Projets

Astro Pi : Mission Zero

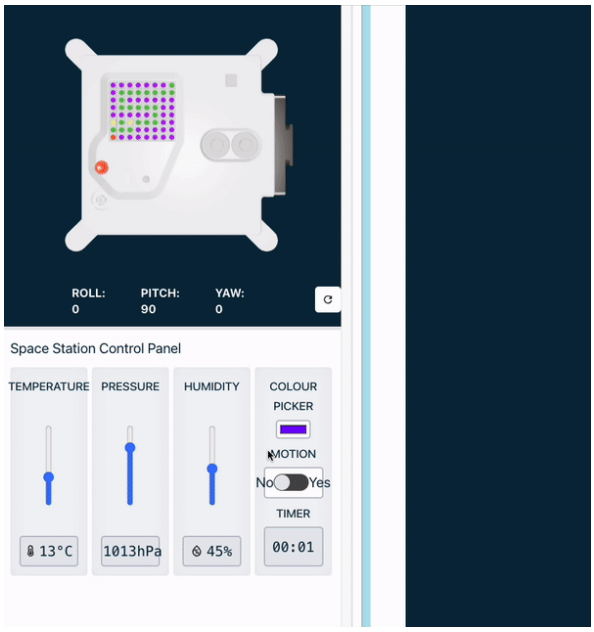
Se préparer pour Mission Zero

Étape 1 Ce que tu vas faire

Termine ce projet pour participer au défi Astro Pi Mission Zero et exécuter ton code dans l'espace sur un ordinateur Astro Pi.

Ton projet va définir la couleur d'arrière-plan d'une image à la couleur que l'Astro Pi détecte. La Station Spatiale Internationale (ISS) sera ainsi plus colorée pour les astronautes à bord. Ton code utilisera le capteur de luminosité des couleurs du nouveau Sense HAT de l'ordinateur Astro Pi Mark II pour y parvenir.

Voici un exemple du type de programme que tu pourrais créer.



Ce dont tu auras besoin

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'éligibilité de l'Astro Pi Mission Zero

Si ton projet répond aux critères d'éligibilité (https://astro-pi.org/fr/mission-zero/eligibility), ton programme terminé sera exécuté sur la Station Spatiale Internationale ! Tu recevras également un certificat spécial qui indique exactement où se trouvait l'ISS lorsque ton programme a été exécuté.

Tu découvriras l'Astro Pi et apprendras à l'utiliser et à faire les actions suivantes :

- Créer des **variables** de couleur à utiliser dans ton image
- Créer et afficher une image sur le Sense HAT
- Détecter la couleur de la lumière à bord de l'ISS

Notes pour les mentors

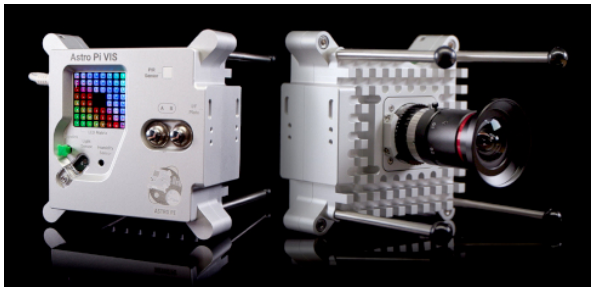
Mission Zero est adapté aux débutants et/ou aux enfants de l'école primaire et peut être réalisé en une session d'une heure sur n'importe quel ordinateur avec accès à Internet. Aucun matériel spécial ni aucune compétence préalable en codage ne sont nécessaires. Tout peut être réalisé dans un navigateur web.

Regroupez les élèves en équipes de quatre au maximum et laissez-nous les guider dans l'écriture d'un court programme Python pour détecter la couleur à bord de l'ISS et créer une image qui utilise cette couleur.

