projekt

Astro Pi: Mission Zero

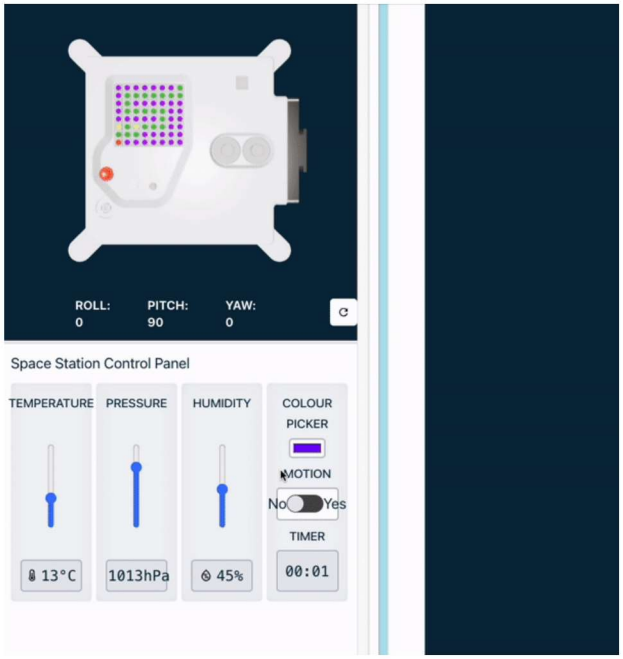
Förbered dig för Mission Zero

Step 1 Du kommer göra

Slutför det här projektet för att delta i Astro Pi Mission Zero-utmaningen och få din kod att köra i rymden på en Astro Pi-dator.

Ditt projekt kommer att ställa in bakgrundsfärgen för en bild till den färg som Astro Pi upptäcker. Detta kommer att göra den internationella rymdstationen (ISS) mer färgstark för astronauterna ombord. Din kod kommer att använda färgljussensorn på den nya Mark II Astro Pi-datorns Sense HAT för att få detta att hända.

Här är ett exempel på den typ av program du kan göra för att köra på en Astro Pi i rymden.



Du kommer behöva

Du kommer att använda Astro Pi-emulatorn i en webbläsare för att skapa ditt program. Du behöver ingen Astro Pi-dator.

Kriterier för Astro Pi Mission Zero

Om ditt projekt uppfyller kvalificeringskriterierna (https://astro-pi.org/sv/mission-zero/eligibility)kommer ditt avslutade program att köras på den internationella rymdstationen! Du kommer också att få ett särskilt certifikat som visar exakt var ISS var när ditt program körde.

Du kommer att lära dig om Astro Pi enheten och hur man styr den, inklusive hur man:

- Skapa färg **-variabler** att använda i din bild
- Designa och visa en bild på Sense HAT
- Känn ljusets färg ombord på ISS

Anteckningar för mentorer

Mission Zero är lämplig för nybörjare till programmering och/eller barn i grundskoleåldern och kan genomföras på en enda 60-minuters session på vilken dator som helst med internetuppkoppling. Ingen speciell hårdvara eller tidigare kodningskunskaper behövs. Allt kan göras i en webbläsare.

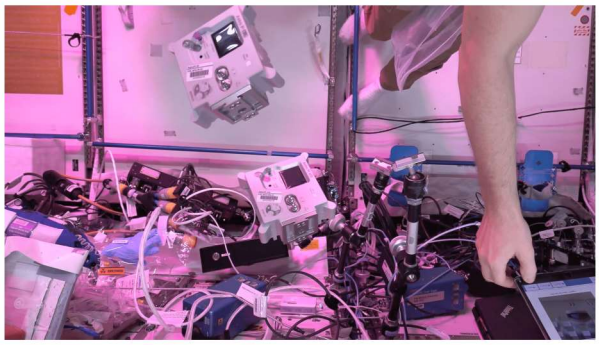
Organisera dina ungdomar i lag om en till fyra, och låt oss guida dem genom att skriva ett kort Python-program för att känna av färgen ombord på ISS och skapa en bild som använder den färgen.

Läs de **officiella riktlinjerna** (https://astro-pi.org/sv/mission-zero/guidelines) för Mission Zero.

Step 2 Vad är en Astro Pi?

En Astro Pi är en Raspberry Pi-dator monterad i en särskilt utformad låda för förutsättningarna i rymden.

Astro Pi-datorer kommer med en uppsättning sensorer och prylar som kan användas för att köra fantastiska vetenskapliga experiment. Denna uppsättning sensorer kallas en “Sense HAT” (som står för “Hardware Attached on Top”). Sense HAT ger Astro Pi möjligheten att “känna av” och göra många typer av mätningar, från temperatur till rörelse, och att mata ut information med hjälp av en 8 x 8 LED-matrisdisplay. Astro Pis har också en joystick och knappar, precis som en spelkonsol!



