Projekte

# Astro Pi: Mission Zero

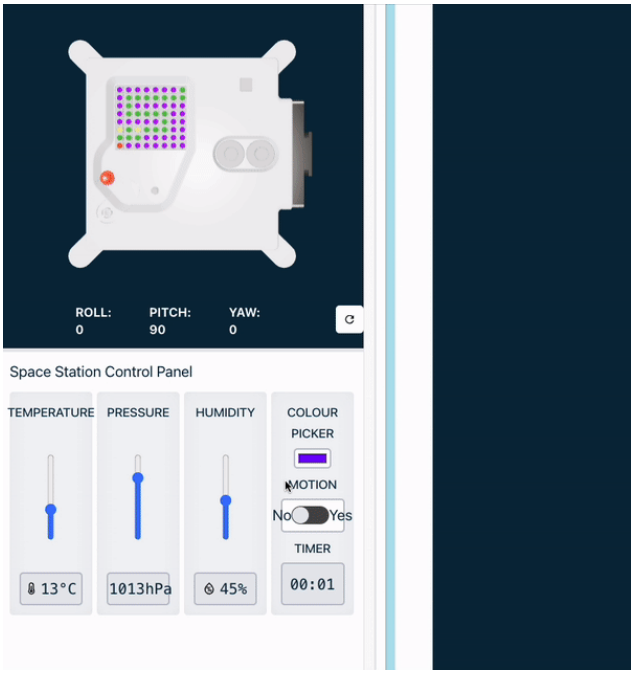
Bereite dich auf Mission Zero vor


## Schritt 1 Das wirst du machen

Schließe dieses Projekt ab, um bei Astro Pi Mission Zero mit zu machen und deinen Code im Weltraum auf einem Astro Pi Computer laufen zu lassen.

Dein Projekt setzt die Hintergrundfarbe eines Bildes auf die Farbe, die der Astro Pi erkennt. Damit wird die Internationale Raumstation (ISS) für die Astronauten an Bord bunter. Dein Code verwendet den Farbsensor auf dem Sense HAT des neuen Mark II Astro Pi-Computers, um dies zu ermöglichen.

Hier ist ein Beispiel für die Art von Programm, das du erstellen könntest, um es auf einem Astro Pi im Weltraum auszuführen.




**Was du brauchen wirst**

Du wirst den Astro Pi Emulator in einem Webbrowser verwenden, um dein Programm zu erstellen. Du brauchst keinen Astro Pi Computer.

**Astro Pi Mission Zero Kriterien**

**Wenn dein Projekt die Eignungskriterien** (https://astro-pi.org/de/mission-zero/eligibility) erfüllt, wird dein abgeschlossenes Programm auf der Internationalen Raumstation durchgeführt! Du erhältst außerdem ein spezielles Zertifikat, das zeigt, wo genau die ISS war, als dein Programm lief.

Du wirst mehr über den Astro Pi erfahren und wie du ihn steuern kannst, einschließlich:

- Erstellen von **Variablen** für für Farbe für dein Bild
- Entwerfen und anzeigen eines Bildes auf dem Sense HAT
- Ermitteln der Lichtfarbe an Bord der ISS

### Hinweise für Lehrer und Mentoren

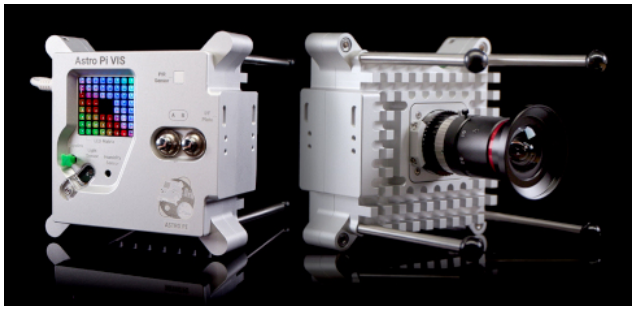
Mission Zero ist für Programmieranfänger und/oder Kinder im Grundschulalter geeignet und kann in einer einzigen 60-minütigen Sitzung auf jedem Computer mit Internetzugang abgeschlossen werden. Es sind keine spezielle Hardware oder vorherige Programmierkenntnisse erforderlich. Alles kann in einem Webbrowser erledigt werden.

Teilen Sie Ihre Schüler in Teams von ein bis zu vier Personen ein und wir zeigen ihnen wie man ein kurzes Python-Programm schreibt, um die Lichtfarbe an Bord der ISS zu ermitteln und ein Bild zu zeichnen, das diese Farbe verwendet.

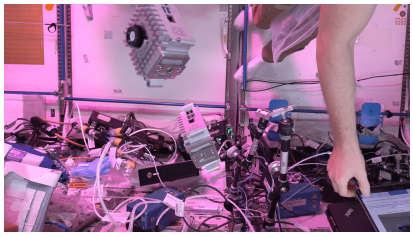
Lesen Sie die **offiziellen Richtlinien** (https://astro-pi.org/de/mission-zero/guidelines) für Mission Zero.

## Schritt 2 Was ist ein Astro Pi?

Ein Astro Pi ist ein Raspberry Pi Computer, in einem Gehäuse, das speziell für Weltraumbedingungen entwickelt wurde.