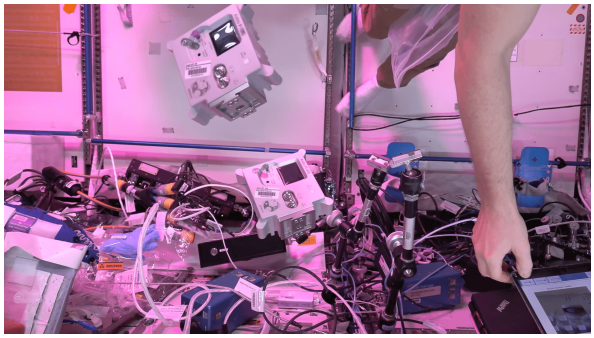
Consultez les **directives officielles** (https://astro-pi.org/fr/mission-zero/guidelines) de Mission Zero.

Étape 2 Qu’est-ce qu’un Astro Pi ?

Un Astro Pi est un ordinateur Raspberry Pi intégré dans un boîtier spécialement conçu pour les conditions spatiales.



Les ordinateurs Astro Pi sont livrés avec un ensemble de capteurs et de gadgets qui peuvent être utilisés pour réaliser de grandes expériences scientifiques. Cet ensemble de capteurs est appelé « Sense HAT » (qui signifie « Hardware Attached on Top »). Le Sense HAT permet à l’Astro Pi de « détecter » et d’effectuer de nombreux types de mesures, de la température au mouvement, et de fournir des informations à l’aide d’un écran matriciel à LED 8 x 8. Les Astro Pi disposent également d’un joystick et de boutons, comme une console de jeu vidéo !

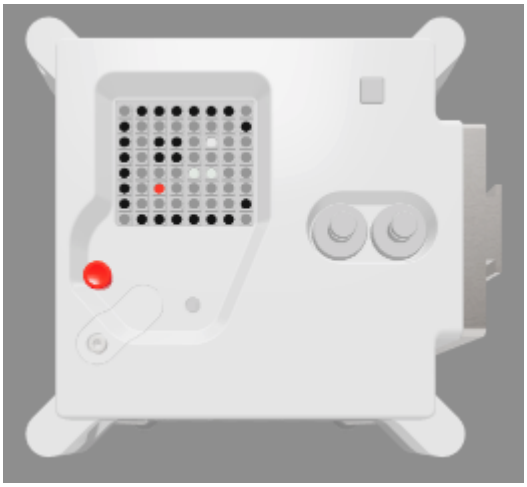


Pour cette mission, tu utiliseras l’émulateur Sense HAT. Cet émulateur simule les principales fonctions de l’Astro Pi dans ton navigateur web.

Étape 3 Afficher une image

La matrice LED de l'Astro Pi peut afficher des couleurs. Dans cette étape, tu vas afficher des images de la nature sur la matrice LED de l'Astro Pi.

> Une **matrice LED** est une grille de LED qui peuvent être contrôlées individuellement ou en groupe pour créer différents effets de lumière. La matrice LED du Sense HAT comporte 64 LED affichées dans une grille de 8 x 8. Les LED peuvent être programmées pour produire une large gamme de couleurs.



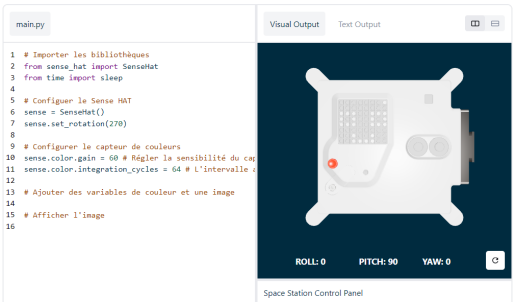
Ouvre le **projet de démarrage de Mission Zero** (https://missions.astro-pi.org/fr/mz/code_submissions/new).

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

Ce code se connecte à l'Astro Pi et fait en sorte que l'écran LED de l'Astro Pi s'affiche correctement et effectue la configuration du capteur de couleurs. Laisse ce code ici car tu en auras besoin.

main.py

```
# Importer les bibliothèques
from sense_hat import SenseHat
from time import sleep

# Configuer le Sense HAT
sense = SenseHat() s
ense.set_rotation(270)

# Configurer le capteur de couleurs
sense.color.gain = 60 # Régler la sensibilité du capteur
sense.color.integration_cycles = 64 # L'intervalle auquel la mesure est effectuée
```

