Projets

Astro Pi : Mission Zero

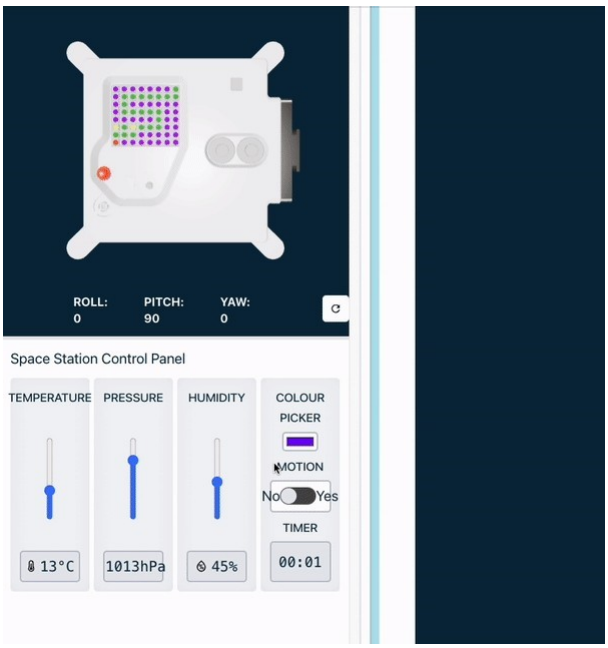
Préparation pour Mission Zero

Étape 1 Ce que tu vas faire

Réalise ce projet pour participer à Astro Pi Mission Zero (https://astro-pi.org/mission-zero ou https://esero.fr/projets/astro-pi/) et voir ton code exécuté dans l'espace sur un ordinateur Astro Pi.

Ton projet consiste à définir la couleur de fond d'une image sur la couleur détectée par Astro Pi. Ainsi, la Station Spatiale Internationale (ISS) sera plus colorée pour les astronautes à bord. Pour cela, ton code utilisera le capteur de luminosité colorimétrique sur le Sense HAT du nouvel ordinateur Mark II Astro Pi.

Voici un exemple du type de programme que tu peux exécuter sur un Astro Pi dans l'espace.



Ce qu'il te faut

Tu vas utiliser l'émulateur Astro Pi dans un navigateur Web pour créer ton programme. Tu n'as pas besoin d'un ordinateur Astro Pi.

Critères d'Astro Pi Mission Zero

Si ton projet remplit les critères d'éligibilité (https://astro-pi.org/mission-zero/eligibility), ton programme sera exécuté à bord de la Station spatiale internationale ! Tu recevras aussi un certificat spécial montrant où se trouvait précisément l'ISS lors de l'exécution de ton programme.

Tu apprendras des choses sur l'ordinateur Astro Pi et comment le contrôler, et notamment comment :

- créer des variables de couleur à utiliser dans ton image ;
- concevoir et afficher une image sur le Sense HAT ;
- capter la couleur de la lumière à bord de l'ISS.

Remarques pour les mentors

Mission Zero est un projet adapté aux débutants en programmation et/ou aux enfants en école primaire, et il peut être réalisé durant une session unique de 60 minutes sur n'importe quel ordinateur doté d'un accès à Internet. Aucun matériel spécial ni aucune compétence préalable en matière de codage ne sont requis. Tout peut être réalisé dans un navigateur Web.

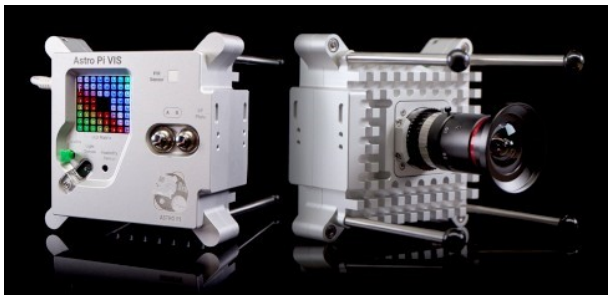
Organisez les enfants par équipes de 1 à 4 et guidez-les dans l'écriture d'un court programme en Python destiné à détecter la couleur à bord de l'ISS et à créer une image utilisant cette couleur.

Consultez les directives officielles (https://astro-pi.org/mission-zero/guidelines) de Mission Zero.

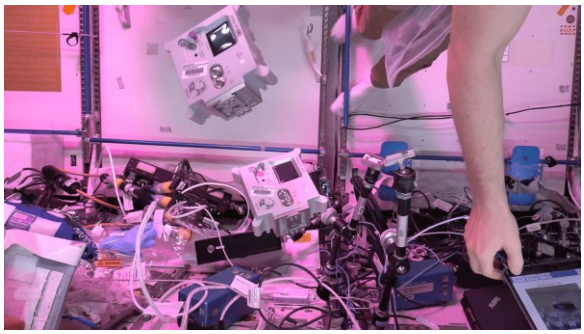
Consultez également les ressources et outils sur le site ESERO France.