Astro Pi Computer verfügen über eine Reihe von Sensoren und Gadgets, mit denen man großartige wissenschaftliche Experimente durchführen kann. Dieser Satz von Sensoren wird als „Sense HAT“ bezeichnet (was für „Hardware Attached on Top“ steht). Das Sense HAT gibt Astro Pi die Möglichkeit, viele Daten zu „erfassen“ und verschiedene Arten von Messungen durchzuführen, von Temperatur bis zu Bewegung, und Informationen über ein 8 x 8 LED-Matrix-Display auszugeben. Die Astro Pis haben auch einen Joystick und Tasten, genau wie eine Videospielkonsole!

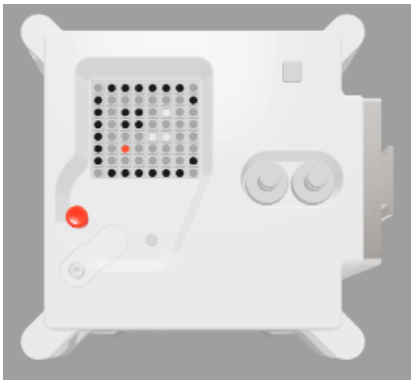


Für diese Mission wirst du den Sense HAT Emulator verwenden. Der Emulator ist eine Software, die alle Funktionen des Astro Pi in deinem Webbrowser simuliert.

## Schritt 3 Zeige ein Bild

Die LED-Matrix des Astro Pi kann Farben darstellen. In diesem Schritt zeigst du Bilder aus der Natur auf der LED-Matrix des Astro Pi an.

> Eine **LED-Matrix** ist ein Raster von LEDs, die einzeln oder als Gruppe gesteuert werden können, um verschiedene Lichteffekte zu erzeugen. Die LED-Matrix des Sense HAT verfügt über 64 LEDs, die in einem 8 x 8-Raster angeordnet sind. Die LEDs können so programmiert werden, dass sie eine breite Palette von Farben erzeugen.



Öffne das **Mission Zero-Starterprojekt** (https://missions.astro-pi.org/de/mz/code_submissions/new).

Du wirst sehen, dass einige Zeilen Code bereits automatisch erscheinen.

Dieser Code verbindet sich mit dem Astro Pi, stellt sicher, dass die LED-Anzeige des Astro Pi richtig herum angezeigt wird und richtet den Farbsensor ein. Lass den Code so dort stehen, weil du ihn noch brauchen wirst.

main.py

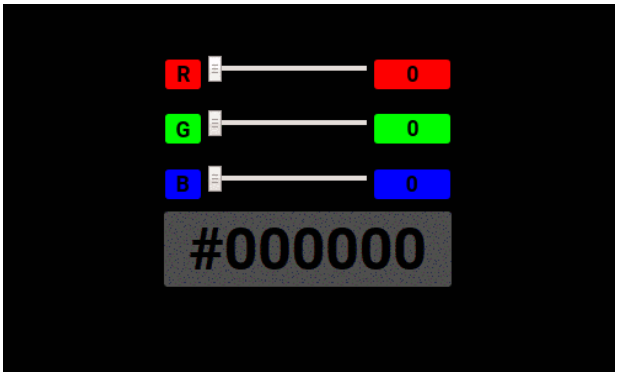
```
# Bibliotheken importieren
from sense_hat import SenseHat
from time import sleep

# Einrichten des Sense HAT
sense = SenseHat()
sense.set_rotation(270)

# Farbsensor einrichten
sense.color.gain = 60 # Stelle die Empfindlichkeit des Sensors ein
sense.color.integration_cycles = 64 # Das Intervall in dem gemessen wird
```


### RGB-Farben

Alle Farben können mit unterschiedlichen Anteilen von rot, grün und blau erzeugt werden. Informationen zu RGB-Farben findest du hier:

**RGB-Farben**

Wenn wir eine Farbe in einem Computerprogramm darstellen möchten, können wir die Mengen an Rot, Blau und Grün definieren, die kombiniert werden, um diese Farbe zu bilden. Diese Werte werden als Zahlen zwischen 0 und 255 gespeichert.



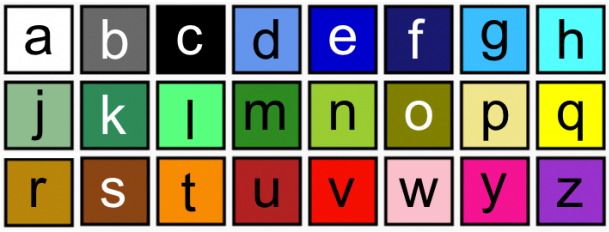

Hier ist eine Tabelle mit einigen Farbwerten:

| Rot | Grün | Blau | Farbe |
|---|---|---|---|
| 255 | 0 | 0 | Rot |
| 0 | 255 | 0 | Grün |
| 0 | 0 | 255 | Blau |
| 255 | 255 | 0 | Gelb |
| 255 | 0 | 255 | Magenta |
| 0 | 255 | 255 | Cyan |

Du findest einen praktischen **Farbwähler zum Experimentieren bei w3schools** (https://www.w3schools.com/colors/colors_rgb.asp).