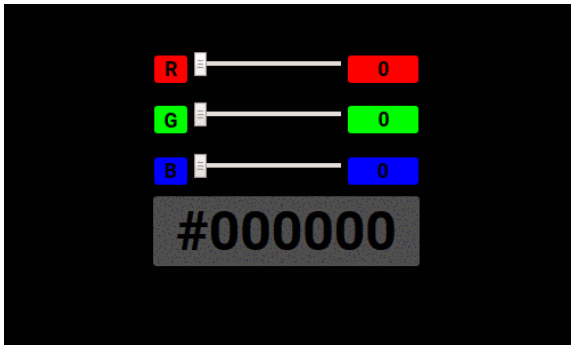


Les couleurs RVB

Tu peux créer des couleurs en utilisant différentes valeurs de rouge, vert et bleu. Tu peux découvrir les couleurs RVB ici :

Les couleurs RVB

Lorsque nous voulons représenter une couleur dans un programme informatique, nous pouvons le faire en définissant les quantités de rouge, de bleu et de vert qui composent cette couleur. Ces quantités sont généralement stockés en un seul octet et donc en un nombre compris entre 0 et 255.



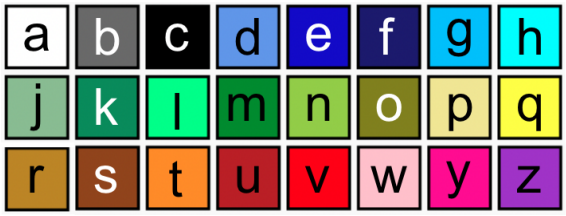
Voici un tableau montrant certaines valeurs de couleur :

Rouge	Vert	Bleu	Couleur
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 peux trouver un joli **sélecteur de couleurs à utiliser sur w3schools** [(https://www.w3schools.com/colors/colors_rgb.asp)](https://www.w3schools.com/colors/colors_rgb.asp).

La matrice LED est une grille 8 x 8. Chaque LED de la grille peut être réglée sur une couleur différente. Voici une liste de variables pour 24 couleurs différentes. Chaque couleur comporte une valeur de rouge, vert et bleu :

Liste des variables de couleur



main.py

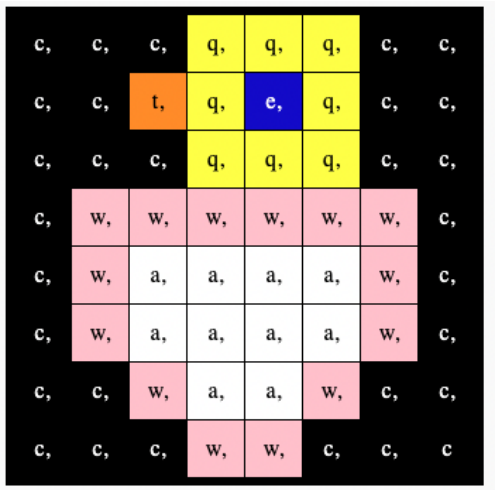
```
# Palette de couleurs
a = (255, 255, 255) # Blanc
b = (105, 105, 105) # Gris foncé
c = (0, 0, 0) # Noir
d = (100, 149, 237) # Bleu bluet
e = (0, 0, 205) # Bleu moyen
f = (25, 25, 112) # Bleu nuit
g = (0, 191, 255) # Bleu ciel profond
h = (0, 255, 255) # Cyan
j = (143, 188, 143) # Vert mer foncé
k = (46, 139, 87) # Vert mer
l = (0, 255, 127) # Vert printemps
m = (34, 139, 34) # Vert forêt
n = (154, 205, 50) # Vert-jaune
o = (128, 128, 0) # Olive
p = (240, 230, 140) # Kaki
q = (255, 255, 0) # Jaune
r = ( 184, 134, 11) # DarkGoldenrod
s = (139, 69, 19) # Brun amande
t = (255, 140, 0) # Orange foncé
u = (178, 34, 34) # Brique de feu
v = (255, 0, 0) # Rouge
w = (255, 192, 203) # Rose
y = (255, 20, 147) # Rose foncé
z = (153, 50, 204) # Orchidée foncée
```

Choisis une image

Choisir : Choisis 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 la liste.

Tu devras **copier** tout le code de l'image que tu as choisie puis le **coller** dans ton projet sous la ligne indiquant `# Ajouter des variables de couleur et une image`.