Étape 2 Qu'est-ce qu'un Astro Pi ?

Un Astro Pi est un ordinateur Raspberry Pi contenu dans un boîtier spécialement conçu pour les conditions dans l'espace.



Les ordinateurs Astro Pi sont dotés d'un ensemble de capteurs et gadgets qui peuvent être utilisés pour mener de super expériences scientifiques. Cet ensemble de capteurs s'appelle un « Sense HAT » (c'est l'abréviation de « Hardware Attached on Top », qui signifie « matériel ajouté »). Le Sense HAT permet à un Astro Pi de « capter » et de réaliser de nombreuses mesures, de la température au mouvement, mais aussi d'afficher des informations grâce à un écran matriciel de 8 x 8 LED. Les Astro Pi sont aussi équipés d'un joystick et de boutons, comme une console de jeux vidéo !

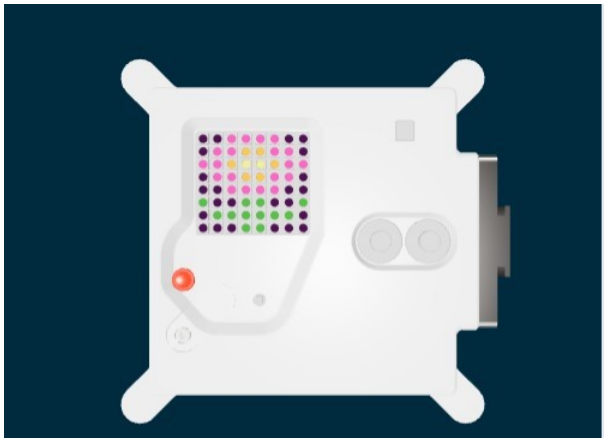


Pour cette mission, tu utiliseras l'émulateur Sense HAT, qui simule les principales fonctions de l'Astro Pi dans ton navigateur Web.

Étape 3 Affiche une image

La matrice à LED de l'Astro Pi peut afficher des couleurs. Au cours de cette étape, tu vas afficher des images représentant l'environnement sur la matrice à LED de l'Astro Pi.

> Une matrice à LED est une grille de LED qui peuvent être commandées individuellement ou en groupe pour créer différents effets de lumière. La matrice à LED du Sense HAT est dotée de 64 LED disposées selon une grille de 8 x 8. Les LED peuvent être programmées pour produire une large palette de couleurs.



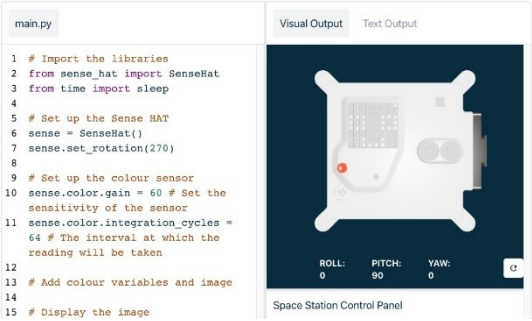
Ouvre le projet de démarrage Mission Zero (http://rpf.io/mzcode).

Tu verras que quelques lignes de code ont été ajoutées automatiquement pour toi.

Ce code permet la connexion à l'Astro Pi, garantit que l'écran à LED de l'Astro Pi s'affiche correctement et configure le capteur de couleur. Laisse ce code tel quel, tu en auras besoin.

main.py

```
# Importe les librairies
from sense_hat import SenseHat from
time import sleep

# Paramètre le Sense HAT
sense = SenseHat()
sense.set_rotation(270)

# Paramètre le récepteur de couleurs
sense.color.gain = 60 # Paramètre de sensibilité du récepteur
sense.color.integration_cycles = 64 # Intervalle de lecture
```

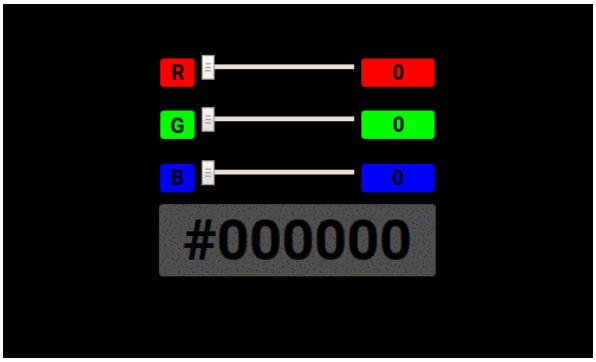


Couleurs RVB (Rouge Vert Bleu)

Il est possible de créer des couleurs en utilisant différentes proportions de rouge, de vert et de bleu. Tu en apprendras plus sur les couleurs RVB ici :

Couleurs RVB

Quand on veut représenter une couleur dans un programme informatique, il faut définir les quantités de rouge, de bleu et de vert qui sont combinées pour obtenir cette couleur. Ces quantités sont stockées sous forme de nombres compris entre 0 et 255.