Die LED-Matrix ist ein 8 x 8 Raster. Jede LED am Raster kann auf eine andere Farbe eingestellt werden. Hier ist eine Liste von Variablen für 24 verschiedene Farben. Jede Farbe hat einen Wert für Rot, Grün und Blau:

**Liste der Farbvariablen**




main.py

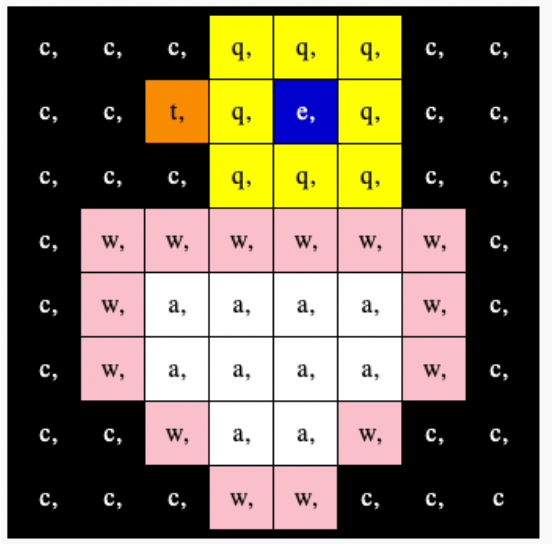
```
# Farbpalette
a = (255, 255, 255) # weiß
b = (105, 105, 105) # dunkelgrau
c = (0, 0, 0) # schwarz
d = (100, 149, 237) # kornblumenblau
e = (0, 0, 205) # mittelblau
f = (25, 25, 112) # mitternachtsblau
g = (0, 191, 255) # tiefes himmelblau
h = (0, 255, 255) # cyan
j = (143, 188, 143) # dunkles meeresgrün
k = (46, 139, 87) # meeresgrün
l = (0, 255, 127) # frühlingsgrün
m = (34, 139, 34) # waldgrün
n = (154, 205, 50) # gelbgrün
o = (128, 128, 0) # oliv
p = (240, 230, 140) # khaki
q = (255, 255, 0) # gelb
r = ( 184, 134, 11) # dunkles goldrute
s = (139, 69, 19) # sattelbraun
t = (255, 140, 0) # dunkelorange
u = (178, 34, 34) # schamott
v = (255, 0, 0) # rot
w = (255, 192, 203) # rosa
y = (255, 20, 147) # dunkelrosa
z = (153, 50, 204) # dunkelorchidee
```

**Wähle ein Bild**

**Auswählen:** Wähle ein Bild aus den folgenden Optionen, um es anzuzeigen. Python speichert die Informationen für ein Bild in einer Liste. Der Code für jedes Bild enthält die verwendeten Farbvariablen und die Liste.

Du musst den gesamten Code für dein ausgewähltes Bild **kopieren** und ihn dann in dein Projekt **einfügen**, unterhalb der Zeile mit der Aufschrift `# Farbvariablen und Bild hinzufügen`.

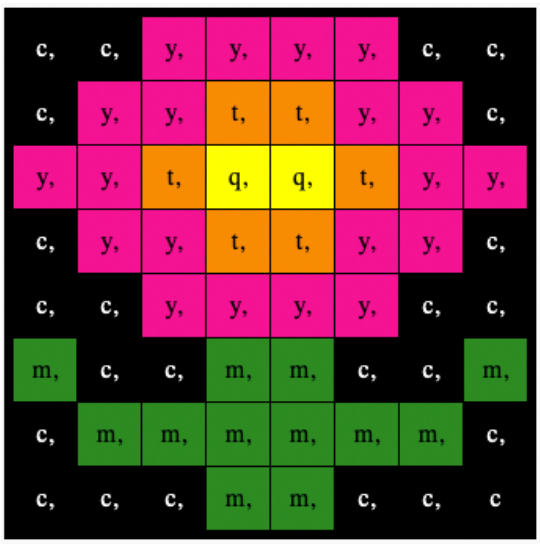
**Huhn**




main.py

```
a = (255, 255, 255) # Weiß
c = (0, 0, 0) # Schwarz
e = (0, 0, 205) # Mittelblau
q = (255, 255, 0) # Gelb
t = (255, 140, 0) # Dunkelorange
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]
```