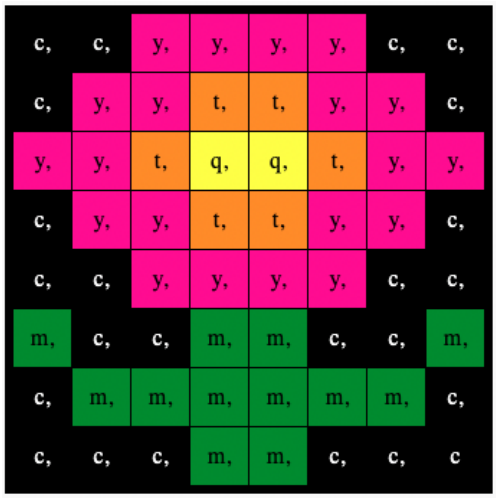
Poulet



main.py

```
a = (255, 255, 255) # Blanc
c = (0, 0, 0) # Noir
e = (0, 0, 205) # Bleu moyen
q = (255, 255, 0) # Jaune
t = (255, 140, 0) # Orange foncé
w = (255, 192, 203) # Rose

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]
```

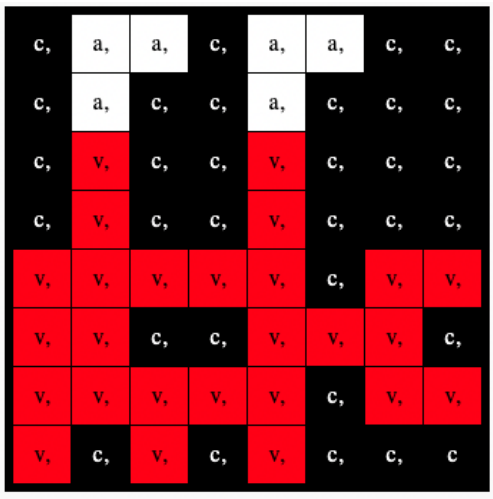
Fleur



main.py

```
c = (0, 0, 0) # Noir
m = (34, 139, 34) # Vert forêt
q = (255, 255, 0) # Jaune
t = (255, 140, 0) # Orange foncé
y = (255, 20, 147) # Rose foncé

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]
```

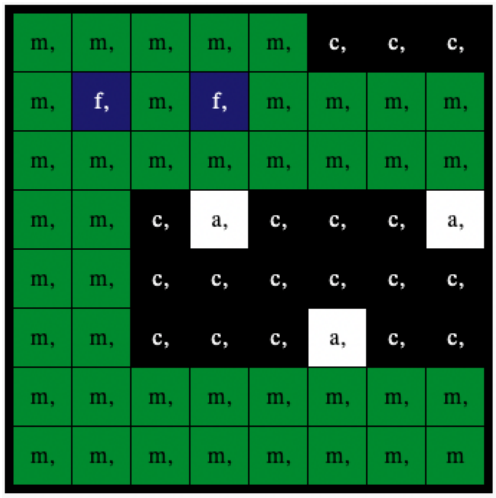
Crabe



main.py

```
a = (255, 255, 255) # Blanc
c = (0, 0, 0) # Noir
v = (255, 0, 0) # Rouge

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]
```

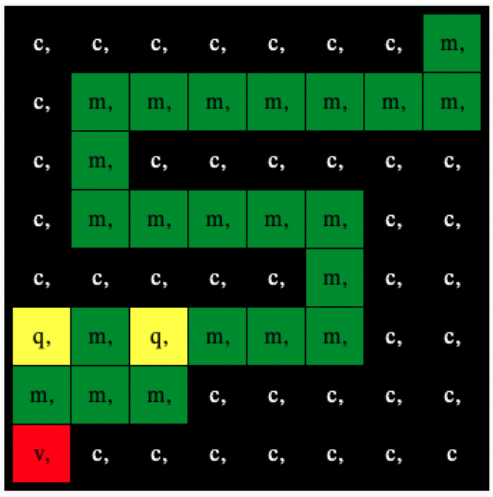
Crocodile



main.py

```
a = (255, 255, 255) # Blanc
c = (0, 0, 0) # Noir
f = (25, 25, 112) # MidnightBlue
m = (34, 139, 34) # Vert forêt

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]
```