För det här uppdraget kommer du att använda Sense HAT-emulatorn som simulerar huvudfunktionerna hos Astro Pi i din webbläsare.

Step 3 Visa en bild

Astro Pis LED-matris kan visa färger. I det här steget kommer du att visa bilder från naturen på Astro Pis LED-matris.

> En **LED-matris** är ett rutnät av lysdioder som kan styras individuellt eller som en grupp för att skapa olika ljuseffekter. LED-matrisen på Sense HAT har 64 lysdioder som visas i ett 8 x 8 rutnät. Lysdioderna kan programmeras för att producera ett brett spektrum av färger.

Öppna **startprojektet Mission Zero** (https://missions.astro-pi.org/sv/mz/code_submissions/new).

Du kommer att se att några rader kod har lagts till för dig automatiskt.

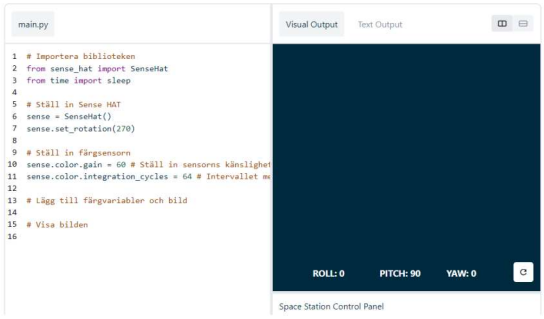
Den här koden ansluter till Astro Pi, ser till att Astro Pis LED-display visas på rätt sätt och ställer in färgsensorn. Lämna kvar koden där, för du kommer att behöva den.

main.py

```
# Importera biblioteken
from sense_hat import SenseHat
from time import sleep

# Ställ in Sense HAT
sense = SenseHat()
sense.set_rotation(270)

# Ställ in färgsensorn
sense.color.gain = 60 # Ställ in sensorns känslighet
sense.color.integration_cycles = 64 # Intervallet med vilket avläsningen kommer att ske
```

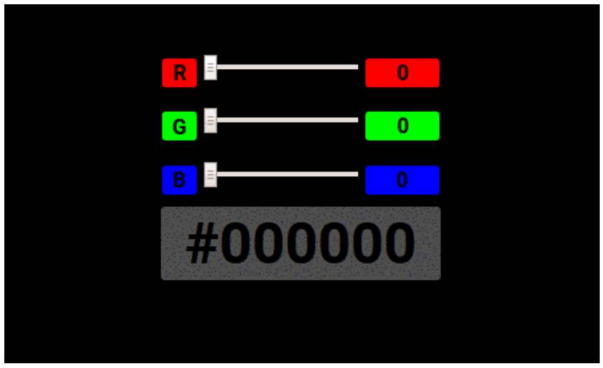


RGB-färger

Färger kan skapas med olika proportioner av rött, grönt och blått. Du kan läsa mer om RGB färger här:

RGB-färger

När vi vill representera en färg i ett datorprogram kan vi göra detta genom att definiera mängden rött, blått och grönt som kombineras för att utgöra den färgen. Dessa belopp lagras som ett tal mellan 0 och 255.



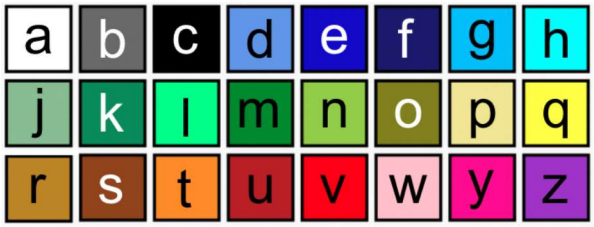
Här är ett tabell som visar några värden för olika färger:

Röd	Grön	Blå	Färg
255	0	0	Röd
0	255	0	Grön
0	0	255	Blå
255	255	0	Gul
255	0	255	Magenta
0	255	255	Cyan

Du kan hitta en fin **färgplockare att leka med på w3schools** (https://www.w3schools.com/colors/colors_rgb.asp).

LED-matrisen är ett 8 x 8 rutnät. Varje lysdiod på nätet kan ställas in på olika färger. Här är en lista med variabler för 24 olika färger. Varje färg har ett värde för rött, grönt och blått:

