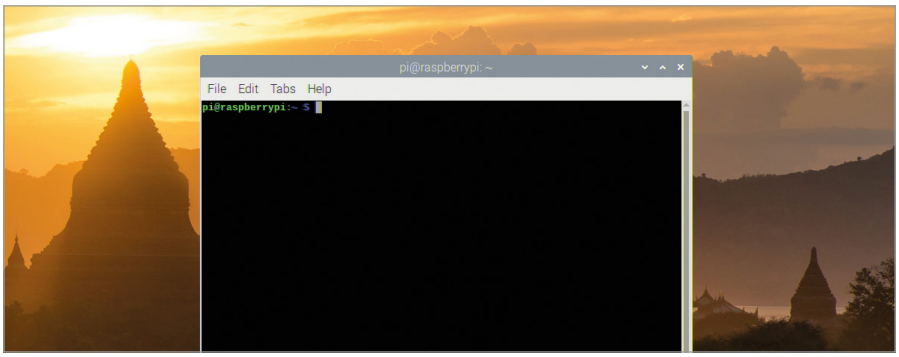
Även om du kan hantera de flesta programvaror på en Raspberry Pi via skrivbordet, kan vissa endast nås med ett textbaserat läge som kallas *kommandoradsgränssnittet (CLI)* i ett program som heter LXTerminal. De flesta användare kommer aldrig att behöva använda CLI, men för dem som vill lära sig mer erbjuder denna bilaga en grundläggande introduktion.

### MER INFORMATION

Denna bilaga är inte utformad för att vara en fullständig guide till Linux-kommandoradsgränssnittet. För en mer detaljerad titt på hur du använder CLI, besök **rpf.io/terminal** i en webbläsare.

## Läsa in LXTerminal

CLI nås via LXTerminal, ett programvarupaket som läser in vad som tekniskt sett kallas en *virtual teletype (VTY) terminal*, ett namn som kommer från datorns tidiga dagar när användare utfärdade kommandon via en stor elektromekanisk skrivmaskin snarare än ett tangentbord och en bildskärm. Om du vill läsa in LXTerminal-paketet klickar du på hallonikonen för att läsa in menyn, väljer kategorin Tillbehör och klickar sedan på LXTerminal.




LXTerminal-fönstret är ett fönster som du kan dra runt på skrivbordet, ändra storlek på, maximera och dölja precis som alla andra fönster. Du kan också göra texten i fönstret större om den är svår att se, eller mindre om du vill få plats med mer i fönstret: klicka på Edit-menyn och välj Zoom In respektive Zoom Out eller tryck på **CTRL**-tangenten på tangentbordet följt av **+** eller **-**.

## Prompten

Det första du ser i LXTerminal är *prompten*, som väntar på dina instruktioner. Prompten på en Raspberry Pi som kör Raspberry Pi OS ser ut så här:

```
pi@raspberrypi:~ $
```

Den första delen av prompten, **`pi`**, är ditt användarnamn. Den andra delen, efter **`@`**, är värdnamnet på den dator du använder, som är **`raspberrypi`** som standard. Efter "**`:`**" kommer en tilde, **`~`**, som är ett kortfattat sätt att hänvisa till din hemkatalog och representerar din *nuvarande arbetskatalog (CWD)*. Slutligen indikerar **`$`**-symbolen att din användare är en *användare utan behörighet*, vilket innebär att du behöver ett lösenord för att utföra uppgifter som att lägga till eller ta bort programvara.

## Kringgå detta

Testa att skriva in följande och tryck sedan på **ENTER**-tangenten:

```
cd Desktop
```

Du kommer att se att prompten ändras till:

```
pi@raspberrypi:~/Desktop $
```

Det visar att din nuvarande arbetskatalog har ändrats: du var i din hemkatalog tidigare, indikerad av ~-symbolen, och nu är du i underkatalogen **Desktop** under din hemkatalog. För att komma dit använde du **cd**-kommandot – *change directory*.

### RÄTT SKIFTLÄGE

Raspberry Pi OS:s kommandoradsgränssnitt är skiftlägeskänsligt, vilket betyder att det har betydelse när kommandon eller namn har stora och små bokstäver. Om du fick ett ”no such file or directory”-meddelande när du försökte byta katalog ska du kontrollera att du hade ett stort D i början av Desktop.

Det finns fyra sätt att gå tillbaka till din hemkatalog: försök med ett åt gången och byt tillbaka till underkatalogen **Desktop** varje gång. Det första är:

```
cd ..
```

**..**-symbolerna är en annan genväg, den här gången för ”katalogen ovanför den här”, även känd som *överordnad katalog*. Eftersom katalogen ovanför **Desktop** är din hemkatalog, tar den dig tillbaka dit. Byt tillbaka till underkatalogen **Desktop** och försök med det andra sättet:

```
cd ~
```

Detta använder ~-symbolen och betyder bokstavligen ”byt till min hemkatalog”. Till skillnad från **cd ..**, som endast tar dig till överordnad katalog för den katalog du befinner dig i, fungerar det här kommandot var som helst. Men det finns ett enklare sätt:

```
cd
```

**cd** går som standard tillbaka till din hemkatalog utan att få namnet på en katalog. Det sista sättet att komma tillbaka till din hemkatalog är att skriva:

```
cd /home/pi
```

Detta använder det som kallas en *absolut sökväg*, som fungerar oavsett vilken katalog du för närvarande arbetar i. Så precis som **cd** i sig självt eller **cd ~** tar denna dig tillbaka till din hemkatalog var du än är, men till skillnad från de andra metoderna behöver du veta ditt användarnamn.

## Hantera filer

Om du vill öva på att arbeta med filer byter du till **Desktop**-katalogen och skriver in följande:

```
touch Test
```

En fil som heter **Test** visas på skrivbordet. **`touch`**-kommandot används normalt för att uppdatera datum- och tidsinformation för en fil, men om filen – som i det här fallet – inte finns, skapar det den.

Försök med följande:

```
cp Test Test2
```

En annan fil som heter **Test2** visas på skrivbordet. Detta är en *kopia* av originalfilen, identisk på alla sätt. Radera den genom att skriva:

```
rm Test2
```

Detta *tar bort* filen och du ser den försvinna.

> **VARNING!**
>
> Till skillnad från när du raderar filer med den grafiska Filhanteraren, som lagrar dem i Papperskorgen så du kan hämta tillbaka dem senare, är filer som raderas med `rm` borta för alltid. Var noggrann när du skriver!

Försök sedan med:

```
mv Test Test2
```

Det här kommandot *flyttar* filen och du ser att din ursprungliga **Test**-fil försvinner och ersätts av **Test2**. Flyttkommandot, **`mv`**, kan användas så här för att byta namn på filer.

När du inte är på skrivbordet måste du ändå kunna se vilka filer som finns i en katalog. Skriv:

```
ls
```

Detta kommando *listar* innehållet i den aktuella katalogen eller någon annan katalog du ger den. För mer information, inklusive att lista dolda filer och rapportera filstorlekar, kan du testa att lägga till några omkopplare:

```
ls -larth
```

Dessa omkopplare styr **ls**-kommandot: **l** kopplar utdatan till en lång vertikal lista; **a** säger till den att visa alla filer och kataloger, inklusive sådana som normalt skulle vara dolda; **r** ändrar den normala sorteringsordningen; **t** sorterar efter tid för ändring, vilket i kombination med **r** ger dig de äldsta filerna högst upp i listan och de nyaste filerna längst ner; och **h** använder läsbara filstorlekar, vilket gör listan lättare att förstå.

## Köra program

Vissa program kan bara köras på kommandoraden, medan andra har både grafiska gränssnitt och kommandoradsgränssnitt. Ett exempel på det senare är Raspberry Pi Software Configuration Tool, som du vanligtvis skulle läsa in från hallonikonmenyn.