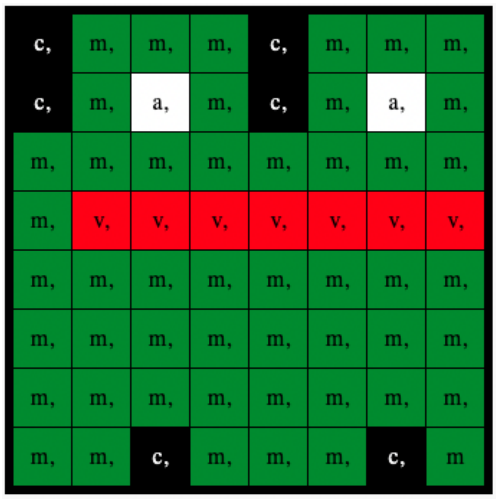
Serpent



main.py

```
c = (0, 0, 0) # Noir
 m = (34, 139, 34) # Vert forêt
 q = (255, 255, 0) # Jaune
 v = (255, 0, 0) # Rouge

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) # Noir
m = (34, 139, 34) # Vert forêt
q = (255, 255, 0) # Jaune
v = (255, 0, 0) # Rouge

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]
```

Recherche : la ligne indiquant `# Afficher l'image` et ajoute une ligne de code pour afficher ton image sur la matrice 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]

# Afficher l'image
sense.set_pixels(image)
```

Appuie sur **Run** en bas de l'éditeur, pour voir ton image s'afficher sur la matrice LED.

Déboguer

Mon code a une erreur de syntaxe :

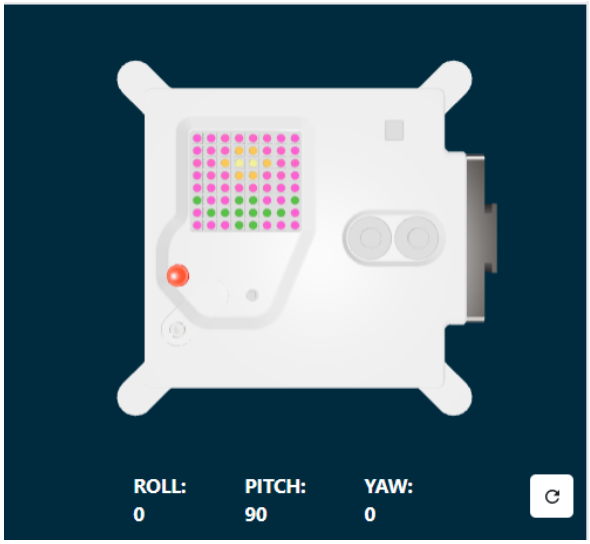
- Vérifie que ton code correspond au code des exemples ci-dessus
- Vérifie que tu as bien indenté le code dans ta liste
- Vérifie que ta liste est entourée de `[` et `]`
- Vérifie que chaque variable de couleur de la liste est séparée par une virgule

Mon image n'apparaît pas :

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

Étape 4 Détecter une couleur

Dans cette étape, tu vas configurer le capteur de luminosité des couleurs et l'utiliser pour détecter la quantité de rouge, de vert et de bleu qui atteint le capteur. Cette couleur sera ensuite utilisée pour coloriser ton image. Un astronaute s'approchant du capteur en chemise bleue verrait une image différente de celle d'un astronaute en chemise rouge.



Quelle que soit l'image choisie, l'arrière-plan utilise la variable `c` qui est définie sur le noir.

Utilise le capteur de couleurs pour coloriser ton arrière-plan.

Ajoute du code avant ta liste d'images pour obtenir la couleur du capteur et modifie ta variable de couleur d'arrière-plan `c` pour utiliser la couleur détectée par le capteur de couleur Sense HAT au lieu du noir.

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

main.py

```
# Ajouter des variables de couleur et une image

c = (0, 0, 0) # Noir
m = (34, 139, 34) # Vert forêt
q = (255, 255, 0) # Jaune
t = (255, 140, 0) # Orange foncé
y = (255, 20, 147) # Rose foncé

rvb = sense.color # obtenir la couleur du capteur
c = (rvb.red, rvb.green, rvb.blue) # utiliser la couleur détectée

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 de couleur sur une couleur de ton choix, puis **exécute** ton code. Ta couleur de fond va changer. Répète ce test avec une nouvelle couleur.