Lista över färgvariabler



main.py

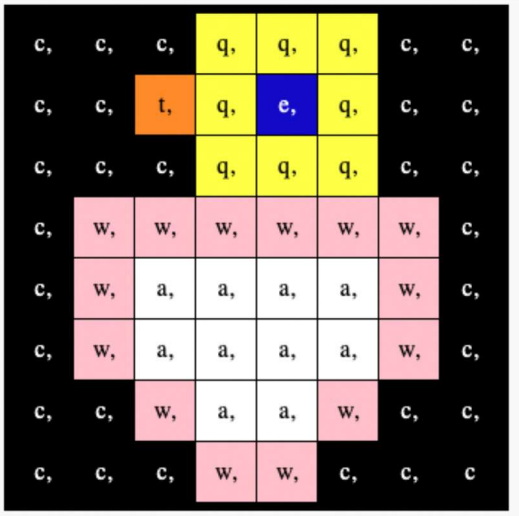
```
# Färgpalett
a = (255, 255, 255) # Vit
b = (105, 105, 105) # DimGrå
c = (0, 0, 0) # Svart
d = (100, 149, 237) # Blåklint
e = (0, 0, 205) # MediumBlå
f = (25, 25, 112) # Midnattsblå
g = (0, 191, 255) # MörkHimmelsBlå
h = (0, 255, 255) # Cyan
j = (143, 188, 143) # MörkHavsGrön
k = (46, 139, 87) # HavsGrön
l = (0, 255, 127) # VårGrön
m = (34, 139, 34) # SkogsGrön
n = (154, 205, 50) # Gulgrön
o = (128, 128, 0) # Oliv
p = (240, 230, 140) # Khaki
q = (255, 255, 0) # Gul
r = ( 184, 134, 11) # MörkGylleneRöd
s = (139, 69, 19) # SadelBrun
t = (255, 140, 0) # MörkOrange
u = (178, 34, 34) # Tegel
v = (255, 0, 0) # Röd
w = (255, 192, 203) # Rosa
y = (255, 20, 147) # MörkRosa
z = (153, 50, 204) # MörkOrkidèe
```

Välj en bild

Välj: Välj en bild att visa från alternativen nedan. Python lagrar informationen för en bild i en lista. Koden för varje bild inkluderar de färgvariabler som används och listan.

Du måste **kopiera** hela koden för din valda bild och sedan **klistra in** den i ditt projekt under raden som säger `# Lägg till färgvariabler och bild`.

Kyckling

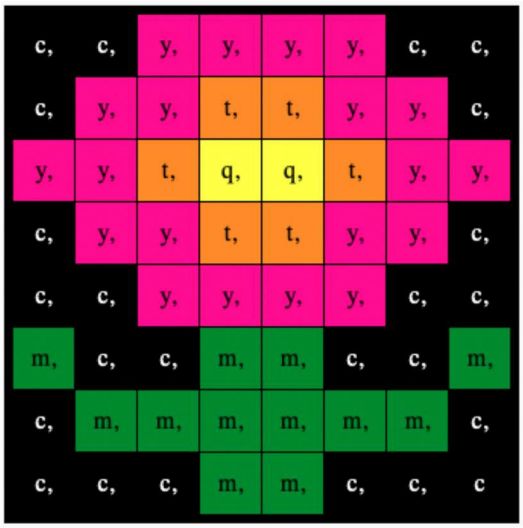


main.py

```
a = (255, 255, 255) # Vit
c = (0, 0, 0) # Svart
e = (0, 0, 205) # MediumBlå
q = (255, 255, 0) # Gul
t = (255, 140, 0) # MörkOrange
w = (255, 192, 203) # Rosa

bild = [
  c, c, c, q, q, q, c, c,
  c, c, t, q, e, q, c, c,
  c, c, c, q, q, q, c, c,
  c, w, w, w, w, w, w, c,
  c, w, a, a, a, a, w, c,
  c, w, a, a, a, a, w, c,
  c, c, w, a, a, w, c, c,
  c, c, c, w, w, c, c, c]
```

Blomma

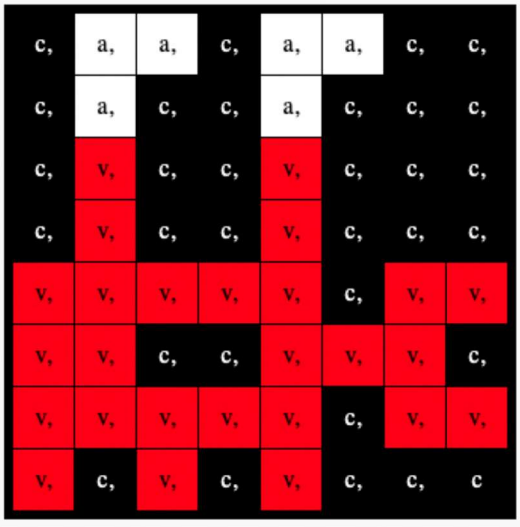


main.py

```
c = (0, 0, 0) # Svart
m = (34, 139, 34) # Skogsgrön
q = (255, 255, 0) # Gul
t = (255, 140, 0) # MörkOrange
y = (255, 20, 147) # Djuprosa

bild = [
  c, c, y, y, y, y, c, c,
  c, y, y, t, t, y, y, c,
  y, y, t, q, q, t, y, y,
  c, y, y, t, t, y, y, c,
  c, c, y, y, y, y, c, c,
  m, c, c, m, m, c, c, m,
  c, m, m, m, m, m, m, c,
  c, c, c, m, m, c, c, c]
```