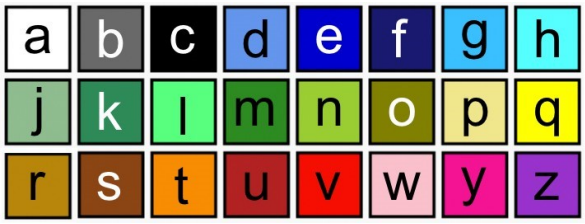
Voici un tableau avec quelques valeurs de couleur :

Rouge	Vert	Bleu	Couleur obtenue
255	0	0	Rouge
0	255	0	Vert
0	0	255	Bleu
255	255	0	Jaune
255	0	255	Magenta
0	255	255	Cyan

Tu trouveras un sélecteur de couleur sympa pour t'amuser sur w3schools (https://www.w3schools.com/colors/colors_rgb.asp).

La matrice à LED est une grille de 8 x 8. Chaque LED de cette grille peut être définie sur une couleur différente. Voici une liste de variables pour 24 couleurs différentes. Chaque couleur dispose d'une valeur pour le rouge, le vert et le bleu :

Liste de variables de couleur



main.py

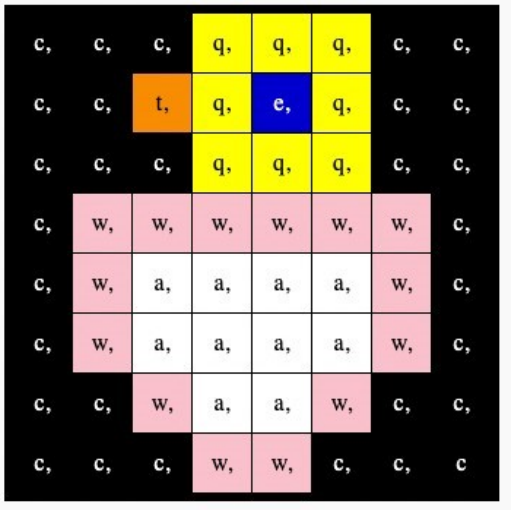
```
# Palette de couleurs
a = (255, 255, 255) # White
b = (105, 105, 105) # DimGray
c = (0, 0, 0) # Black
d = (100, 149, 237) # CornflowerBlue
e = (0, 0, 205) # MediumBlue
f = (25, 25, 112) # MidnightBlue
g = (0, 191, 255) # DeepSkyBlue
h = (0, 255, 255) # Cyan
j = (143, 188, 143) # DarkSeaGreen
k = (46, 139, 87) # SeaGreen
l = (0, 255, 127) # SpringGreen
m = (34, 139, 34) # ForestGreen
n = (154, 205, 50) # YellowGreen
o = (128, 128, 0) # Olive
p = (240, 230, 140) # Khaki
q = (255, 255, 0) # Yellow
r = (184, 134, 11) # DarkGoldenrod
s = (139, 69, 19) # SaddleBrown
t = (255, 140, 0) # DarkOrange
u = (178, 34, 34) # Firebrick
v = (255, 0, 0) # Red
w = (255, 192, 203) # Pink
y = (255, 20, 147) # DeepPink
z = (153, 50, 204) # DarkOrchid
```

Choisis une image

Choisis : sélectionne une image à afficher parmi les options ci-dessous. Python stocke les informations d'une image dans une liste. Le code de chaque image comprend les variables de couleur utilisées et cette liste.

Tu dois copier tout le code de l'image choisie et le coller dans ton projet sous la ligne qui dit `#Add colour variables and image.`

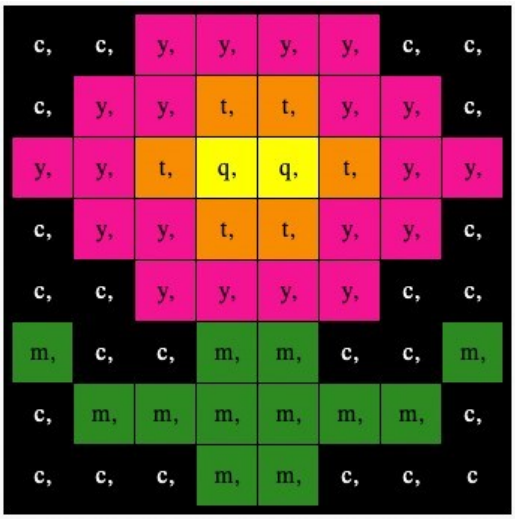
Poulet



main.py

```
a = (255, 255, 255) # White
c = (0, 0, 0) # Black
e = (0, 0, 205) # MediumBlue
q = (255, 255, 0) # Yellow
t = (255, 140, 0) # DarkOrange
w = (255, 192, 203) # Pink

image = [
  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]
```