Astuce : tu devras cliquer sur « Exécuter » chaque fois que tu changeras la couleur.

Fais tourner ton programme en boucle

Le programme Mission Zero de l'Astro Pi peut être exécuté pendant 30 secondes maximum. Tu utiliseras ce temps pour vérifier à plusieurs reprises le capteur de couleurs et actualiser l'image.

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

- détecter la dernière couleur
- mettre à jour la couleur d'arrière-plan de l'image
- faire une pause d'une seconde

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

Ajoute du code juste au-dessus pour configurer ta boucle `for` pour `28` répétitions.

main.py

```
for i in range(28):
rvb = sense.color # obtenir la couleur du capteur
c = (rvb.red, rvb.green, rvb.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` pour qu'il se trouve **à l'intérieur** de la boucle `for`.

Astuce : pour indenter plusieurs lignes, mets en surbrillance les lignes que tu souhaites indenter puis 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):
    rvb = sense.color # obtenir la couleur du capteur
    c = (rvb.red, rvb.green, rvb.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]

    # Afficher l'image

    sense.set_pixels(image)
```

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

main.py

```
    # Afficher l'image

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

Astuce : assure-toi que cette ligne de code est indentée dans ta boucle `for`.

Test : Exécute ton code et modifie le sélecteur de couleur plusieurs fois lorsque ton projet est en cours d'exécution. Vérifie que ton image s'actualise pour utiliser la couleur détectée lors de sa prochaine exécution.

L'image cessera de s'actualiser lorsque la boucle se terminera afin que le programme ne dure pas plus de 30 secondes.

Déboguer

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

- Vérifie que ton code correspond au code des exemples ci-dessus
- Vérifie que tu as indenté le code dans ta boucle `for`
- Vérifie que ta liste est entourée de `[` et `]`
- Vérifie que chaque variable de couleur de la liste est séparée par une virgule

Mon code s'exécute pendant plus de 30 secondes :

- Diminue le nombre de fois où ta boucle « for » s'exécute, de 28 à 25 ou même 20.
- Diminue la durée du « sleep », de 1 seconde 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 ton animation a fini de fonctionner.

Astuce : assure-toi de **ne pas** indenter la ligne de code `sense.clear()` car tu veux que cela ne s'exécute qu'une fois à la fin de ton animation.

main.py

```
# Afficher l'image

  sense.set_pixels(image)
  sleep(1)