Krabba

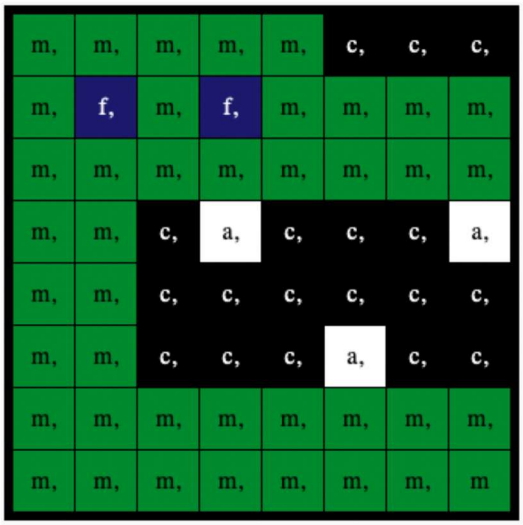


main.py

```
a = (255, 255, 255) # Vit
c = (0, 0, 0) # Svart
v = (255, 0, 0) # Röd

bild = [
  c, a, a, c, a, a, c, c,
  c, a, c, c, a, c, c, c,
  c, v, c, c, v, c, c, c,
  c, v, c, c, v, c, c, c,
  v, v, v, v, v, c, v, v,
  v, v, c, c, v, v, v, c,
  v, v, v, v, v, c, v, v,
  v, c, v, c, v, c, c, c]
```

Krokodil

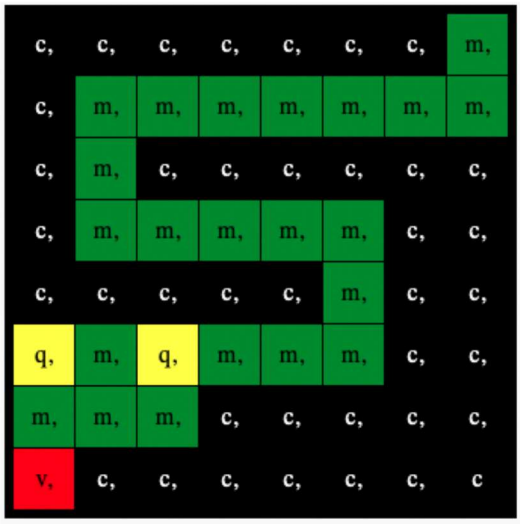


main.py

```
a = (255, 255, 255) # Vit
c = (0, 0, 0) # Svart
f = (25, 25, 112) # Midnattsblå
m = (34, 139, 34) # Skogsgrön

bild = [
  m, m, m, m, m, c, c, c,
  m, f, m, f, m, m, m, m,
  m, m, m, m, m, m, m, m,
  m, m, c, a, c, c, c, a,
  m, m, c, c, c ,c ,c ,c,
  m, m, c, c, c, a, c, c,
  m, m, m, m, m, m, m, m,
  m, m, m, m, m, m, m, m]
```

Orm

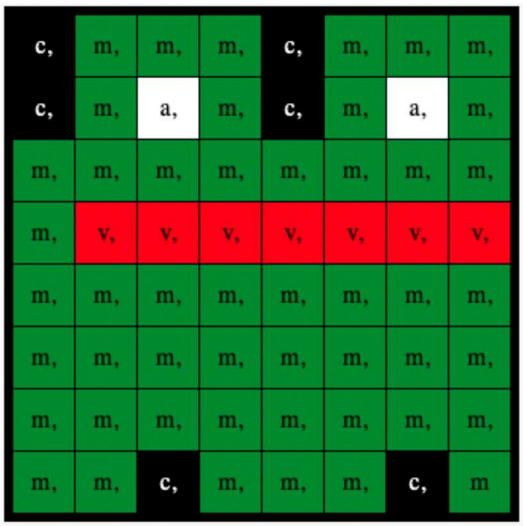


main.py

```
c = (0, 0, 0) # Svart
 m = (34, 139, 34) # Skogsgrön
 q = (255, 255, 0) # Gul
 v = (255, 0, 0) # Röd

bild = [
  c, c, c, c, c, c, c, m,
  c, m, m, m, m, m, m, m,
  c, m, c, c, c, c, c, c,
  c, m, m, m, m, m, c, c,
  c, c, c, c, c, m, c, c,
  q, m, q, m, m, m, c, c,
  m, m, m, c, c, c, c, c,
  v, c, c, c, c, c, c, c]
```