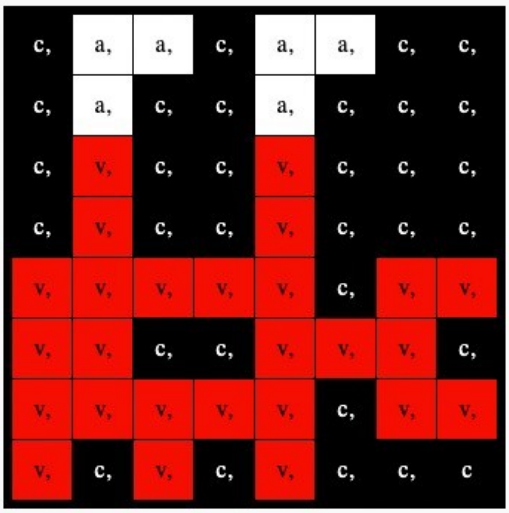
Fleur



main.py

```
c = (0, 0, 0) # Black
m = (34, 139, 34) # ForestGreen
q = (255, 255, 0) # Yellow
t = (255, 140, 0) # DarkOrange
y = (255, 20, 147) # DeepPink

image = [
  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]
```

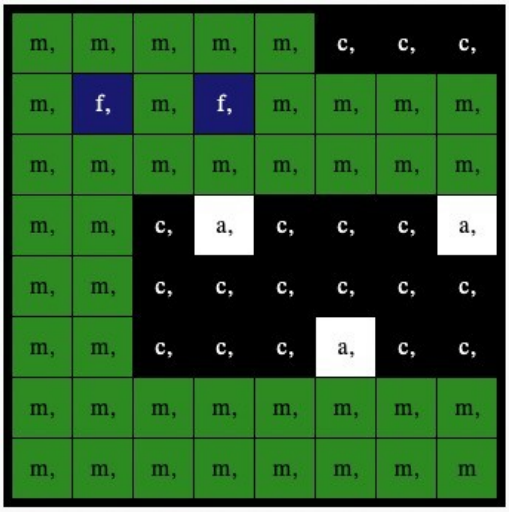
Crabe



main.py

```
a = (255, 255, 255) # White
c = (0, 0, 0) # Black
v = (255, 0, 0) # Red

image = [
  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]
```

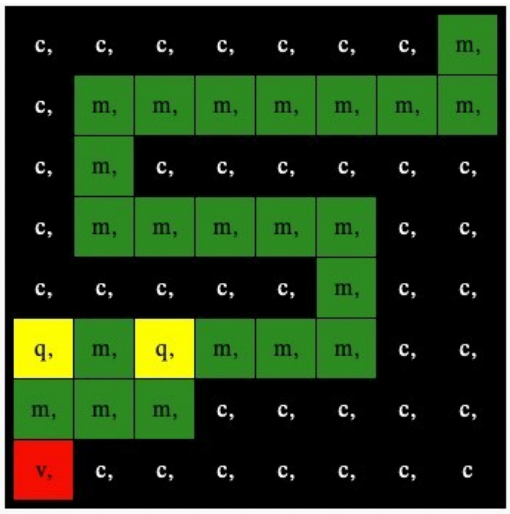
Crocodile



main.py

```
a = (255, 255, 255) # White
c = (0, 0, 0) # Black
f = (25, 25, 112) # MidnightBlue
m = (34, 139, 34) # ForestGreen

image = [
  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]
```

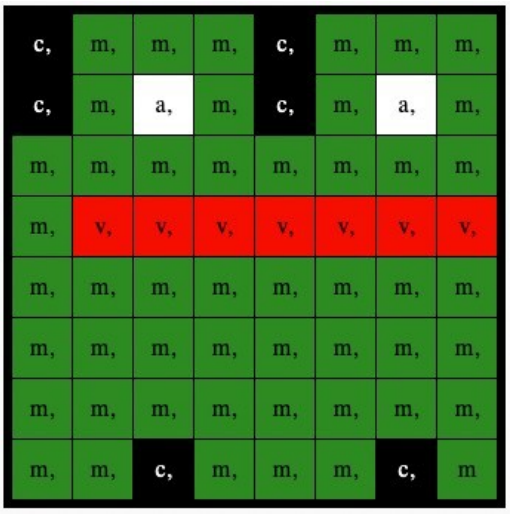
Serpent



main.py

```
c = (0, 0, 0) # Black
 m = (34, 139, 34) # ForestGreen
 q =  (255,  255, 0) # Yellow
 v =  (255,  0, 0) # Red

image = [
  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]
```