sense.clear()
```

Test : Exécute à nouveau ton code. Lorsque ton projet est terminé, la matrice LED est effacée, ce qui fait que toutes les lumières deviennent noires (éteintes).

Déboguer

La matrice LED devient noire toutes les secondes :

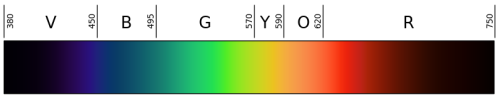
- Vérifie que tu n'as pas indenté le code `sense.clear()` dans ta boucle `for`

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

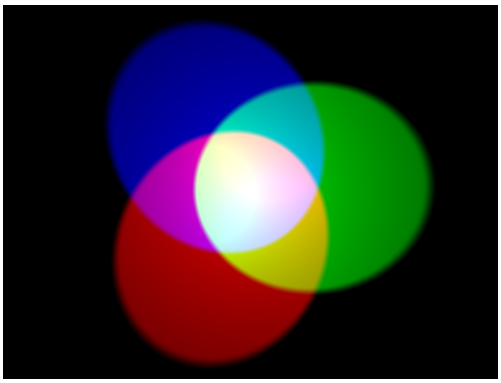
Tu peux mélanger ta propre couleur ou utiliser les valeurs de la liste de couleurs pour créer ta nouvelle couleur `x`.

Représenter des couleurs avec des nombres

La couleur d'un objet dépend de la couleur de la lumière qu'il réfléchit ou émet. La lumière peut avoir différentes longueurs d'onde et la couleur de la lumière dépend de sa longueur d'onde. La couleur donnée par la lumière en fonction de sa longueur d'onde peut être vue dans le diagramme ci-dessous. Tu pourrais reconnaître cela comme les couleurs de l'arc-en-ciel.



Les humains voient les couleurs grâce à des cellules spéciales dans leurs yeux. Ces cellules sont appelées *cônes*. Nous avons trois types de cellules de cône et chaque type détecte soit une lumière rouge, bleue ou verte. Par conséquent, toutes les couleurs que nous voyons ne sont que des mélanges de couleurs rouge, bleu et vert.



Dans le mélange de couleurs additif, trois couleurs (rouge, vert et bleu) sont utilisées pour créer d'autres couleurs. Dans l'image ci-dessus, il y a trois spots de même luminosité, un pour chaque couleur. En l'absence de toute couleur, le résultat est noir. Si les trois couleurs sont mélangées, le résultat est blanc. Lorsque rouge et vert se combinent, le résultat est jaune. Lorsque rouge et bleu se combinent, le résultat est magenta. Lorsque bleu et vert se combinent, le résultat est cyan. Il est possible d'obtenir encore plus de couleurs en variant la luminosité des trois couleurs originales utilisées.

Les ordinateurs enregistrent tout sous forme de uns et de zéros. Ces uns et ces zéros sont souvent combinés en ensembles de 8, appelés **octets**.

Un seul octet peut représenter n'importe quel nombre de 0 à 255.

Lorsque nous voulons représenter une couleur dans un programme informatique, nous pouvons le faire en définissant les quantités de rouge, de bleu et de vert qui composent cette couleur. Ces quantités sont généralement stockées sous la forme d'un seul octet et donc d'un nombre compris entre 0 et 255.

Voici un tableau montrant certaines valeurs de couleur:

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

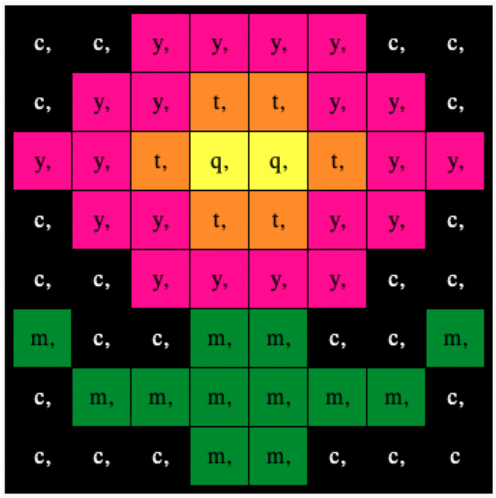
Tu peux trouver un bon **sélecteur de couleurs sur W3Schools** (https://www.w3schools.com/colors/colors_rgb.asp).

main.py

```
    # Afficher l'image

      sense.set_pixels(image)
      sleep(1)

    x = (178, 34, 34)  # choisis tes propres valeurs de rouge, vert et bleu entre 0 et 255 sense.clear(x)
```

Test : Exécute à nouveau ton code. Lorsque ton projet est terminé, la matrice LED s'efface et utilise la couleur que tu as choisie. Tu peux changer puis tester la couleur autant de fois que tu le souhaites.

Exemple de code terminé



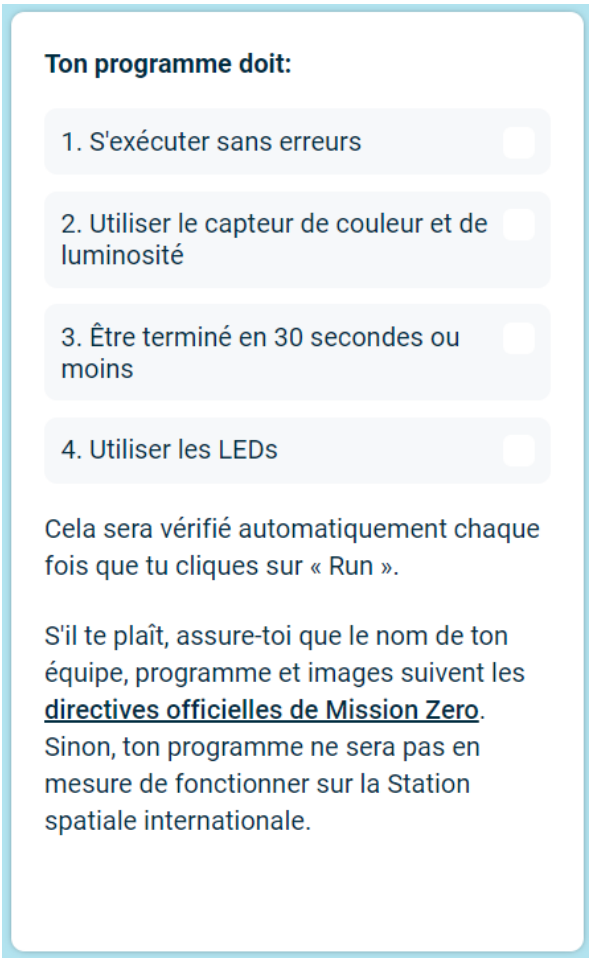
main.py