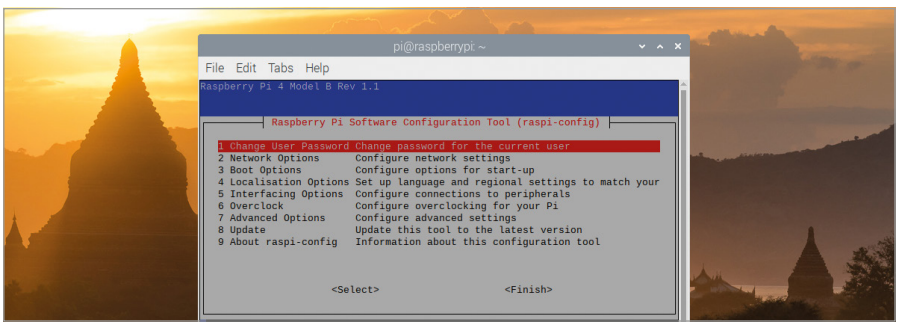
Skriv:

```
raspi-config
```

Du får ett felmeddelande om att programvaran bara kan köras som *root*, systemadministratörskontot på Raspberry Pi. Det kommer också att berätta hur du gör det genom att skriva:

```
sudo raspi-config
```

**sudo**-delen av kommandot betyder *switch-user do* och säger till Raspberry Pi OS att köra kommandot som root-användare.




Du behöver bara använda **sudo** när ett program behöver förhöjda *behörigheter*, till exempel när det installerar eller avinstallerar programvara eller justerar systeminställningar. Ett spel bör till exempel aldrig köras med **sudo**.

Tryck på **TAB**-tangenten två gånger för att välja Finish och tryck på **ENTER** för att avsluta Raspberry Pi Software Configuration Tool och återgå till kommandoradsgränssnittet.
Skriv slutligen:

```
exit
```

Detta avslutar kommandoradsgränssnittssessionen och stänger LXTerminal-appen.

## Använda en TTY

LXTerminal-programmet är inte det enda sättet att använda kommandoradsgränssnittet: du kan också växla till en av flera terminaler som redan körs, en *teletype* eller *TTY*. Håll in **CTRL**- och **ALT**-tangenterna på tangentbordet och tryck på **F2**-tangenten för att växla till ”tty2”.


Du måste logga in igen med ditt användarnamn och lösenord, varefter du kan använda kommandoradsgränssnittet precis som i LXTerminal. En TTY är användbar när huvudgränssnittet på skrivbordet, oavsett anledning, inte fungerar.

Om du vill växla bort från en TTY håller du **CTRL+ALT** intryckt och trycker sedan på **F7**: skrivbordet visas igen. Tryck på **CTRL+ALT+F2** igen så växlar du tillbaka till ”tty2” – och allt du körde i det kommer att finnas kvar.

Innan du byter igen skriver du:

```
exit
```

Tryck sedan på **CTRL+ALT+F7** för att gå tillbaka till skrivbordet. Anledningen till att du ska avsluta innan du byter från en TTY är att alla som har tillgång till tangentbordet kan byta till en TTY och om du fortfarande är inloggad kommer de att få åtkomst till ditt konto även om de inte kan ditt lösenord!

Grattis: du har tagit dina första steg mot att bemästra Raspberry Pi OS-kommandoradsgränssnittet!

## Bilaga D

# Mer information

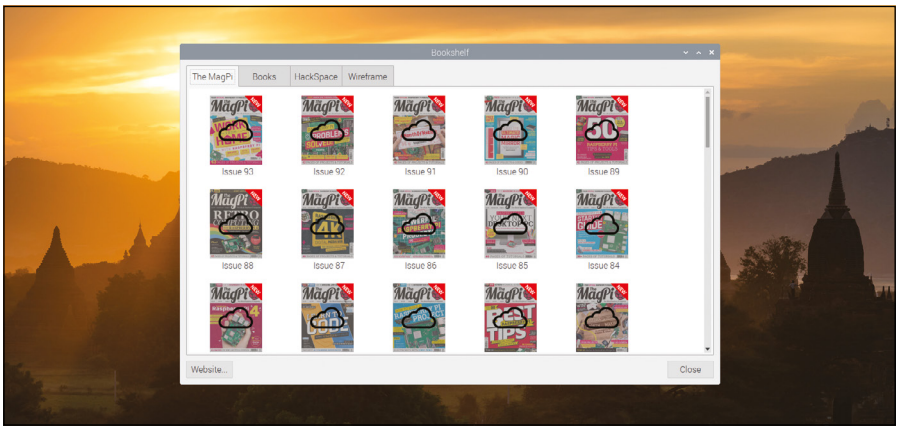

*Den officiella nybörjarguiden för Raspberry Pi* är utformad för att du ska komma igång med Raspberry Pi, men den visar långt ifrån allt du kan göra. Raspberry Pi-communityn finns över hela världen och människor använder dem för allt från spel och avkänningsapplikationer till robotteknik och artificiell intelligens. Det finns en massa inspiration där ute.

Den här bilagan belyser några källor till projektidéer, lektionsplaner och annat material som fungerar som ett bra nästa steg nu när du har arbetat dig igenom *Nybörjarguiden*.

## Bookshelf

**Raspberry Menu > Help > Bookshelf**

Bookshelf är en applikation som ingår i Raspberry Pi OS, vilken hjälper dig att bläddra, ladda ner och läsa digitala versioner av Raspberry Pi Press-publikationer – inklusive den här *Nybörjarhandboken för Raspberry Pi*. Läs in den genom att klicka på hallonmenyikonen, välja Help och klicka på Bookshelf. Sedan kan du bläddra bland en rad olika tidskrifter och böcker, alla gratis att ladda ner och läsa på din fritid.




## Raspberry Pi Blog

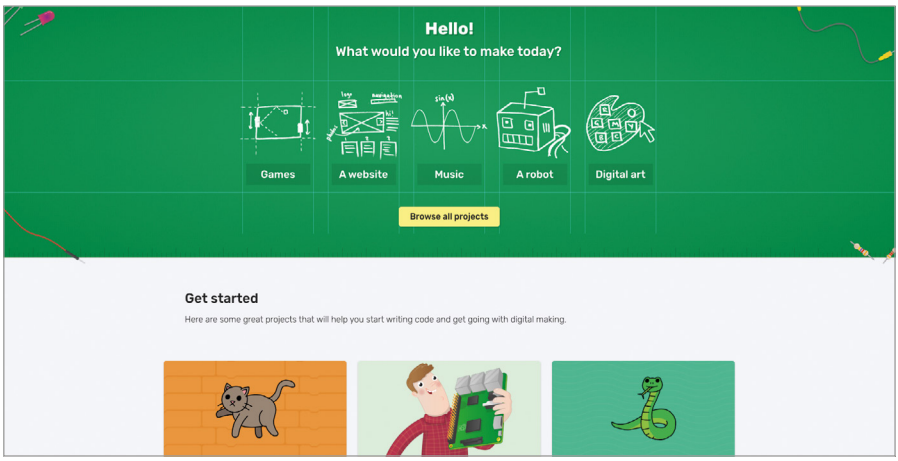
**rpf.io/blog**

Den officiella bloggen är ditt första stopp för de senaste nyheterna om allt som rör Raspberry Pi. Den täcker allt från nya maskinvarulanseringar och utbildningsmaterial till samlingar av de bästa communityprojekten, -kampanjerna och -initiativen. Om du vill hålla dig uppdaterad om allt som rör Raspberry Pi är det här rätta stället.