Grenouille



main.py

```
c = (0, 0, 0) # Black
m = (34, 139, 34) # ForestGreen
q = (255, 255, 0) # Yellow
v = (255, 0, 0) # Red

image = [
  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]
```

Trouve :la ligne qui dit `#Display the image` et ajoute une ligne de code pour afficher ton image sur la matrice à LED :

main.py

```
image = [
  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]

# Display the image
sense.set_pixels(image)
```

Appuie sur Run au bas de l'éditeur pour afficher ton image sur la matrice à LED.

Débogage

Mon code contient une erreur de syntaxe :

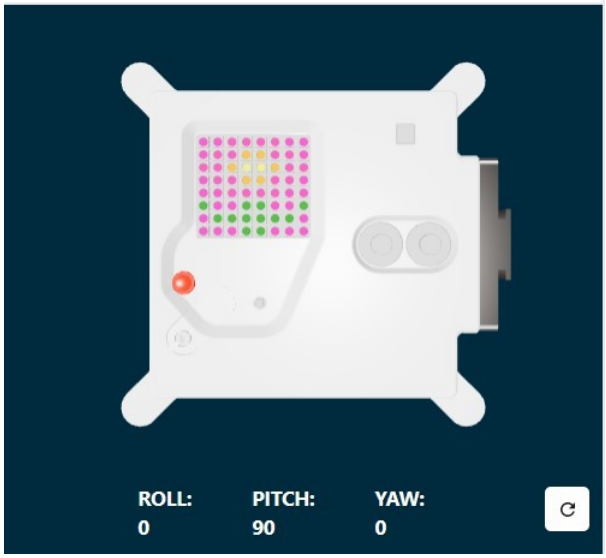
- Vérifie que ton code correspond au code dans les exemples ci-dessus.
- Vérifie que tu as indenté le code dans ta liste.
- Vérifie que ta liste est entourée par `[` et `]`.
- Vérifie que les variables de couleur dans la liste sont séparées

par une virgule. Mon image ne s'affiche pas :

- Vérifie que ton code `sense.set_pixels(image)` n'est pas indenté.

Étape 4 Détecte une couleur

Durant cette étape, tu vas configurer le capteur de luminosité colorimétrique et l'utiliser pour détecter la quantité de rouge, de vert et de bleu qui atteint le capteur. Cette couleur sera alors appliquée à l'image que tu as choisie. Si un astronaute habillé en bleu passe devant le capteur, il ne verra pas la même image qu'un astronaute habillé en rouge.



Quelle que soit l'image choisie, le fond utilise la variable `c`, qui correspond à la couleur noire.

Utilise le capteur de couleur pour colorer le fond de ton image.

Ajoute le code avant la liste de ton image pour obtenir la couleur du capteur et change la variable de couleur `c` de ton fond d'image pour utiliser la couleur détectée par le capteur de couleur du Sense HAT au lieu du noir.

Conseil : tu n'as pas besoin de saisir les commentaires qui commencent par « # » (ils sont là pour expliquer le code). main.py

```
# Add colour variables and image

c = (0, 0, 0) # Black
m = (34, 139, 34) # ForestGreen
q = (255, 255, 0) # Yellow
t = (255, 140, 0) # DarkOrange
y = (255, 20, 147) # DeepPink

rgb = sense.color # get the colour from the sensor
c = (rgb.red, rgb.green, rgb.blue) # use the sensed colour

image = [
  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 : déplace le curseur jusqu'à la couleur de ton choix puis exécute ton code. La couleur de fond de ton image change. Répète ce test avec une nouvelle couleur

Conseil : tu dois cliquer sur « Run » chaque fois que tu changes de couleur.

Boucle ton programme

Le programme Astro Pi Mission Zero peut tourner pendant 30 secondes au maximum. Tu vas utiliser cette durée pour vérifier le capteur de couleur et mettre à jour l'image de manière répétée.

Ton code va utiliser une boucle `for` pour s'exécuter 28 fois. À chaque fois, il va :

- détecter la dernière couleur ;
- mettre à jour la couleur de fond de l'image ;
- faire une pause d'une seconde.

Trouve ta ligne de code `rgb = sense.color`.

Ajoute au-dessus le code de configuration de ta boucle `for`

pour 28 répétitions. main.py