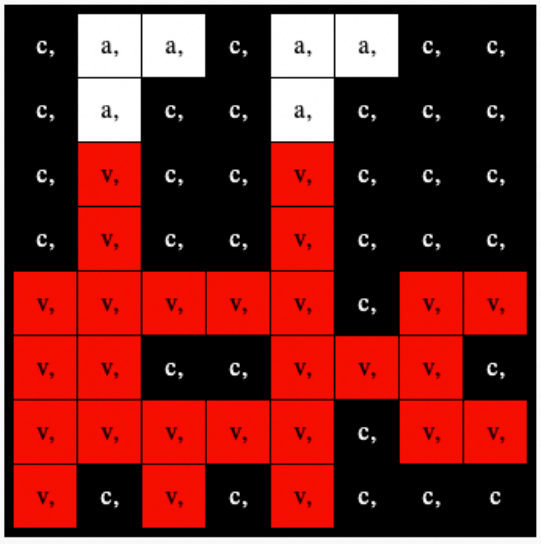
**Blume**




main.py

```
c = (0, 0, 0) # Schwarz
m = (34, 139, 34) # Waldgrün
q = (255, 255, 0) # Gelb
t = (255, 140, 0) # Dunkelorange
y = (255, 20, 147) # Dunkelrosa

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

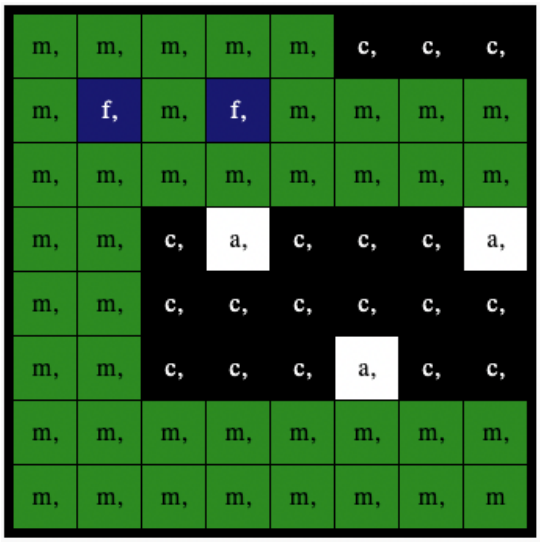
**Krabbe**




main.py

```
a = (255, 255, 255) # Weiß
c = (0, 0, 0) # Schwarz
v = (255, 0, 0) # Rot

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]
```

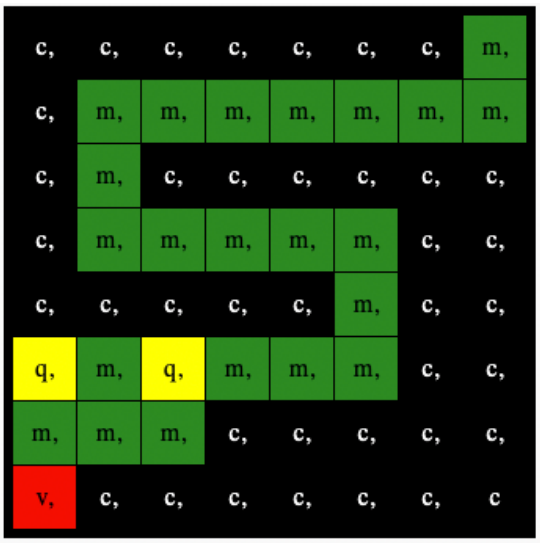
**Krokodil**




main.py

```
a = (255, 255, 255) # Weiß
c = (0, 0, 0) # Schwarz
f = (25, 25, 112) # Mitternachtsblau
m = (34, 139, 34) # Waldgrün

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]
```

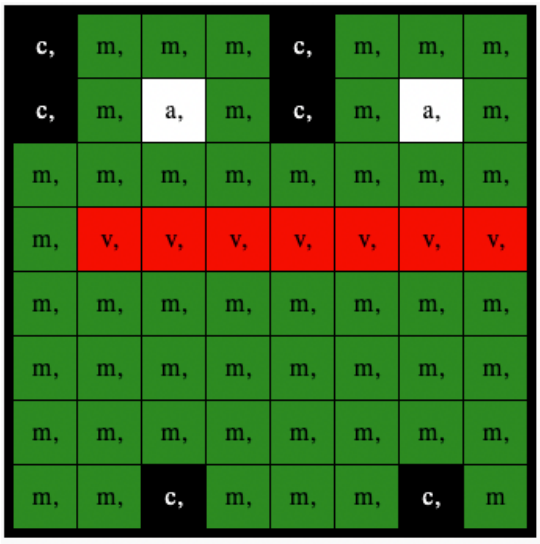
**Schlange**




main.py

```
c = (0, 0, 0) # Schwarz
 m = (34, 139, 34) # Waldgrün
 q = (255, 255, 0) # Gelb
 v = (255, 0, 0) # Rot

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]
```

**Frosch**




main.py

```
c = (0, 0, 0) # Schwarz
m = (34, 139, 34) # Waldgrün
q = (255, 255, 0) # Gelb
v = (255, 0, 0) # Rot

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]
```

**Suche:** die Zeile `# das Bild anzeigen` und füge eine Zeile Code hinzu, um dein Bild auf der LED-Matrix anzuzeigen:

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]

  # Das Bild anzeigen
  sense.set_pixels(bild)
```

Drücke **Ausführen** am unteren Rand des Editors, um dein Bild auf der LED-Matrix anzuzeigen.

**Fehlersuche**

Mein Code hat einen Syntaxfehler:

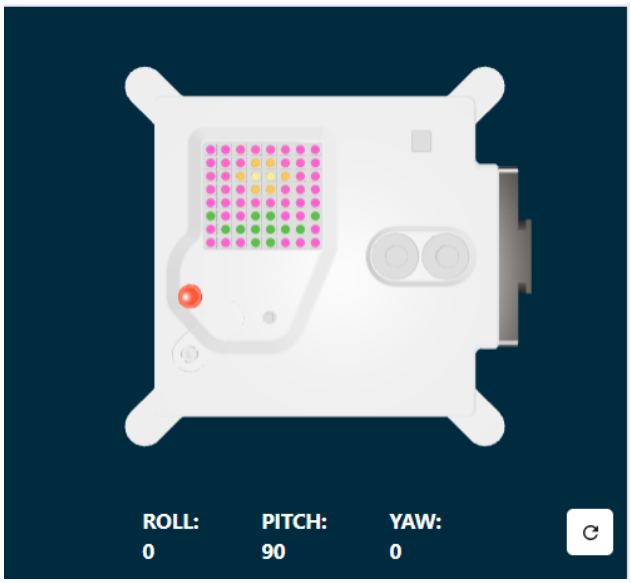
- Überprüfe, ob dein Code mit dem Code in den obigen Beispielen übereinstimmt
- Überprüfe, ob du den Code richtig eingerückt hast
- Überprüfe, ob deine Liste von `[` und `]`umgeben ist
- Überprüfe, ob die Farbvariablen in der Liste durch ein Kommas getrennt sind