Groda



main.py

```
c = (0, 0, 0) # Svart
m = (34, 139, 34) # Skogsgrön
q = (255, 255, 0) # Gul
v = (255, 0, 0) # Röd

bild = [
  c, m, m, m, c, m, m, m,
  c, m, q, m, c, m, q, m,
  m, m, m, m, m, m, m, m,
  m, v, v, v, v, v, v, v,
  m, m, m, m, m, m, m, m,
  m, m, m, m, m, m, m, m,
  m, m, m, m, m, m, m, m,
  m, m, c, m, m, m, c, m]
```

Hitta: raden som säger `# Visa bilden` och lägg till en kodrad för att visa din bild på LED-matrisen:

main.py

```
bild = [
  c, c, c, q, q, q, c, c,
  c, c, t, q, e, q, c, c,
  c, c, c, q, q, q, c, c,
  c, w, w, w, w, w, w, c,
  c, w, a, a, a, a, w, c,
  c, w, a, a, a, a, w, c,
  c, c, w, a, a, w, c, c,
  c, c, c, w, w, c, c, c]

# Visa bilden
sense.set_pixels(bild)
```

Tryck på **Kör** längst ner i editorn för att se din bild visas på LED-matrisen.

Felsökning

Min kod har ett syntaxfel:

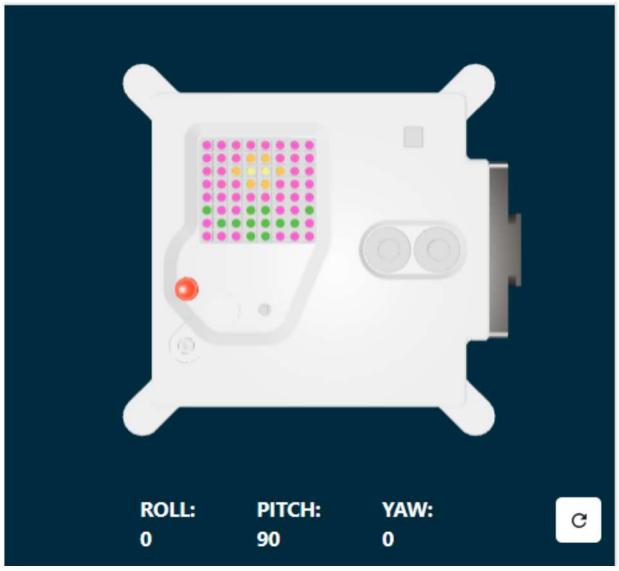
- Kontrollera att din kod matchar koden i exemplen ovan
- Kontrollera att du har dragit in koden i din lista
- Kontrollera att din lista är omgiven av `[` och `]`
- Kontrollera att varje färgvariabel i listan är avgränsad med ett kommatecken

Min bild visas inte:

- Kontrollera att din `sense.set_pixels(bild)` inte är indragen

Step 4 Känn en färg

I det här steget kommer du att ställa in färgljussensorn och använda den för att känna av mängden rött, grönt och blått som når sensorn. Denna färg kommer sedan att användas för att färglägga din valda bild. En astronaut som går upp till sensorn i en blå skjorta skulle se en annan bild än en astronaut i en röd skjorta.



Vilken bild du än väljer använder bakgrunden variabeln `c` som är inställd på svart.

Använd färgsensorn för att färga din bakgrund.

Lägg till kod före din bildlista för att få färgen från sensorn och ändra din `c` bakgrundsfärgvariabel för att använda färgen som avkänns av Sense HAT-färgsensorn istället för svart.

Tips: Du behöver inte skriva kommentarerna som börjar med '#' (de är till för att förklara koden).

main.py

```
# Lägg till färgvariabler och bild

c = (0, 0, 0) # Svart
m = (34, 139, 34) # Skogsgrön
q = (255, 255, 0) # Gul
t = (255, 140, 0) # MörkOrange
y = (255, 20, 147) # Djuprosa

rgb = sense.color # hämta färgen från sensorn
c = (rgb.red, rgb.green, rgb.blue) # använd den avkända färgen

bild = [
  c, c, y, y, y, y, c, c,
  c, y, y, t, t, y, y, c,
  y, y, t, q, q, t, y, y,
  c, y, y, t, t, y, y, c,
  c, c, y, y, y, y, c, c,
  m, c, c, m, m, c, c, m,
  c, m, m, m, m, m, m, c,
  c, c, c, m, m, c, c, c]
```

Test: Flytta färgreglaget till en färg som du väljer och sedan **kör** din kod. Din bakgrundsfärg kommer att ändras. Upprepa detta test igen med en ny färg.

Tips: Du måste klicka på "Kör" varje gång du ändrar färg.

Loopa ditt program

Astro Pi Mission Zero-programmet får köras i upp till 30 sekunder. Du kommer att använda denna tid för att upprepade gånger kontrollera färgsensorn och uppdatera bilden.