```
for i in range(28):
rgb = sense.color # get the colour from the sensor c =
(rgb.red, rgb.green, rgb.blue)

image = [
  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]
```

Tu dois maintenant indenter tout ton code sous la boucle `for` de manière à ce qu'il soit inclus dans la boucle `for`.

Conseil : Pour indenter plusieurs lignes, mets en surbrillance les lignes que tu veux indenter et appuie sur la touche `Tab` de ton clavier (généralement au-dessus de la touche `Verr Maj` du clavier).

main.py

```
for i in range(28):
  rgb = sense.color # get the colour from the sensor c =
  (rgb.red, rgb.green, rgb.blue)

  image = [
    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]

  # Display the image

  sense.set_pixels(image)
```

Au bas de ton code, ajoute un `sleep` d'une seconde dans ta boucle :

```
# Display the image

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

Conseil : assure-toi que cette ligne de code est indentée

Test : exécute ton code et modifie le sélecteur de couleur à plusieurs reprises pendant l'exécution de ton projet. Vérifie que ton image se met à jour et utilise la couleur détectée lors de son exécution suivante.

L'image arrête de se mettre à jour lorsque la boucle se termine, de sorte que le programme ne s'exécute pas pendant plus de 30 secondes.

Débogage

Mon code contient une erreur de syntaxe ou ne s'exécute pas comme prévu :

- Vérifie que ton code correspond au code dans les exemples ci-dessus.
- Vérifie que tu as indenté le code dans ta boucle `for`.
- Vérifie que ta liste est entourée par `[` et `]`.
- Vérifie que les variables de couleur dans la liste sont séparées

par une virgule. Mon code s'exécute pendant plus de

30 secondes :

- Diminue le nombre d'exécutions de ta boucle de 28 à 25 ou même 20.
- Diminue la longueur de la pause de 1 à 0,5 seconde.

Ajoute `sense.clear()` à la fin de ton code pour effacer l'image à la fin de ta boucle. Cela t'aidera à voir quand l'exécution de ton animation se termine.

Conseil : assure-toi de ne pas indenter la ligne de code `sense.clear()` car elle ne doit s'exécuter qu'une seule fois à la fin de ton animation.

main.py

```
# Display the image

  sense.set_pixels(image)
  sleep(1)

sense.clear()
```

Test : exécute ton code de nouveau. Lorsque ton projet a fini de s'exécuter, la matrice à LED s'efface en éteignant toutes les lumières.

Débogage

La matrice à LED s'éteint toutes les secondes :

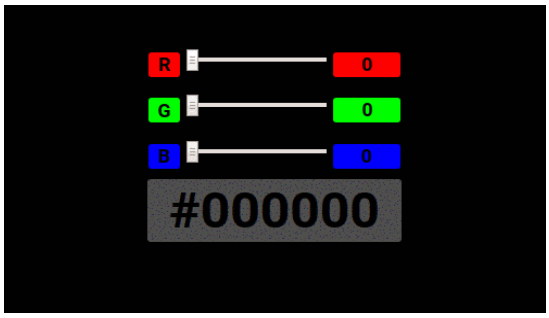
- Vérifie si tu n'as pas indenté la ligne de code `sense.clear()` dans ta boucle `for`.

Ajoute le code pour effacer la matrice à LED à une couleur de ton choix. Crée une variable appelée `x` pour stocker ta nouvelle couleur.

Tu peux réaliser ton propre mélange ou utiliser les valeurs de la liste de couleurs pour créer ta nouvelle couleur `x`.

Couleurs RVB

Quand on veut représenter une couleur dans un programme informatique, il faut définir les quantités de rouge, de bleu et de vert qui sont combinées pour obtenir cette couleur. Ces quantités sont stockées sous forme de nombres compris entre 0 et 255.

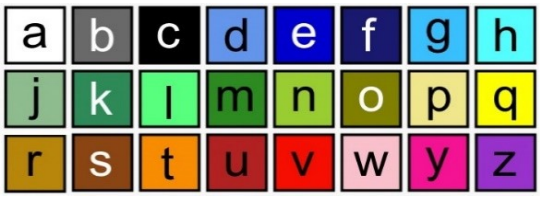


Voici un tableau avec quelques valeurs de couleur :

Rouge	Vert	Bleu	Couleur obtenue
255	0	0	Rouge
0	255	0	Vert
0	0	255	Bleu
255	255	0	Jaune
255	0	255	Magenta
0	255	255	Cyan

Tu trouveras un sélecteur de couleur sympa pour t'amuser sur w3schools (https://www.w3schools.com/colors/colors_rgb.asp).