## Raspberry Pi Projects

**rpf.io/projects**

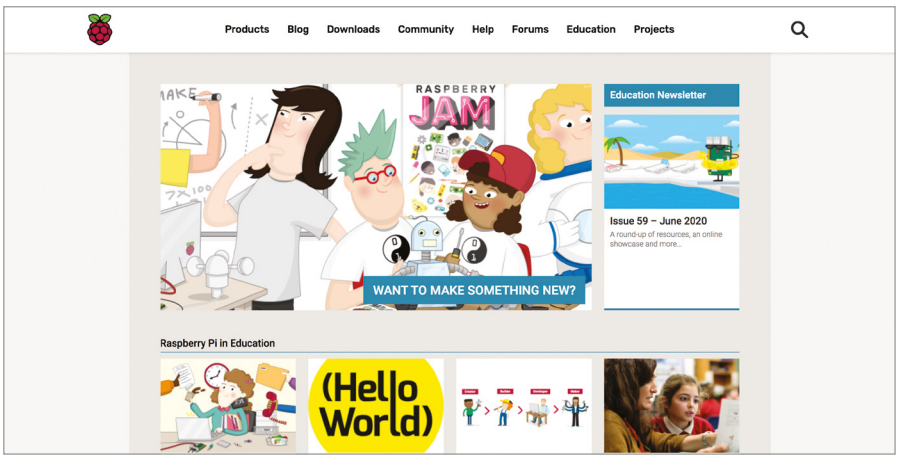
Den officiella Raspberry Pi Projects-webbplatsen erbjuder steg-för-steg-projekthandledning i en rad kategorier, från att skapa spel och musik till att bygga en egen webbplats eller en Raspberry Pi-driven robot. De flesta projekt finns även tillgängliga på olika språk och täcker en rad svårighetsgrader som passar alla från absoluta nybörjare till erfarna tillverkare.




## Raspberry Pi Education

**rpf.io/education**

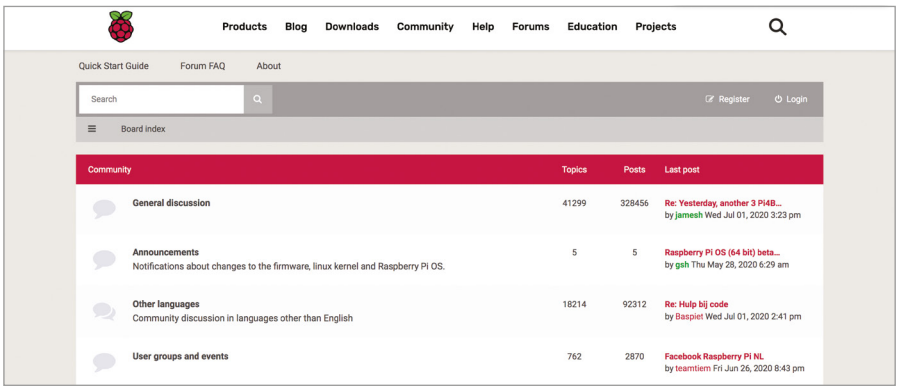
Den officiella Raspberry Pi Education-webbplatsen erbjuder nyhetsbrev, onlineutbildning och projekt med pedagoger i åtanke. Webbplatsen länkar även till ytterligare resurser, inklusive Picademy-utbildningsprogrammet, de volontärstyrda kodningsprogrammen Code Club och CoderDojo och globala Raspberry Jam-evenemang.



## Raspberry Pi Forums

**rpf.io/forums**

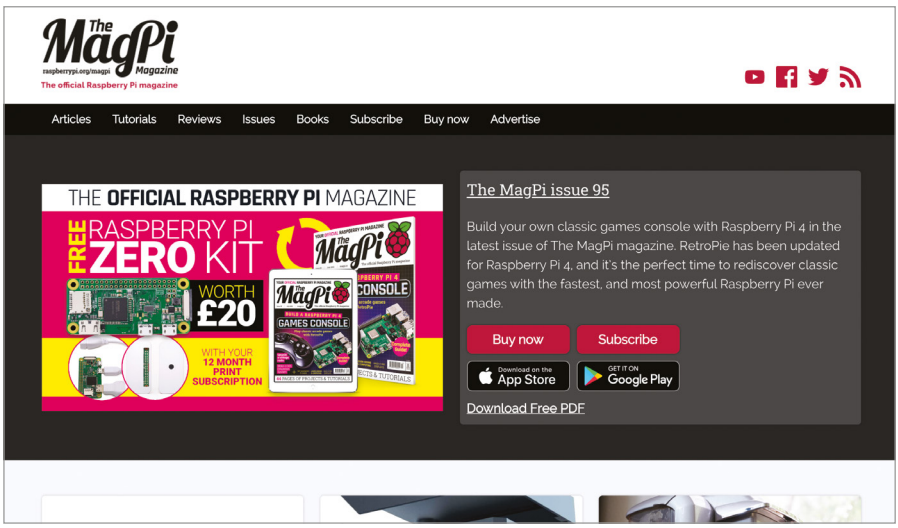
I Raspberry Pi Forums kan Raspberry Pi-fans träffas och chatta om allt från nybörjarfrågor till djupt tekniska ämnen – och det finns till och med ett ”off-topic”-område för allmän chatt!




## The MagPi Magazine

**magpi.cc**

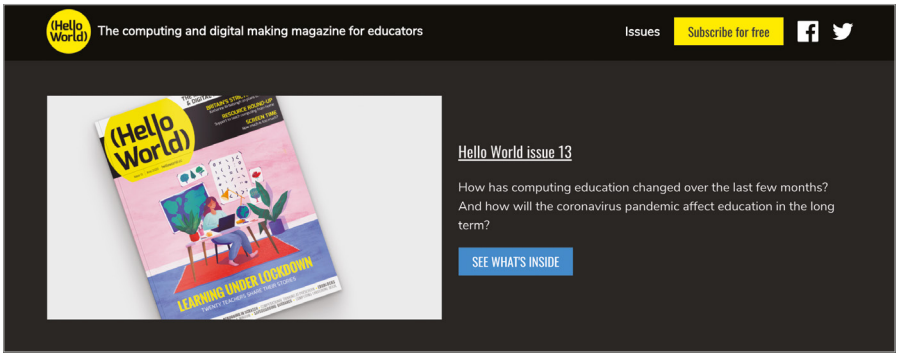
Den officiella Raspberry Pi-tidskriften The MagPi är en månadspublikation som täcker allt från handledning och guider till recensioner och nyheter och som till stor del kan ges ut tack vare stödet från den globala Raspberry Pi-communityn. Den finns i alla välsorterade tidningsbutiker och stormarknader och kan även laddas ner som en gratis digital version under Creative Commons-licensen. The MagPi publicerar även böcker och boksamlingar i en rad olika ämnen, som finns att köpa i tryckt format eller ladda ner gratis.



## Hello World Magazine

**helloworld.cc**

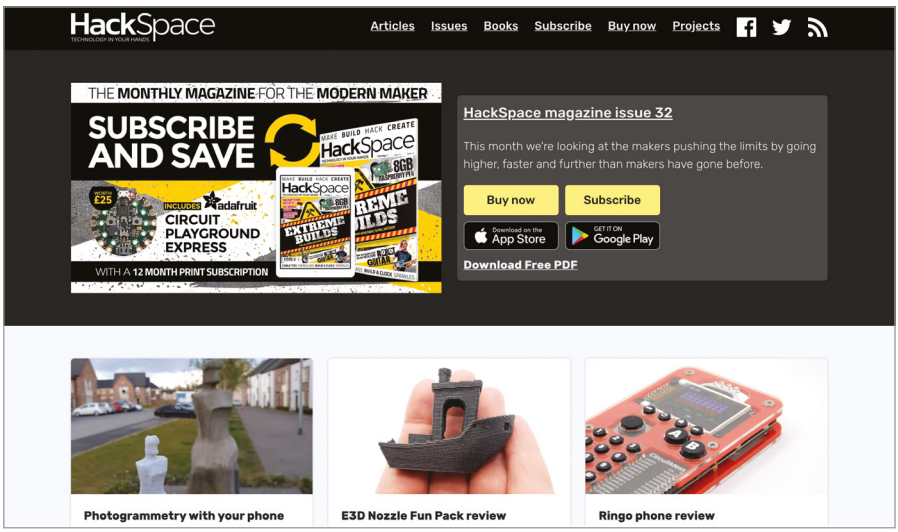
Hello World publiceras tre gånger om året och finns tillgänglig utan kostnad för lärare, volontärer och bibliotekarier i Storbritannien. Alla andra kan ladda ner gratis digitala exemplar under Creative Commons-licensen, och prenumerationer på den tryckta versionen finns kommersiellt tillgängliga.