Din kod kommer att använda en `for` loop för att köra 28 gånger. **Varje** gång kommer den att:

- känna den senaste färgen
- uppdatera bildens bakgrundsfärg
- pausa i en sekund

Hitta din `rgb = sense.color` kodrad.

Lägg till kod ovanför den för att ställa in din `for` -loop för `28` repetitioner.

main.py

```
for i in range(28):
rgb = sense.color # hämta färgen från sensorn
c = (rgb.red, rgb.green, rgb.blue)

bild = [
  c, c, y, y, y, y, c, c,
  c, y, y, t, t, y, y, c,
  y, y, t, q, q, t, y, y,
  c, y, y, t, t, y, y, c,
  c, c, y, y, y, y, c, c,
  m, c, c, m, m, c, c, m,
  c, m, m, m, m, m, m, c,
  c, c, c, m, m, c, c, c]
```

Du måste nu indentera all din kod under `for` loopen så att den är **inuti** `for` loopen.

Tips: För att indentera flera rader, markera de rader som du vill indentera och tryck sedan på `Tab` -tangenten på ditt tangentbord (vanligtvis ovanför `Caps Lock` -tangenten på tangentbordet).

main.py

```
2   for i in range(28):
      rgb = sense.color # hämta färgen från sensorn
      c = (rgb.red, rgb.green, rgb.blue)

      bild = [
        c, c, y, y, y, y, c, c,
        c, y, y, t, t, y, y, c,
        y, y, t, q, q, t, y, y,
        c, y, y, t, t, y, y, c,
        c, c, y, y, y, y, c, c,
        m, c, c, m, m, c, c, m,
        c, m, m, m, m, m, m, c,
        c, c, c, m, m, c, c, c]

      # Visa bilden
17
      sense.set_pixels(bild)
```

Längst ned i koden lägger du till en `-sleep` på en sekund i din loop:

main.py

```
    # Visa bilden

    sense.set_pixels(bild)
    sleep(1)
```

Tips: Se till att denna kodrad är indenterad i din `for` -loop.

Testa: Kör din kod och ändra färgväljaren flera gånger medan ditt projekt körs. Kontrollera att din bild uppdateras för att använda den avkända färgen vid nästa körning.

Bilden kommer att sluta uppdateras när loopen är klar så att programmet inte körs i mer än 30 sekunder.

Felsökning

Min kod har ett syntaxfel eller fungerar inte som förväntat:

- Kontrollera att din kod matchar koden i exemplen ovan
- Kontrollera att du har indenterat koden i din `for` -loop
- Kontrollera att din lista är omgiven av `[` och `]`
- Kontrollera att varje färgvariabel i listan är avgränsad med ett kommatecken

Min kod körs i längre än 30 sekunder:

- Minska antalet gånger din for loop kör, från 28 till 25 eller till och med 20.
- Minska längden på sleep, från 1 sekund till 0,5 sekunder.

Lägg till `sense.clear()` i slutet av din kod för att rensa bilden i slutet av din loop. Detta hjälper dig att se när din animation har körts färdigt.

Tips: Se till att du **inte** indenterar raden `sense.clear()` eftersom du vill att den bara ska köras en gång i slutet av din animering.

main.py

```
    # Visa bilden

      sense.set_pixels(bild)
      sleep(1)

    sense.clear()
```

Testa: Kör din kod igen. När ditt projekt har körts klart försvinner LED-matrisen, vilket gör att alla lampor blir svarta (släckta).

Felsökning

LED-matrisen blir svart varje sekund:

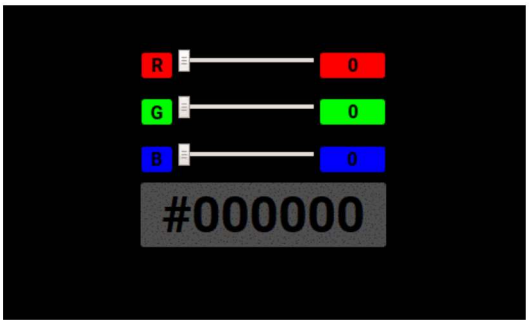
- Kontrollera att du inte har indenterat koden `sense.clear()` i din `for` -loop

Lägg till kod för att rensa LED-matrisen till en färg som du väljer. Skapa en variabel som heter `x` för att lagra din nya färg.

Du kan blanda din egen färg eller använda värdena från listan över färger för att skapa din nya `x`-färg.

RGB-färger

När vi vill representera en färg i ett datorprogram kan vi göra detta genom att definiera mängden rött, blått och grönt som kombineras för att utgöra den färgen. Dessa belopp lagras som ett tal mellan 0 och 255.