Liste de variables de couleur



main.py

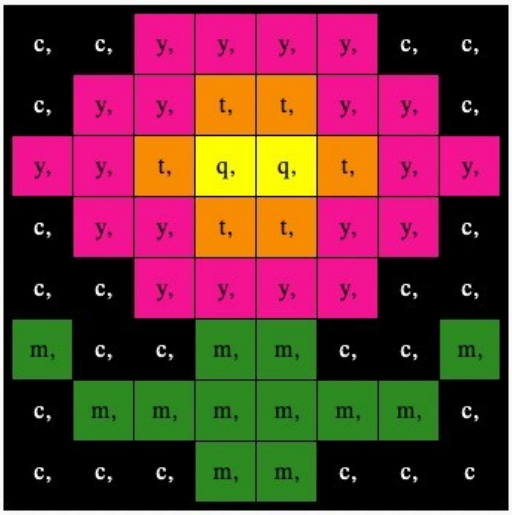
```
# Colour palette
a = (255, 255, 255) # White
b = (105, 105, 105) # DimGray
c = (0, 0, 0) # Black
d = (100, 149, 237) # CornflowerBlue
e = (0, 0, 205) # MediumBlue
f = (25, 25, 112) # MidnightBlue
g = (0, 191, 255) # DeepSkyBlue
h = (0, 255, 255) # Cyan
j = (143, 188, 143) # DarkSeaGreen
k = (46, 139, 87) # SeaGreen
l = (0, 255, 127) # SpringGreen
m = (34, 139, 34) # ForestGreen
n = (154, 205, 50) # YellowGreen
o = (128, 128, 0) # Olive
p = (240, 230, 140) # Khaki
q = (255, 255, 0) # Yellow
r = (184, 134, 11) # DarkGoldenrod
s = (139, 69, 19) # SaddleBrown
t = (255, 140, 0) # DarkOrange
u = (178, 34, 34) # Firebrick
v = (255, 0, 0) # Red
w = (255, 192, 203) # Pink
y = (255, 20, 147) # DeepPink
z = (153, 50, 204) # DarkOrchid
```

main.py

```
# Display the image

  sense.set_pixels(image)
  sleep(1)

x = (178, 34, 34) # choose your own red, green, blue values between 0 - 255 sense.clear(x)
```

Test : exécute ton code de nouveau. Lorsque ton projet a fini de s'exécuter, la matrice à LED s'efface et affiche la couleur choisie. Tu peux modifier et tester la couleur autant de fois que tu le souhaites.

Exemple de code terminé

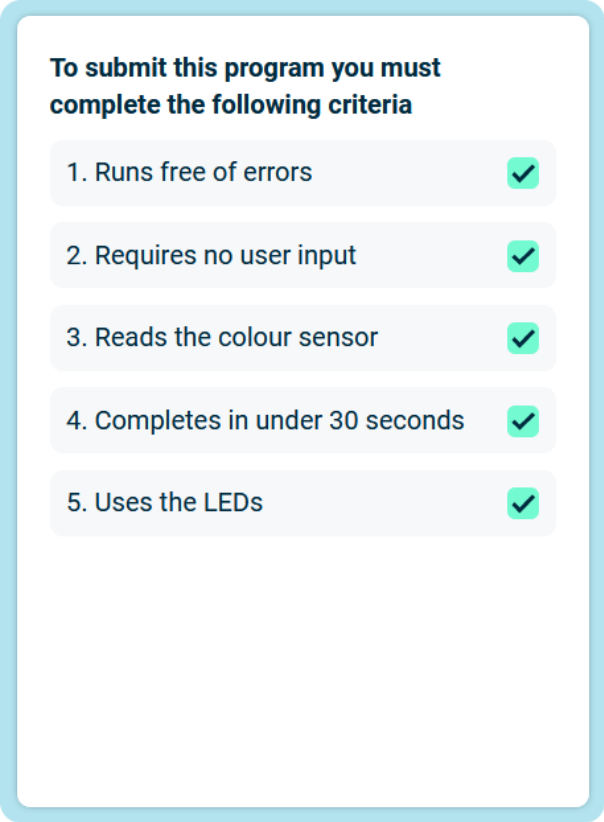


main.py

```
# Import the libraries
from sense_hat import SenseHat
from time import sleep

# Set up the Sense HAT sense
= SenseHat()
sense.set_rotation(270)

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

# Add colour variables and image

c = (0, 0, 0) # Black
m = (34, 139, 34) # ForestGreen
q = (255, 255, 0) # Yellow
t = (255, 140, 0) # DarkOrange
y = (255, 20, 147) # DeepPink

for i in range(28):
  rgb = sense.color # get the colour from the sensor c =
  (rgb.red, rgb.green, rgb.blue)

  image = [
    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] #

  Display the image

  sense.set_pixels(image)
  sleep(1)

x = (178, 34, 34) # choose your own red, green, blue values between 0 - 255 sense.clear(x)
```

Étape 5 Soumets ton projet

Ton code doit suivre certaines règles pour que tu puisses le soumettre afin qu'il soit exécuté à bord de la Station spatiale internationale. Si ton code suit ces règles, elles s'affichent en vert au bas de l'émulateur Sense HAT lorsque tu exécutes le programme.