Mein Bild wird nicht angezeigt:

- Überzeuge dich, dass dein `sense.set_pixels(bild)` nicht eingerückt ist

## Schritt 4 Bestimme eine Farbe

In diesem Schritt richtest du den Farbsensor ein und verwendest ihn, um die Menge an Rot, Grün und Blau zu erfassen, die den Sensor erreicht. Diese Farbe wird dann verwendet, um dein ausgewähltes Bild einzufärben. Ein Astronaut, der in einem blauen Hemd auf den Sensor zugeht, würde ein anderes Bild sehen als ein Astronaut in einem roten Hemd.




Unabhängig davon, welches Bild du wählst, verwendet der Hintergrund die Variable `c`, die auf Schwarz gesetzt ist.

Verwende den Farbsensor, um deinen Hintergrund einzufärben.

Füge Code vor der Liste mit deinem Bild hinzu, um die Farbe vom Sensor zu erhalten und ändere deine `c` Hintergrundfarbenvariable, um die Farbe zu verwenden, die der Sense HAT Farbsensor anstelle von Schwarz erfasst.

**Tipp:** Du musst keine Kommentare eingeben, die mit '#' beginnen (sie sind da, um den Code zu erklären).

main.py

```
# Farbvariablen und Bild hinzufügen

c = (0, 0, 0) # Schwarz
m = (34, 139, 34) # Waldgrün
q = (255, 255, 0) # Gelb
t = (255, 140, 0) # Dunkelorange
y = (255, 20, 147) # Dunkelrosa

rgb = sense.color # erhalte die Farbe vom Sensor
c = (rgb.red, rgb.green, rgb.blue) # verwende die Farbe

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:** Bewege den Farbregler auf eine Farbe deiner Wahl und **starte** deinen Code. Deine Hintergrundfarbe ändert sich. Wiederhole diesen Test mit einer neuen Farbe.

**Tipp:** Du musst jedes Mal auf 'Ausführen' klicken, wenn du die Farbe änderst.

**Mache eine Programmschleife**

Das Programm Astro Pi Mission Zero darf bis zu 30 Sekunden laufen. Du wirst diese Zeit nutzen, um den Farbsensor wiederholt abzufragen und das Bild zu aktualisieren.

Ihr Code verwendet eine `for`-Schleife, um 28 Mal ausgeführt zu werden. **Jedes** mal wird es:

- die neueste Farbe ermitteln
- die Hintergrundfarbe des Bildes aktualisieren
- eine Sekunde pausieren

**Finde** deine `rgb = sense.color` Codezeile.

Code darüber **hinzufügen** um deine `for`-Schleife für `28` Wiederholungen einzurichten.

main.py

```
for i in range(28):
rgb = sense.color # holt die Farbe vom Sensor
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 musst jetzt deinen gesamten Code unter der `for`-Schleife einrücken, sodass er **innerhalb** der `for`-Schleife sitzt.

**Tipp:** Um mehrere Zeilen einzurücken, markiere die Zeilen, die du einrücken möchtest, und drücke dann die Taste `Tab` auf deiner Tastatur (normalerweise über der Taste `Caps Lock` auf der Tastatur).

main.py

```
2  for i in range(28):
     rgb = sense.color # holt die Farbe vom Sensor
     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]

17   # Das Bild anzeigen

     sense.set_pixels(bild)
```

Füge am Ende deines Codes einen `Sleep` Befehl mit einer Sekunde innerhalb deiner Schleife hinzu:

main.py

```
# Das Bild anzeigen

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

**Tipp:** Stelle sicher, dass diese Codezeile innerhalb deiner `for`-Schleife eingerückt ist.

**Test:** Führe deinen Code aus und ändere die Farbauswahl mehrmals während dein Projekt läuft. Überprüfe, ob dein Bild aktualisiert wird, und die erfasste Farbe beim nächsten Durchlauf verwendet.

Das Bild wird nicht mehr aktualisiert, wenn die Schleife beendet ist, so dass das Programm nicht länger als 30 Sekunden läuft.

**Fehlersuche**

Mein Code hat einen Syntaxfehler oder läuft nicht wie erwartet:

- Überprüfe, ob dein Code mit dem Code in den obigen Beispielen übereinstimmt
- Überprüfe, ob du den Code in der `for`-Schleife richtig eingerückt hast
- Überprüfe, ob deine Liste von `[` und `]`umgeben ist
- Überprüfe, ob die Farbvariablen in der Liste durch Kommas getrennt sind

Mein Code läuft länger als 30 Sekunden:

- Verringere die Anzahl der Durchläufe deiner for-Schleife von 28 auf 25 oder sogar 20.
- Verringern Sie die Schlafdauer von 1 Sekunde auf 0,5 Sekunden.

Füge `sense.clear()` am Ende deines Codes hinzu, um das Bild am Ende deiner Schleife zu löschen. Damit siehst du, wenn deine Animation beendet ist.

**Tipp:** Stelle sicher, dass du die Codezeile **sense.clear()** `nicht` einrückst, da diese nur einmal am Ende deiner Animation ausgeführt werden soll.

main.py