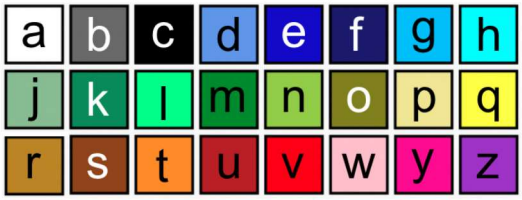


Här är ett tabell som visar några värden för olika färger:

Röd	Grön	Blå	Färg
255	0	0	Röd
0	255	0	Grön
0	0	255	Blå
255	255	0	Gul
255	0	255	Magenta
0	255	255	Cyan

Du kan hitta en fin **färgplockare att leka med på w3schools** [(https://www.w3schools.com/colors/colors_rgb.asp)](https://www.w3schools.com/colors/colors_rgb.asp).

Lista över färgvariabler



main.py

```
# Färgpalett
a = (255, 255, 255) # Vit
b = (105, 105, 105) # DimGrå
c = (0, 0, 0) # Svart
d = (100, 149, 237) # Blåklint
e = (0, 0, 205) # MediumBlå
f = (25, 25, 112) # Midnattsblå
g = (0, 191, 255) # MörkHimmelsBlå
h = (0, 255, 255) # Cyan
j = (143, 188, 143) # MörkHavsGrön
k = (46, 139, 87) # HavsGrön
l = (0, 255, 127) # VårGrön
m = (34, 139, 34) # SkogsGrön
n = (154, 205, 50) # Gulgrön
o = (128, 128, 0) # Oliv
p = (240, 230, 140) # Khaki
q = (255, 255, 0) # Gul
r = ( 184, 134, 11) # MörkGylleneRöd
s = (139, 69, 19) # SadelBrun
t = (255, 140, 0) # MörkOrange
u = (178, 34, 34) # Tegel
v = (255, 0, 0) # Röd
w = (255, 192, 203) # Rosa
y = (255, 20, 147) # MörkRosa
z = (153, 50, 204) # MörkOrkidèe
```

main.py

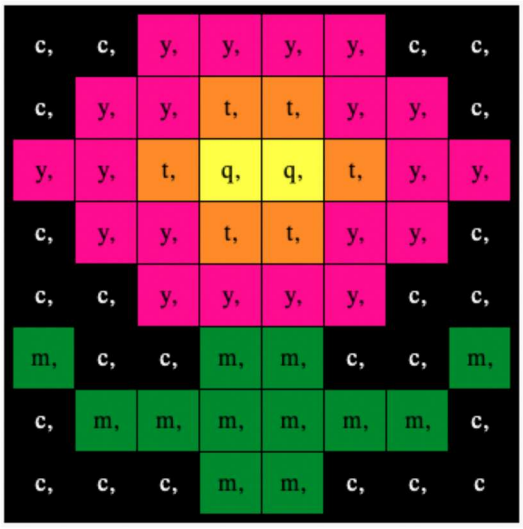
```
    # Visa bilden

      sense.set_pixels(bild)
      sleep(1)

    x = (178, 34, 34)  # välj dina egna röda, gröna, blå värden mellan 0 - 255
    sense.clear(x)
```

Test: Kör din kod igen. När ditt projekt är klart kommer LED-matrisen att rensas till din valda färg. Du kan ändra och sedan testa färgen så många gånger du vill.

Färdigt kodexempel



main.py

```
# Importera biblioteken
from sense_hat import SenseHat
from time import sleep

# Ställ in Sense HAT
sense = SenseHat()
sense.set_rotation(270)

# Ställ in färgsensorn
sense.color.gain = 60 # Set the sensitivity of the sensor
sense.color.integration_cycles = 64 # The interval at which the reading will be taken

# Lägg till färgvariabler och bild

c = (0, 0, 0) # Svart
m = (34, 139, 34) # Skogsgrön
q = (255, 255, 0) # Gul
t = (255, 140, 0) # Mörkorange
y = (255, 20, 147) # Djuprosa

for i in range(28):
  rgb = sense.color # hämta färgen från sensornr
  c = (rgb.red, rgb.green, rgb.blue)

  bild = [
    c, c, y, y, y, y, c, c,
    c, y, y, t, t, y, y, c,
    y, y, t, q, q, t, y, y,
    c, y, y, t, t, y, y, c,
    c, c, y, y, y, y, c, c,
    m, c, c, m, m, c, c, m,
    c, m, m, m, m, m, m, c,
    c, c, c, m, m, c, c, c]

  # Visa bilden

  sense.set_pixels(bild)
  sleep(1)

x = (178, 34, 34)  # välj dina egna röda, gröna, blå värden mellan 0 - 255
sense.clear(x)
```

Step 5 Skicka ditt bidrag

Du kan nu gå in i **Astro Pi Mission Zero** (https://astro-pi.org/sv/mission-zero)-utmaningen med koden du har skrivit.