## HackSpace Magazine

**hsmag.cc**

HackSpace Magazine riktar sig till en bredare publik än The MagPi genom att titta på tillverkarcommunityn med maskinvaru- och programvarurecensioner, handledningar och intervjuer. Om du är intresserad av att vidga vyerna bortom Raspberry Pi är HackSpace Magazine ett utmärkt ställe att börja på. Den finns i tryckt version på stormarknader och i tidningsbutiker och kan laddas ner gratis i digital form.



# Bilaga E

# Verktyget Raspberry Pi Configuration

Verktyget Raspberry Pi Configuration är ett kraftfullt paket som kan justera flertalet inställningar på Raspberry Pi, från de tillgängliga gränssnitten för program till att kontrollera den via ett nätverk. Det kan dock vara lite skrämmande för nykomlingar, så den här bilagan går igenom var och en av inställningarna i tur och ordning och förklarar deras syften.

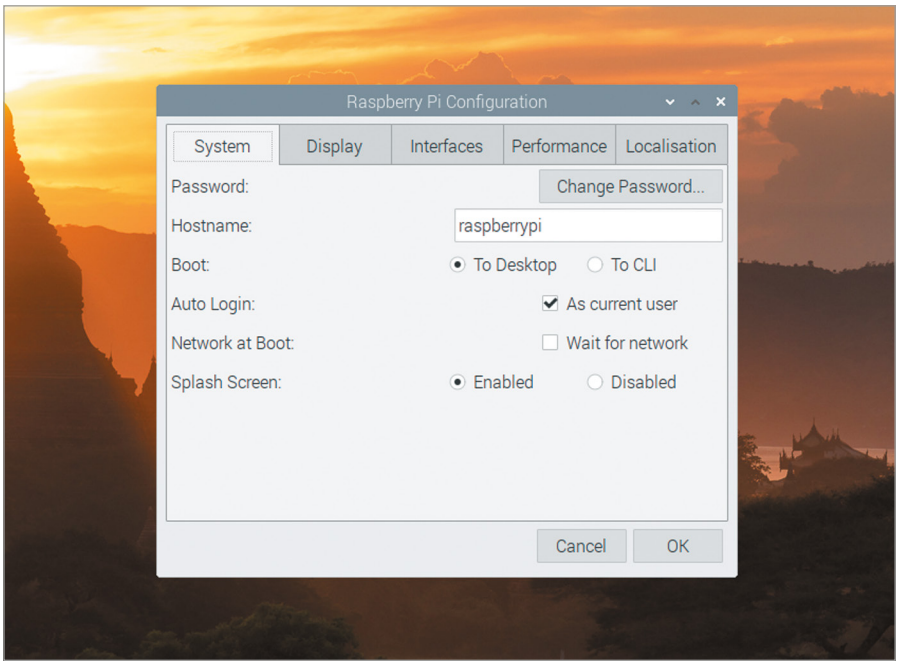
Du kan läsa in verktyget Raspberry Pi Configuration från hallonikonmenyn under kategorin Inställningar. Det kan också köras från kommandoradsgränssnittet eller LXTerminal med kommandot **`raspi-config`**. Layouterna för kommandoradsversionen och den grafiska versionen är olika, med alternativ som visas i olika kategorier, beroende på vilken version du använder. Denna bilaga är baserad på den grafiska versionen.

### VARNING!

Om du inte är säker på om du behöver ändra en viss inställning är det bäst att låta verktyget Raspberry Pi Configuration vara. Om du lägger till ny maskinvara i Raspberry Pi, till exempel en ljud-HAT eller en Camera Module, står det i instruktionerna vilken inställning du ska ändra. Annars bör standardinställningarna i allmänhet lämnas orörda.

## System-fliken

System-fliken innehåller alternativ som styr olika Raspberry Pi OS-systeminställningar.




- **Password:** Klicka på knappen ”Change Password…” för att ställa in ett nytt lösenord för ditt nuvarande användarkonto. Som standard är detta ”pi” -kontot.

- **Hostname:** Namnet som identifierar en Raspberry Pi på nätverk. Om du har mer än en Raspberry Pi på samma nätverk måste de alla ha ett unikt namn.

- **Boot:** Om du ställer in denna på ”To Desktop” (standard) laddas det välbekanta Raspberry Pi OS-skrivbordet. Om du ställer in den till ”To CLI” laddas kommandoradsgränssnittet som beskrivs i **Bilaga C, Kommandoradsgränssnittet**.

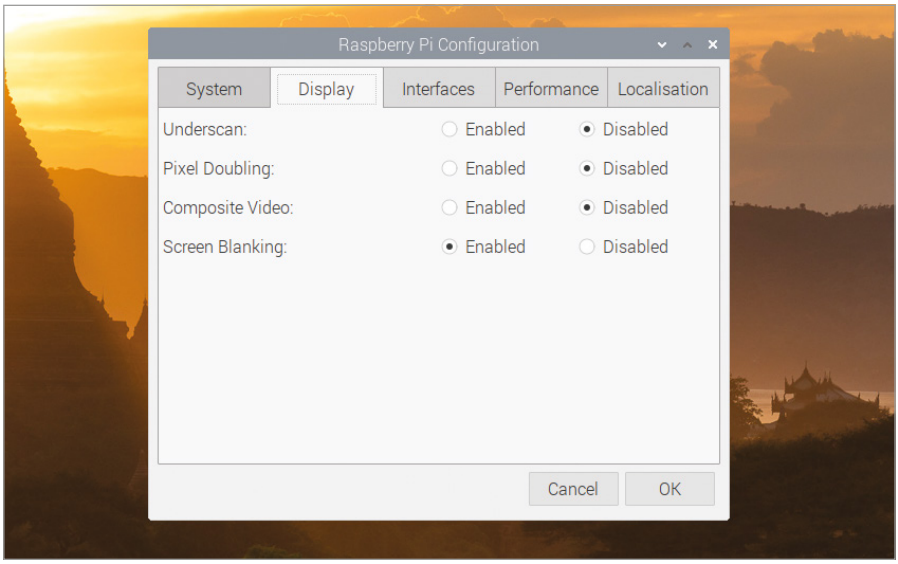
- **Auto Login:** När ”Log in as user ’pi’” är markerat (standard) laddar Raspberry Pi OS skrivbordet utan att du behöver skriva in ditt användarnamn och lösenord.

- **Network at Boot:** När "Wait for network” är markerat laddas inte Raspberry Pi OS förrän det har en fungerande nätverksanslutning.

- **Splash Screen:** När den är inställd på ”Enabled” (standard) döljs Raspberry Pi OS:s startmeddelanden bakom en grafisk välkomstskärm.

## Display-fliken

Display-fliken innehåller inställningar som styr hur skärmen visas.




- **Overscan:** Denna inställning styr huruvida videoutdatan på Raspberry Pi inkluderar svarta staplar runt kanterna, för att kompensera för ramen på många tv-apparater. Om du ser svarta staplar ställer du in denna till ”Disabled”; om inte, lämna den på ”Enabled”.

- **Pixel Doubling:** Om du använder en högupplöst men liten bildskärm kan du aktivera Pixel Doubling för att göra allt på skärmen större och lättare att se.