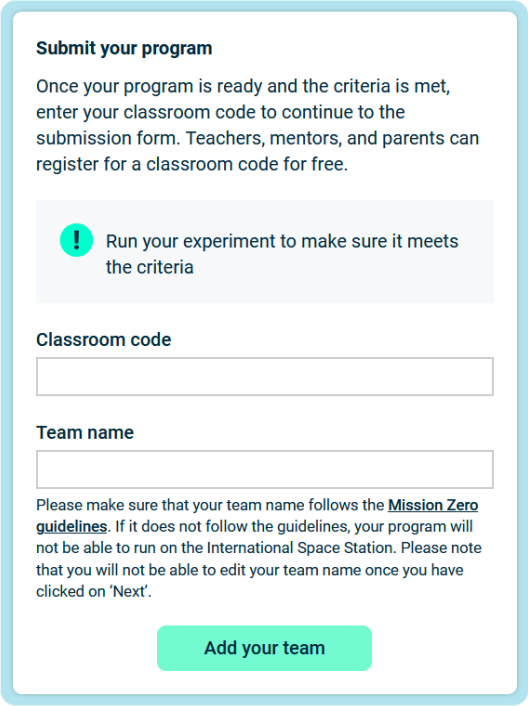
Conseil : teste ton code avec quelques réglages de couleur différents (à l'aide du sélecteur) pour t'assurer qu'il s'exécute toujours correctement.

Assure-toi de suivre les directives officielles (https://astro-pi.org/mission-zero/guidelines) de Mission Zero. S'il ne suit pas les directives, ton programme ne pourra pas s'exécuter à bord de la Station spatiale internationale.

Le nom de ton équipe ou ton code ne doivent contenir aucun des éléments suivants :

- Quoi que ce soit susceptible d'être interprété comme étant de nature illégale, politique ou sensible
- Des drapeaux, car ils peuvent être considérés comme politiquement sensibles
- Quoi que ce soit de désagréable ou nuisible pour autrui
- Des données personnelles comme un numéro de téléphone, un lien vers un réseau social ou une adresse e-mail
- Des images obscènes
- Des caractères spéciaux ou des émojis
- Des gros mots ou des injures

Saisis le code de ta classe et le nom de ton équipe dans les champs au bas de la page — ton mentor te les donnera.



Des remarques pour les mentors sont proposées dans l'étape Introduction (https://projects.raspberrypi.org/en/projects/astro-pi-mission-zero/0).

Appuie sur le bouton Add your team pour soumettre ton code. Note qu'il est impossible de modifier un programme une fois qu'il a été soumis.

Ton mentor recevra un e-mail confirmant la soumission de ton programme.

Si tu le souhaites, tu peux partager le lien vers ton code sur les réseaux sociaux pour expliquer aux gens que le code que tu as créé sera exécuté dans l'espace !

Étape 6 Et ensuite ? — Autres projets Astro Pi

Maintenant que tu as terminé ta mission, pourquoi ne pas te lancer dans de nouveaux projets utilisant les autres capteurs de l'Astro Pi ?

Si tu te sens capable de relever le défi, participe au projet Mission Space Lab (https://astro-pi.org/missions/space-lab/ ou sur https://esero.fr/projets/astro-pi/) ! Constitue une équipe de deux à six personnes et travaillez ensemble comme de vrais scientifiques spatiaux à la conception de votre propre expérience. Les équipes qui proposeront les meilleures idées seront récompensées par un kit Astro Pi pour les aider dans leur mission.

Tu peux aussi t'investir dans l'un de nos autres projets Sense HAT :

- Découvres-en plus sur le Sense HAT (https://projects.raspberrypi.org/en/projects/getting-started-with-the-sense-hat) et sur tout ce qu'il peut faire
- Crée de jolis scintillements aléatoires (https://projects.raspberrypi.org/en/projects/sense-hat-random-sparkles) sur l'écran à LED du Sense HAT
- Crée un jeu de Flappy l'astronaute (https://projects.raspberrypi.org/en/projects/flappy-astronaut)
- Défie tes amis au jeu du labyrinthe de billes (https://projects.raspberrypi.org/en/projects/sense-hat-marble-maze)
- Recrée le classique jeu Pong (https://projects.raspberrypi.org/en/projects/sense-hat-pong)