```
    # Das Bild anzeigen

      sense.set_pixels(bild)
      sleep(1)

    sense.clear()
```

**Test:** Führe deinen Code erneut aus. Wenn dein Projekt fertig ist, wird die LED-Matrix gelöscht, wobei alle Lichter schwarz (aus) sind.

**Fehlersuche**

Die LED-Matrix wird jede Sekunde schwarz:

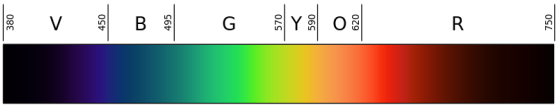
- Stelle sicher, dass du den Code `sense.clear()` in deiner `for`-Schleife nicht eingerückt hast

Füge Code hinzu, um die LED-Matrix auf eine Farbe deiner Wahl zu löschen. Erstelle eine Variable namens `x`, um deine gewählte Farbe zu speichern.

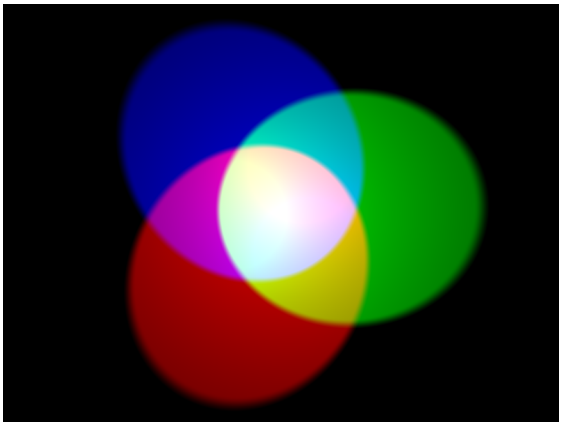
Du kannst deine eigene Farbe mischen oder die Werte aus der Farbliste verwenden, um deine neue `x`-Farbe zu erstellen.

**Farben mit Zahlen darstellen**

Die Farbe eines Objekts hängt von der Farbe des Lichts ab, das es reflektiert oder ausstrahlt. Licht kann verschiedene Wellenlängen haben und die Farbe des Lichts hängt von seiner Wellenlänge ab. Die Farbe des Lichts in Abhängigkeit von seiner Wellenlänge ist in der folgenden Abbildung zu sehen. Du kannst hier die Farben des Regenbogens erkennen.




Menschen sehen Farben aufgrund spezieller Zellen in ihren Augen. Diese Zellen werden *Zapfen* genannt. Wir haben drei Arten von Zapfen und jeder Typ erkennt entweder rotes, blaues oder grünes Licht. Daher sind alle Farben, die wir sehen, nur Mischungen der Farben Rot, Blau und Grün.



Bei der additiven Farbmischung werden drei Farben (Rot, Grün und Blau) verwendet, um andere Farben zu erzeugen. Im obigen Bild leuchten drei Scheinwerfer gleicher Helligkeit, einer für jede Farbe. Wenn keine Farbe vorhanden ist, ist das Ergebnis schwarz. Wenn alle drei Farben gemischt werden, ist das Ergebnis weiß. Wenn Rot und Grün kombiniert werden, ist das Ergebnis Gelb. Wenn Rot und Blau kombiniert werden, ist das Ergebnis Magenta. Wenn Blau und Grün kombiniert werden, ist das Ergebnis Cyan. Es ist möglich, noch mehr Farben zu erzeugen, indem die Helligkeit der drei verwendeten Originalfarben geändert wird.

Computer speichern alles in Form von Einsen und Nullen. Diese Einsen und Nullen werden oft in Achtergruppen organisiert, die **Bytes** genannt werden.

Ein einzelnes Byte kann eine beliebige Zahl von 0 bis 255 darstellen.

Wenn wir eine Farbe in einem Computerprogramm wiedergeben wollen, können wir die Mengen an Rot, Blau und Grün definieren, aus denen diese Farbe besteht. Diese Mengen werden in der Regel als ein einzelnes Byte gespeichert, also als eine Zahl zwischen 0 und 255.

Hier ist eine Tabelle mit einigen Farbwerten:

| **Rot** | **Grün** | **Blau** | **Farbe** |
|---|---|---|---|
| 255 | 0 | 0 | Rot |
| 0 | 255 | 0 | Grün |
| 0 | 0 | 255 | Blau |
| 255 | 255 | 0 | Gelb |
| 255 | 0 | 255 | Magenta |
| 0 | 255 | 255 | Cyan |

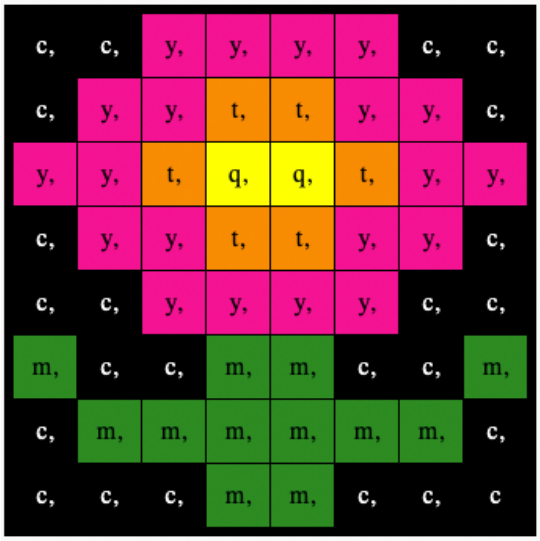
Du kannst einen praktischen **Farbwähler zum Experimentieren bei w3schools** (https://www.w3schools.com/colors/colors_rgb.asp) finden.

main.py