- **Composite Video:** Detta styr den komposita videoutgången som finns tillgänglig på det kombinerade ljud-video-uttaget (AV), när det används med en TRRS-adapter (tip-ring-ring-sleeve). Om du vill använda den komposita videoutgången istället för HDMI ställer du in den som ”Enabled”; annars låter du den vara inaktiverad.

- **Screen Blanking:** Med det här alternativet kan du slå på och stänga av skärmsläckning (tidsgräns som släcker ner skärmen efter några minuter).

## Interfaces-fliken

Interfaces-fliken innehåller inställningar som styr tillgängliga maskinvarugränssnitt på Raspberry Pi.

- **Camera:** Aktiverar eller inaktiverar Camera Serial Interface (CSI) för användning med en Raspberry Pi Camera Module.

- **SSH:** Aktiverar/inaktiverar Secure Shell-gränssnittet (SSH). Det låter dig öppna ett kommandoradsgränssnitt på Raspberry Pi från en annan dator i nätverket med en SSH-klient.

- **VNC:** Aktiverar/inaktiverar Virtual Network Computing-gränssnittet (VNC). Det låter dig visa skrivbordet på Raspberry Pi från en annan dator i nätverket med en VNC-klient.

- **SPI:** Aktiverar eller inaktiverar Serial Peripheral Interface (SPI), som används för att styra vissa maskinvarutillägg som ansluts till GPIO-stiften.

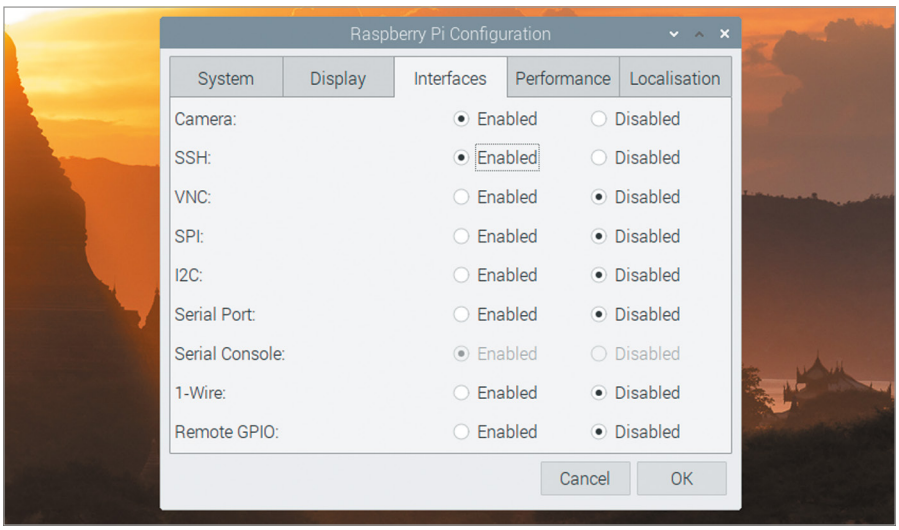
- **I2C:** Aktiverar eller inaktiverar Inter-Integrated Circuit-gränssnittet (I²C), som används för att styra vissa maskinvarutillägg som ansluts till GPIO-stiften.

- **Serial Port:** Aktiverar eller inaktiverar Raspberry Pi:s serieport, som finns tillgänglig på GPIO-stiften.

- **Serial Console:** Aktiverar eller inaktiverar seriekonsolen, ett kommandoradsgränssnitt som finns tillgängligt på serieporten. Det här alternativet är endast tillgängligt om inställningen Serial Port ovan är inställd på Enabled.

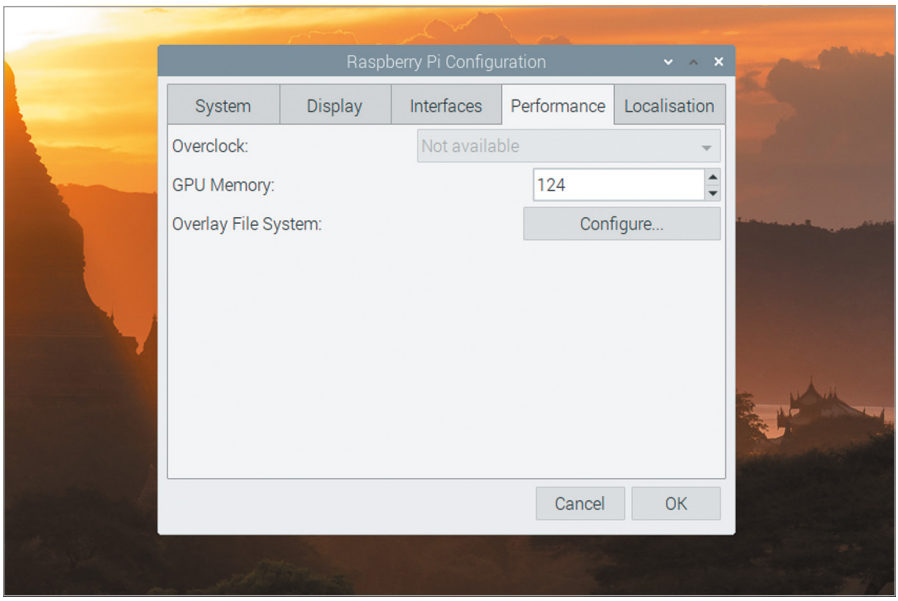
- **1-Wire:** Aktiverar eller inaktiverar 1-Wire-gränssnittet (I²C), som används för att styra vissa maskinvarutillägg som ansluts till GPIO-stiften.

- **Remote GPIO:** Aktiverar eller inaktiverar en nätverkstjänst som tillåter dig att styra Raspberry Pi:s stift från en annan dator på nätverket med hjälp av GPIO Zero-biblioteket. Mer information om fjärransluten GPIO finns på **gpiozero.readthedocs.io**.



## Performance-fliken

Performance-fliken innehåller inställningar som styr hur mycket minne som är tillgängligt och hur snabbt Raspberry Pi:s processor körs.



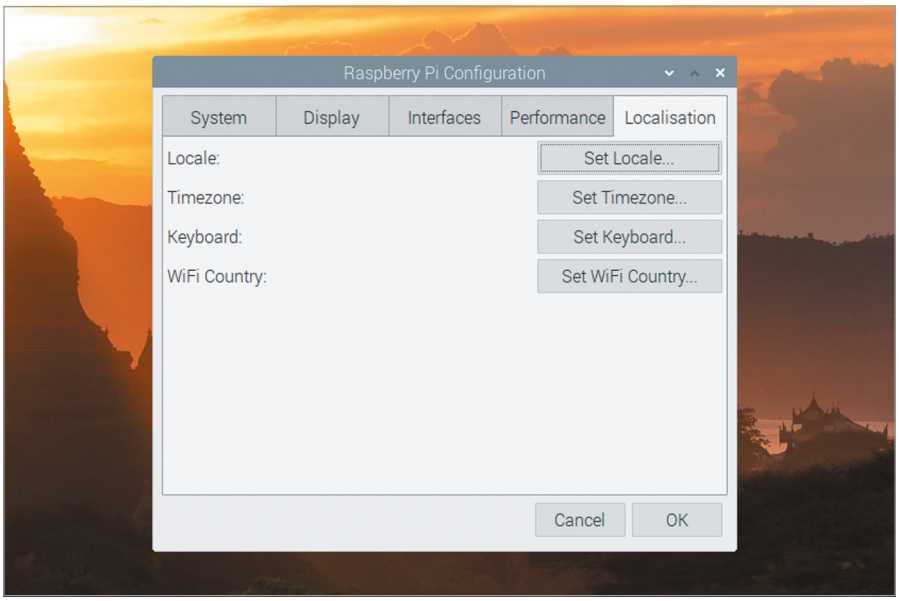

- **Overclock:** Låter dig välja mellan en rad inställningar som ökar prestandan hos Raspberry Pi på bekostnad av ökad strömförbrukning, värmeproduktion och möjlig minskad total livslängd. Finns ej tillgängligt på alla modeller av Raspberry Pi.