Det finns några regler som din kod måste följa för att du ska kunna skicka den så att den körs på den internationella rymdstationen. Om din kod följer dem kommer reglerna längst ner i **Sense HAT-emulator** att lysa grönt när du kör programmet.

Tips: Testa din kod med några olika färginställningar (med väljaren) för att se till att den alltid fungerar korrekt.

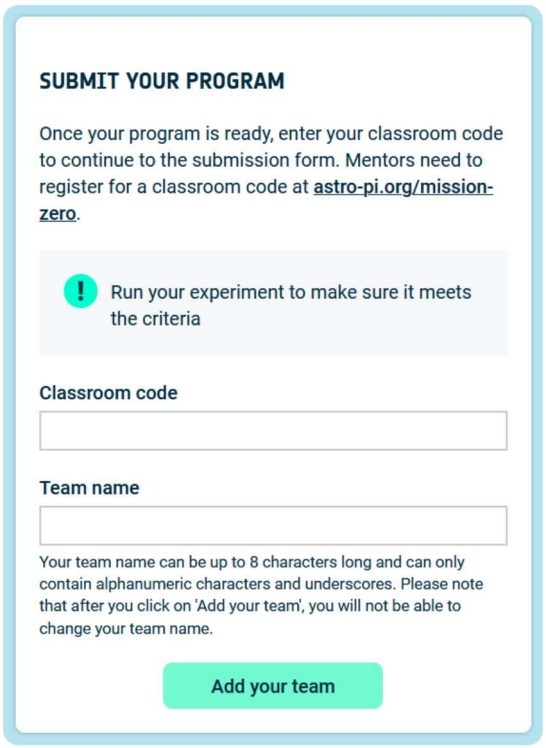
Se till att ditt bidrag följer de **officiella riktlinjerna** (https://astro-pi.org/sv/mission-zero/guidelines) för Mission Zero. Om det inte följer riktlinjerna kommer ditt program inte att kunna köras på den internationella rymdstationen.

Vänligen inkludera inte något av följande i ditt lagnamn eller kod:

- Allt som kan tolkas som olagligt, politiskt eller känsligt
- Flaggor, eftersom de kan anses vara politiskt känsliga
- Allt som hänvisar till obehag eller skada på en annan person
- Personuppgifter såsom telefonnummer, användarnamn till sociala medier och e-postadresser
- Obscena bilder

- Specialtecken eller emojis
- Dåligt språk eller svordomar

Ange din klassrumskod och lagnamn i rutan längst ner - din mentor kommer att berätta vad din kod är.



Noteringar för lärare och mentorer finns i steget **Introduktion** (https://projects.raspberrypi.org/sv-SE/projects/astro-pi-mission-zero/0).

Tryck på knappen **Lägg till ditt lag** för att ange din kod. Observera att ett program inte kan ändras när det väl har skickats in.

Din mentor kommer att få ett e-postmeddelande för att bekräfta din anmälan.

Om du vill kan du dela länken till din kod på sociala medier för att berätta för människor att den kod som du skrev kommer att köras i rymden!

Step 6 Vad härnäst: fler Astro Pi-projekt

Nu när du har avslutat ditt uppdrag, varför inte prova några fler projekt med de andra sensorerna på Astro Pi?

Om du känner dig självsäker kan du delta i **Mission Space Lab** (https://astro-pi.org/missions/space-lab/)! Skapa ett lag med två till sex personer och arbeta tillsammans som riktiga rymdforskare för att utforma ert eget experiment. De bästa idéerna som skickas in kommer att få ett Astro Pi-kit att använda för att hjälpa till med ditt uppdrag.

Alternativt kanske du vill testa något av våra andra Sense HAT-projekt:

- Lär dig **mer om Sense HAT** (https://projects.raspberrypi.org/sv-SE/projects/getting-started-with-the-sense-hat) och de andra sakerna den kan göra
- Skapa några vackra **slumpmässiga gnistor** (https://projects.raspberrypi.org/sv-SE/projects/sense-hat-random-sparkles) på Sense HAT:s LED-skärm
- Skapa ett **Flappy Astronaut** (https://projects.raspberrypi.org/sv-SE/projects/flappy-astronaut)-spel
- Utmana dina vänner med en **marmorlabyrint** (https://projects.raspberrypi.org/sv-SE/projects/sense-hat-marble-maze) spel
- Återskapa det klassiska spelet **Pong** (https://projects.raspberrypi.org/sv-SE/projects/sense-hat-pong)

Detta projekt översattes av frivilliga:

David Norlin

Tack vare frivilliga kan vi ge människor runt om i världen möjligheten att lära på sitt eget språk. Du kan hjälpa oss att nå ut till fler personer genom att bli frivillig översättare - läs mer på **rpf.io/translate** (https://rpf.io/translate).