```
# Das Bild anzeigen

  sense.set_pixels(bild)
  sleep(1)

x = (178, 34, 34) # wähle deine eigenen roten, grünen, blauen Werte zwischen 0 - 255 sense.clear(x)
```

**Test:** Führe deinen Code erneut aus. Wenn dein Projekt fertig ist, leuchtet die LED-Matrix in der von dir gewählten Farbe. Du kannst die Farbe beliebig oft ändern und dann testen.

**Vollständiges Code-Beispiel**



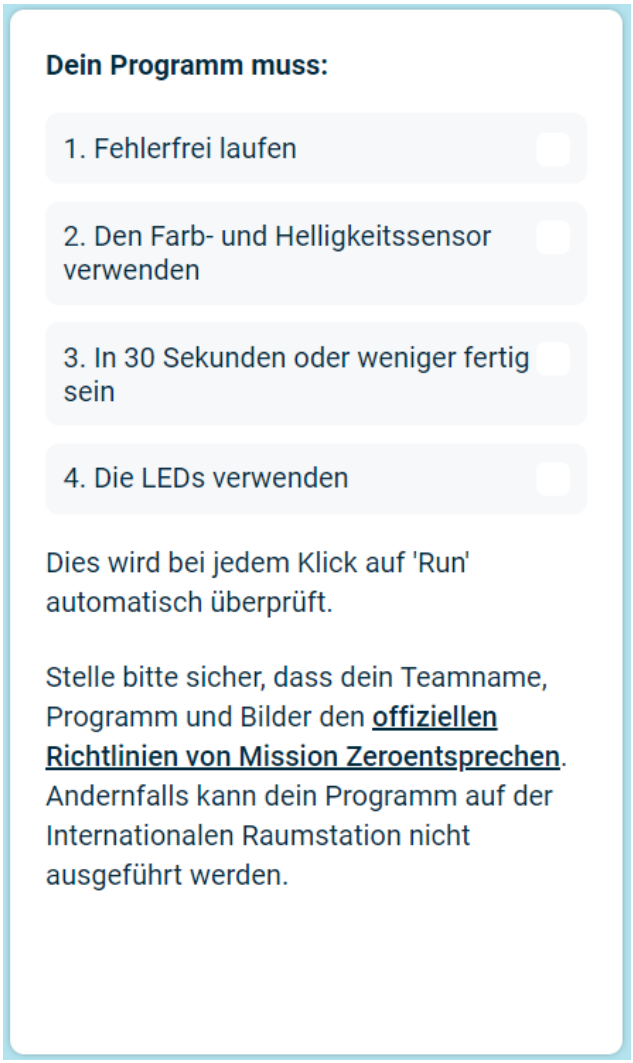

main.py

```
# Bibliotheken importieren
from sense_hat import SenseHat
from time import sleep

# Einrichten des Sense HAT
sense = SenseHat()
sense.set_rotation(270)

# Farbsensor einrichten
sense.color.gain = 60 # Stelle die Empfindlichkeit des Sensors ein
sense.color.integration_cycles = 64 # Das Intervall in dem gemessen wird

# Farbvariablen und Bild hinzufügen

c = (0, 0, 0) # Schwarz
m = (34, 139, 34) # Waldgrün
q = (255, 255, 0) # Gelb
t = (255, 140, 0) # Dunkelorange
y = (255, 20, 147) # Dunkelrosa

for i in range(28):
  rgb = sense.color # holt die Farbe vom Sensor
  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]

  # Das Bild anzeigen

  sense.set_pixels(bild)
  sleep(1)