- **GPU Memory:** Låter dig ställa in den mängd minne som endast ska användas av Raspberry Pi:s grafikprocessor. Värden som är högre än standardvärdet kan förbättra prestandan för komplicerade 3D-renderings- och GPU-processer för allmänna syften (GPGPU) på bekostnad av att tillgängligt minne på Raspberry Pi OS minskar; lägre värden kan förbättra prestandan för minnesintensiva processer på bekostnad av att 3D-rendering, kamera och utvalda videouppspelningsfunktioner blir långsammare eller otillgängliga.

- **Overlay File System:** Låter dig stänga ner Raspberry Pi:s filsystem så att ändringar endast görs på en virtuell RAM-skiva istället för att skrivas till microSD-kortet, så att kortet är tomt när du startar om.

## Localisation-fliken

Localisation-fliken innehåller inställningar som styr vilken region Raspberry Pi är utformad för att fungera i, inklusive inställningar för tangentbordslayout.




- **Locale:** Låter dig välja din locale, en systeminställning som inkluderar språk, land och teckenuppsättning. Observera att om du byter språk här kommer det endast att ändra det visade språket i applikationer som det finns en översättning för.

- **Timezone:** Låter dig välja din regionala tidszon. Du väljer ett område i världen följt av närmaste stad. Om Raspberry Pi är ansluten till nätverket men klockan visar fel tid beror det vanligtvis på att fel tidszon har valts.

- **Keyboard:** Låter dig välja tangentbordstyp, språk och layout. Om du märker att tangentbordet skriver fel bokstäver eller symboler kan du korrigera det här.

- **WiFi Country:** Låter dig ställa in ditt land i radioreglementssyfte. Se till att välja det land där din Raspberry Pi används: att välja ett annat land kan göra det omöjligt att ansluta till närliggande Wi-Fi-åtkomstpunkter och kan vara ett brott mot radiolagen. Ett land måste ställas in innan Wi-Fi-frekvenser kan användas.

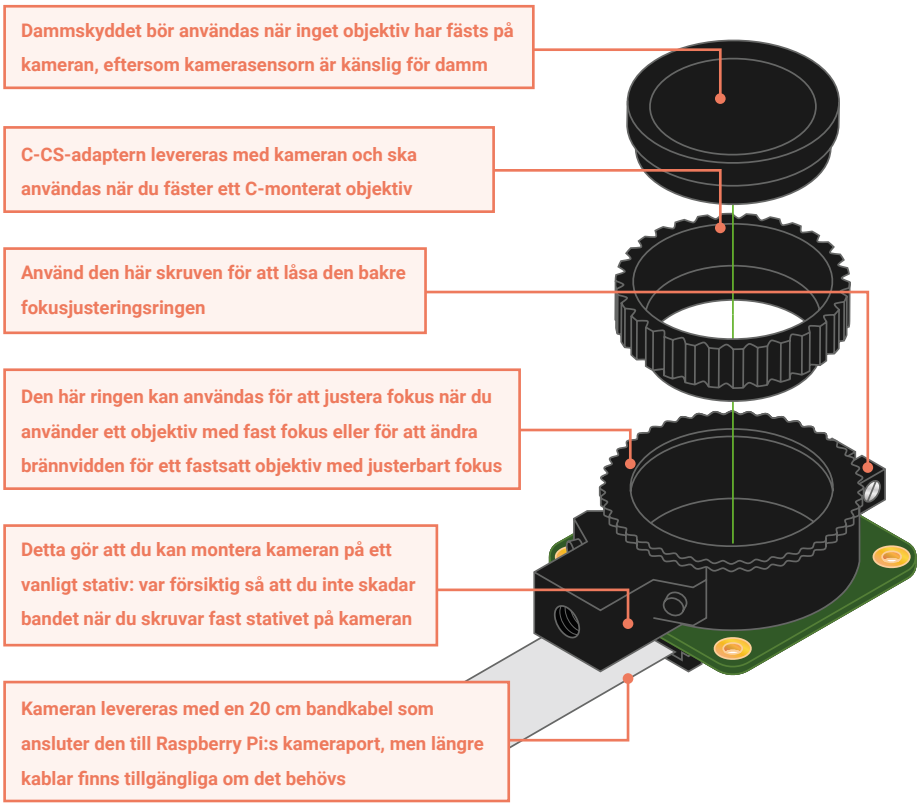
Bilaga F

# Installera High Quality Camera

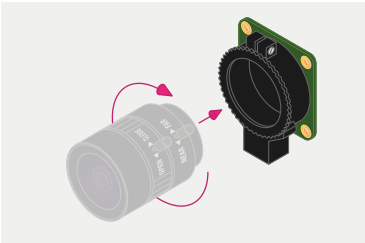
High Quality Camera (förkortat HQ Camera) kan ta bilder med högre upplösning än den standardmässiga Camera Module. Till skillnad från den senare har den inte ett redan anslutet objektiv. Den kan istället användas med alla standard C- eller CS-monterade objektiv; 6 mm- och 16 mm-objektiv finns att köpa tillsammans med kameran för att hjälpa dig att komma igång.

## 6 mm CS-monterat objektiv

Ett billigt 6 mm-objektiv finns tillgängligt för HQ Camera. Detta objektiv är lämpligt för grundläggande fotografering. Det kan också användas för makrofotografering eftersom det kan fokusera objekt på mycket korta avstånd.

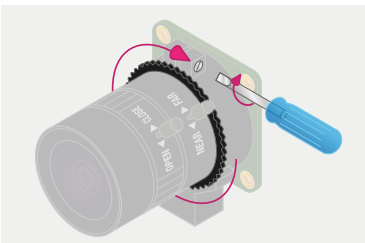



## 01 Fästa objektivet



6 mm-objektivet är en CS-monterad enhet, så den behöver inte C-CS-adapterringen (se diagrammet ovan). Det fokuserar inte ordentligt om adaptern sitter på, så ta bort den vid behov. Vrid sedan objektivet medurs hela vägen in i den bakre fokusjusteringsringen.

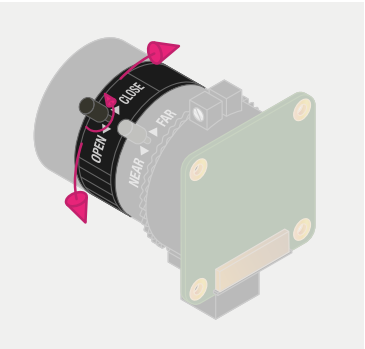
## 02 Bakre fokusjusteringsring och låsskruv



Den bakre fokusjusteringsringen ska skruvas in helt för kortast möjliga bakre brännvidd. Använd låsskruven för bakre fokus för att se till att den inte rör sig ur denna position när du justerar bländare eller fokus.

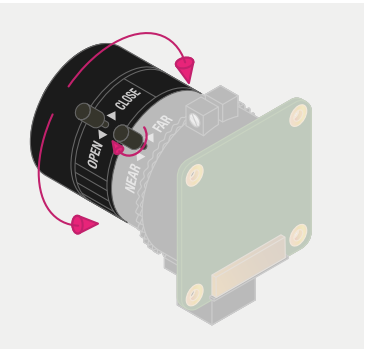
## 03 Bländare

För att justera bländaren håller du kameran med objektivet vänt från dig. Vrid mittringen medan du håller den yttre ringen, längst bort från kameran, stadig. Vrid medurs för att stänga bländaren och minska bildens ljusstyrka. Vrid moturs för att öppna bländaren. När du är nöjd med ljusnivån drar du åt skruven på sidan av objektivet för att låsa bländaren.




## 04 Fokus

Lås först den inre fokusringen, märkt ”NEAR ◀▶ FAR”, i position genom att dra åt skruven. Håll nu kameran med objektivet vänt från dig. Håll i de två yttre ringarna på objektivet och vrid dem båda medurs tills bilden kommer i fokus – det krävs fyra eller fem varv. För att justera fokus vrider du de två yttre ringarna medurs för att fokusera på ett närliggande objekt. Vrid dem moturs för att fokusera på ett avlägset objekt. Du kan behöva justera bländaren igen efter detta.

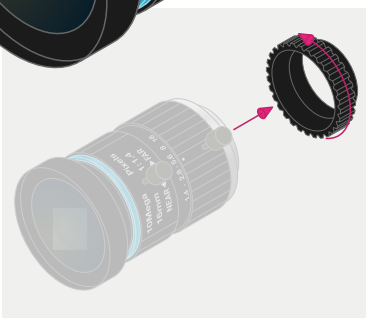



### 16 mm C-monterat objektiv

16 mm-objektivet ger en högre bildkvalitet än 6 mm-objektivet. Den har en snäv synvinkel som är mer lämpad för att se avlägsna objekt.

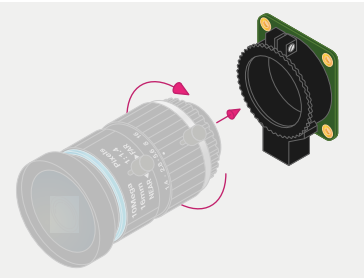

## 01 Fästa C-CS-adaptern

Se till att C-CS-adaptern som medföljer HQ Camera är monterad på 16 mm-objektivet. Objektivet är en C-monterad enhet, så det har längre bakre fokus än 6 mm-objektivet och behöver därför adaptern.



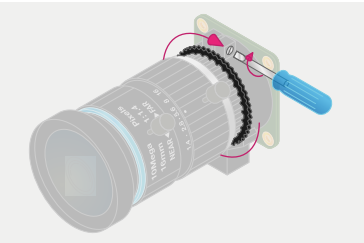
## 02 Fäst objektivet på kameran

Vrid 16 mm-objektivet och C-CS-adaptern medurs hela vägen in i den bakre fokusjusteringsringen.



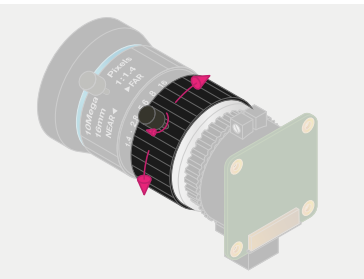
## 03 Bakre fokusjusteringsring och låsskruv

Den bakre fokusjusteringsringen ska skruvas in helt. Använd låsskruven för bakre fokus för att se till att den inte rör sig ur denna position när du justerar bländare eller fokus.



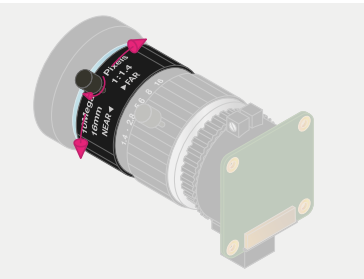
## 04 Bländare

För att justera bländaren håller du kameran med objektivet vänt från dig. Vrid den inre ringen, närmast kameran, medan du håller kameran stadig. Vrid medurs för att stänga bländaren och minska bildens ljusstyrka. Vrid moturs för att öppna bländaren. När du är nöjd med ljusnivån drar du åt skruven på sidan av objektivet för att låsa bländaren i position.



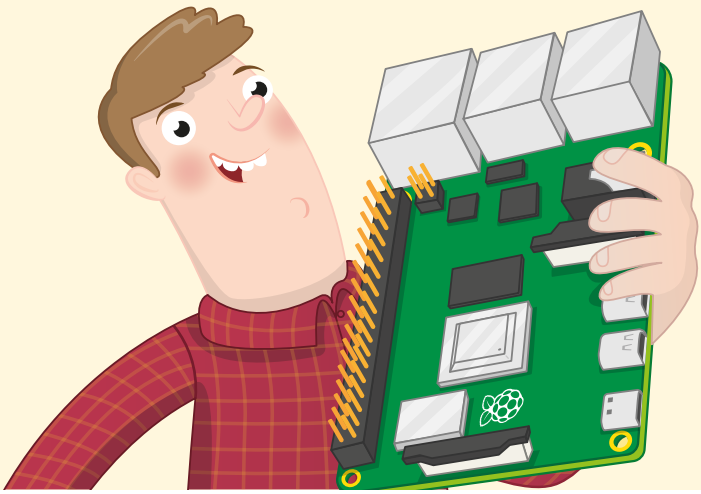
## 05 Fokus

För att justera fokus håller du kameran med objektivet vänt från dig. Vrid fokusringen, märkt ”NEAR ◀▶ FAR”, moturs för att fokusera på ett närliggande objekt. Vrid den medurs för att fokusera på ett avlägset objekt. Du kan behöva justera bländaren igen efter detta.



## Bilaga G

# Raspberry Pi-specifikationer

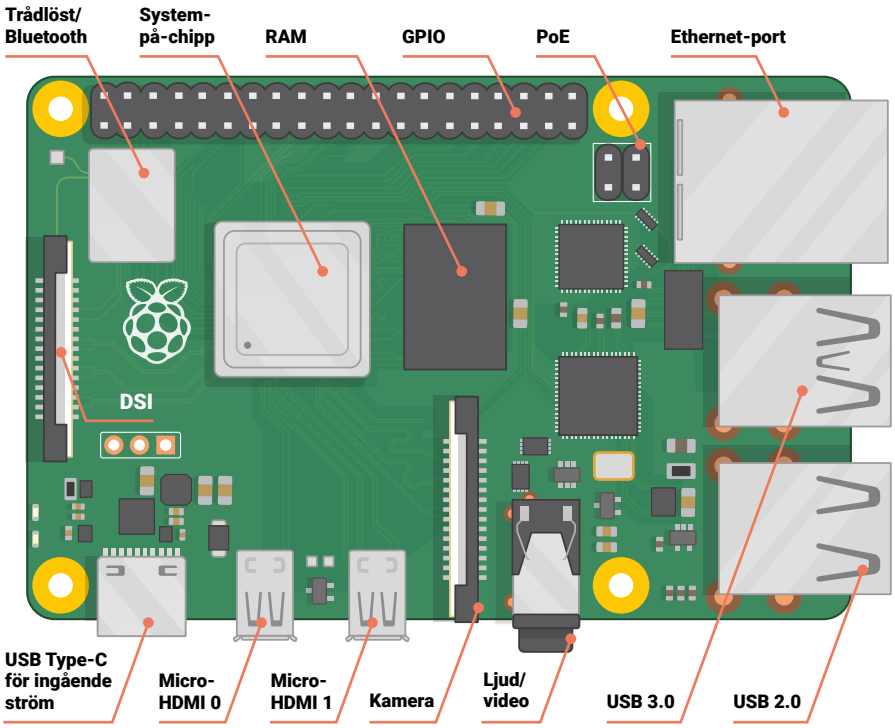


De olika komponenterna och funktionerna i en dator kallas specifikationer, och om du kollar på specifikationerna får du den information du behöver för att jämföra två datorer. Dessa specifikationer kan verka förvirrande till en början. De är mycket tekniska och du behöver inte känna till dem för att använda en Raspberry Pi, men de ingår här för den nyfikne läsaren.