x = (178, 34, 34) # wähle deine eigenen roten, grünen, blauen Werte zwischen 0 - 255 sense.clear(x)
```

## Schritt 5 Deinen Beitrag einsenden

Du kannst nun die **Astro Pi Mission Zero** (https://astro-pi.org/de/mission-zero) Herausforderung betreten, indem du den Code verwendest, den du geschrieben hast.

Dein Code muss einige Regeln einhalten, damit du ihn einreichen kannst um auf der Internationalen Raumstation ausgeführt zu werden. Wenn du sie richtig befolgst, werden die Regeln unten im **Sense HAT emulator** nach dem Ausführen des Programms grün angezeigt.



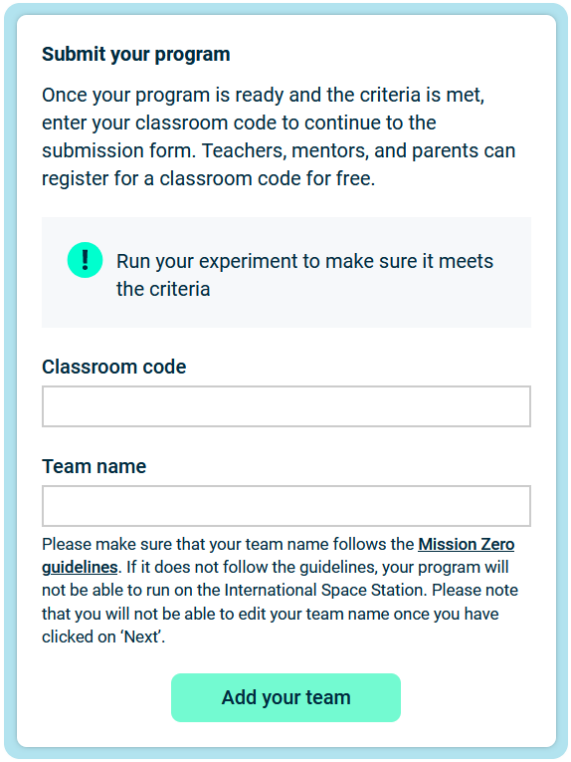

**Tipp:** Teste deinen Code mit ein paar anderen Farbeinstellungen (mit dem Picker) um sicherzustellen, dass er immer korrekt läuft.

Bitte stelle sicher, dass dein Beitrag den **offiziellen Richtlinien** (https://astro-pi.org/de/mission-zero/guidelines) für Mission Zero entspricht. Wenn es den Richtlinien nicht entspricht, kann dein Programm nicht auf der Internationalen Raumstation ausgeführt werden.

Bitte fügen Sie Folgendes nicht in Ihren Teamnamen oder Code ein:

- Alles, was als illegal, politisch oder heikel interpretiert werden könnte
- Flaggen, da sie als politisch sensibel angesehen werden können
- Alles, was auf Unannehmlichkeiten oder Schaden für eine andere Person hinweist
- Personenbezogene Daten wie Telefonnummern, Social-Media-Adressen und E-Mail-Adressen
- Obszöne Bilder
- Sonderzeichen oder Emojis
- Schimpfwörter oder Fluchen

Gib deinen Klassenraumcode und Teamnamen in das Feld unten ein – dein Mentor wird dir deinen Code mitteilen.




**Hinweise für Mentoren** sind im Schritt **Einleitung** (https://projects.raspberrypi.org/de-DE/projects/astro-pi-mission-zero/0) zu finden.

Drücke die Schaltfläche **Füge dein Team hinzu**, um deinen Code einzureichen. Bitte beachte, dass ein Programm nach dem Absenden nicht geändert werden kann.

Dein Mentor wird eine E-Mail erhalten, um deine Einsendung zu bestätigen.

Wenn du möchtest, kannst du den Link zu deinem Code auf sozialen Netzwerken teilen, um allen zu berichten, dass der Code, den du geschrieben hast, im Weltraum ausgeführt wird!

## Schritt 6 Wie geht es weiter – weitere Astro Pi-Projekte

Jetzt, wo du deine Mission beendet hast, hättest du doch sicher Lust noch ein paar Projekte mit den anderen Sensoren auf dem Astro Pi auszuprobieren?

Wenn du es dir zutraust, könntest du an der **Mission Space Lab** (https://astro-pi.org/missions/space-lab/) teilnehmen! Stelle ein Team von zwei bis sechs Personen zusammen und arbeitet wie echte Weltraumwissenschaftler zusammen, um euer eigenes Experiment zu entwerfen. Die besten eingereichten Ideen erhalten ein Astro Pi-Kit, das euch bei eurer Mission unterstützen könnte.

Alternativ möchtest du vielleicht eines unserer anderen Sense HAT-Projekte ausprobieren:

- Erfahre **mehr über den Sense HAT** (https://projects.raspberrypi.org/de-DE/projects/getting-started-with-the-sense-hat) und die anderen Möglichkeiten, die es bietet
- Erzeuge auf dem LED-Display des Sense HAT ein paar hübsche **zufällige Glitzer** (https://projects.raspberrypi.org/de-DE/projects/sense-hat-random-sparkles)
- Erstelle ein **Flappy Astronaut** (https://projects.raspberrypi.org/de-DE/projects/flappy-astronaut)-Spiel
- Fordere deine Freunde mit einem **Murmel-Labyrinth** (https://projects.raspberrypi.org/de-DE/projects/sense-hat-marble-maze)-Spiel heraus
- Erstelle das klassische Spiel **Pong** (https://projects.raspberrypi.org/de-DE/projects/sense-hat-pong) neu

---

Dieses Projekt wurde von freiwilligen Helfern übersetzt:

Karl Schuh
Lars Reime

Dank freiwilliger Helfer können wir Menschen auf der ganzen Welt die Möglichkeit geben, in ihrer eigenen Sprache zu lernen. Du kannst uns helfen, mehr Menschen zu erreichen, indem Du dich freiwillig zum Übersetzen meldest - weitere Informationen unter **rpf.io/translate** (https://rpf.io/translate).

---