System-på-chipp för Raspberry Pi 4 Model B och Raspberry Pi 400 är en Broadcom BCM2711B0, som du kan läsa på metallocket om du tittar noggrant (på Raspberry Pi 4). Detta har fyra 64-bit ARM Cortex-A72-centralprocessorer (CPU), som alla körs vid 1,5 GHz eller 1,8 GHz (1,5 eller 1,8 tusen miljoner cykler per sekund) och en Broadcom VideoCore VI (Six) grafikprocessor (GPU) som körs vid 500 MHz (500 miljoner cykler per sekund) för videoprocesser och för 3D-renderingsprocesser såsom spel.

System-på-chipp är anslutet till 2 GB, 4 GB eller 8 GB (två, fyra eller åtta tusen miljoner byte) – 4 GB på Raspberry Pi 400 – av LPDDR4 (Low-Power Double-Data-Rate 4) RAM (random-access memory) som körs vid 3 200 MHz (tretusen tvåhundra miljoner cykler per sekund). Detta minne delas mellan centralprocessorn och grafikprocessorn. MicroSD-kortplatsen stöder upp till 512 GB (512 tusen miljoner byte) lagring.

Ethernet-porten stöder upp till gigabitanslutningar (1 000 Mbit/s, 1000-Base-T), medan frekvensen stöder 802.11ac Wi-Fi-nätverk som körs på frekvensbanden 2,4 GHz och 5 GHz, Bluetooth 5.0 och Bluetooth Low Energy-anslutningar (BLE).




I en punktlista ser Raspberry Pi 4:s specifikationer ut så här:

- **CPU:** 1,5 GHz 64-bit quad-core ARM Cortex-A72
- **GPU:** 500 MHz VideoCore VI
- **RAM:** 1 GB, 2 GB eller 4 GB av LPDDR4
- **Nätverksaktivitet:** Gigabit Ethernet, dual-band 802.11ac, Bluetooth 5.0, Bluetooth Low Energy
- **Ljud-/videoutgångar:** 3,5 mm analogt AV-uttag, 2 × mikro-HDMI 2.0
- **Perifer anslutning:** 2 × USB 2.0-portar, 2 × USB 3.0-portar, Camera Serial Interface, Display Serial Interface (DSI)
- **Lagringsenhet:** microSD, upp till 512 GB
- **Strömförsörjning:** 5 volt vid 3 ampere via USB Type-C
- **Tillägg:** GPIO med 40 stift, Power over Ethernet-kompatibilitet (med ytterligare maskinvara)

Raspberry Pi 400:s specifikationer är:

- **CPU:** 1,8 GHz 64-bit quad-core ARM Cortex-A72
- **GPU:** 500 MHz VideoCore VI
- **RAM:** 4 GB av LPDDR4
- **Nätverksaktivitet:** Gigabit Ethernet, dual-band 802.11ac, Bluetooth 5.0, Bluetooth Low Energy
- **Ljud-/videoutgångar:** 2 × micro-HDMI 2.0
- **Perifer anslutning:** 1 × USB 2.0-port, 2 × USB 3.0-portar
- **Lagringsenhet:** microSD, upp till 512 GB (16 GB medföljer)
- **Strömförsörjning:** 5 volt vid 3 ampere via USB Type-C
- **Tillägg:** GPIO med 40 stift

# Bilaga H

# Säkerhets- och användarhandbok för Raspberry Pi

**Raspberry Pi**

Designad och distribuerad av
Raspberry Pi-Trading Ltd
Maurice Wilkes Building
Cowley Road
Cambridge
CB4 0DS
Storbritannien
www.raspberrypi.org

**Raspberry Pi** Regleringsefterlevnad och säkerhetsinformation

Raspberry Pi 4 Model B
FCC-ID: 2ABCB-RPI4B
IC-ID: 20953-RPI4B

Raspberry Pi 400
FCC-ID: 2ABCB-RPI400
IC-ID: 20953-RPI400

VIKTIGT: Innan du ansluter till strömförsörjningen, se installationsanvisningarna på **www.raspberrypi.org/safety**

**VARNING: Cancer och reproduktiv skada – www.P65Warnings.ca.gov.**

All regleringsinformation och alla certifikat finns på **www.raspberrypi.org/compliance**

IFETEL: 2019LAB-ANCE4957
Certifieringsnummer för Raspberry Pi 4 Model B.

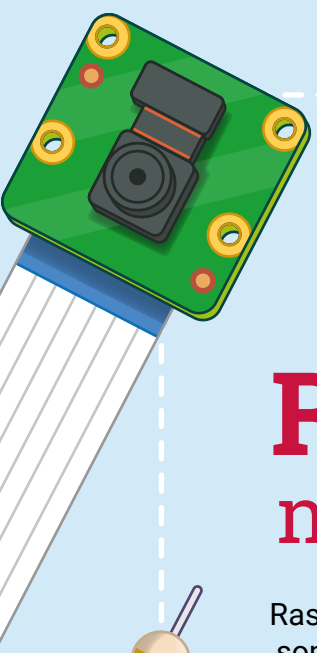
TRA-registreringsnummer
ER73381/19
Certifieringsnummer för Raspberry Pi 4 Model B.

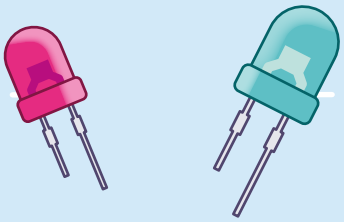
CCAO19LP1120T2
Certifieringsnummer för Raspberry Pi 4 Model B.

NTC
Typ godkänd:
NR.: ESD-GEC-1920098C
Certifieringsnummer för Raspberry Pi 4 Model B.

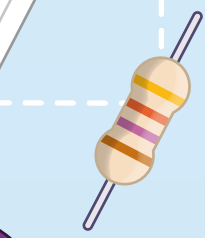
TA-2019/750 GODKÄND
Certifieringsnummer för Raspberry Pi 4 Model B.

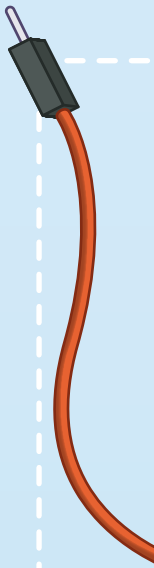
De använda varumärkena HDMI, HDMI High-Definition Multimedia Interface och HDMI-logotypen är varumärken eller registrerade varumärken som tillhör HDMI Licensing Administrator, Inc. i USA och andra länder.





# DEN OFFICIELLA
# Raspberry Pi
# nybörjarguiden för

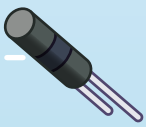




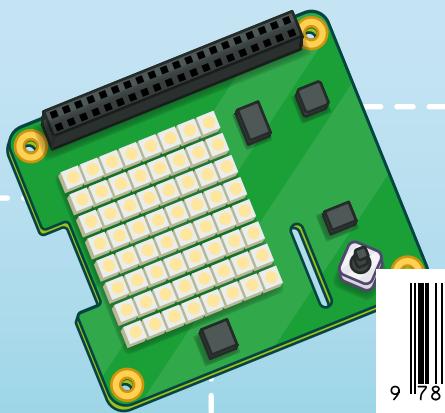
Raspberry Pi är en liten, smart, brittiskbyggd dator som är full av potential. Raspberry Pi är tillverkad med samma teknik som du hittar i en smarttelefon och är utformad för att hjälpa dig att lära dig att programmera, upptäcka hur datorer fungerar och skapa egna fantastiska saker. Den här boken skrevs för att visa hur lätt det är att komma igång.

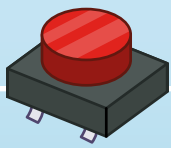
## Du får lära dig:

- att konfigurera Raspberry Pi, installera operativsystemet och börja använda den helt funktionella datorn.
- att börja programmera projekt med steg-för-steg-anvisningar med programspråken Scratch 3 och Python.
- att experimentera med att ansluta elektroniska komponenter och prova att skapa fantastiska projekt.









raspberrypi.org

9 781912 047840

Raspberry Pi PRESS