```
# Importer les bibliothèques
from sense_hat import SenseHat
from time import sleep

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

# Configurer le capteur de couleurs
sense.color.gain = 60 # Régler la sensibilité du capteur
sense.color.integration_cycles = 64 # L'intervalle auquel la mesure est effectuée

# Ajouter des variables de couleur et une image

c = (0, 0, 0) # Noir
m = (34, 139, 34) # Vert forêt
q = (255, 255, 0) # Jaune
t = (255, 140, 0) # Orange foncé
y = (255, 20, 147) # Rose foncé

for i in range(28):
  rvb = sense.color # obtenir la couleur du capteur
  c = (rvb.red, rvb.green, rvb.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]

  # Afficher l'image

  sense.set_pixels(image)
  sleep(1)

x = (178, 34, 34)  # choisis tes propres valeurs de rouge, vert et bleu entre 0 et 255
sense.clear(x)
```

Étape 5 Soumettre ton programme

Tu pouvez maintenant entrer le défi **Astro Pi Mission Zero** (https://astro-pi.org/fr/mission-zero) en utilisant le code que tu avez écrit.

Ton code doit respecter quelques règles pour que tu puisses le soumettre et l'exécuter dans la Station Spatiale Internationale. Si ton code les respecte, les règles situées en bas de l'**émulateur Sense HAT** s'allumeront en vert lorsque tu exécuteras le programme.



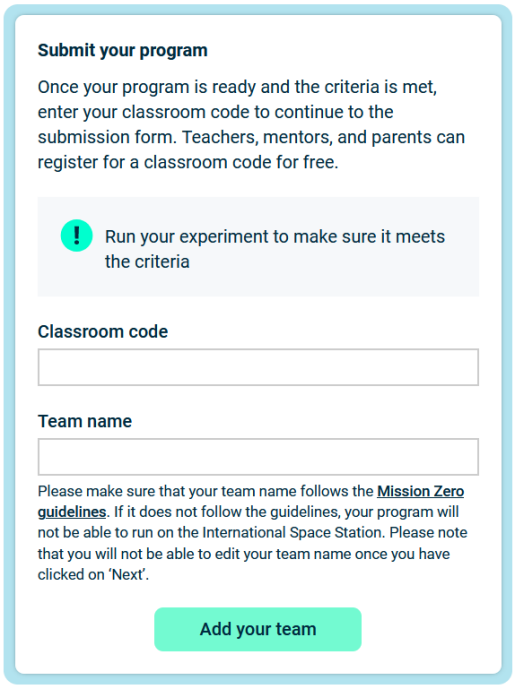
Astuce : teste ton code avec quelques paramètres de couleur différents (en utilisant le sélecteur) pour t'assurer qu'il fonctionne toujours correctement.

Assure-toi que ton programme respecte les **directives officielles** (https://astro-pi.org/fr/mission-zero/guidelines) de Mission Zero. Sinon, ton programme ne sera pas en mesure de fonctionner sur la Station spatiale internationale.

N'inclus aucun des éléments suivants dans le nom ou le code de ton équipe :

- Tout ce qui pourrait être interprété comme étant de nature illégale, politique ou sensible
- Des drapeaux, car ils peuvent être considérés comme politiquement sensibles
- Des choses qui sont désagréables ou qui pourraient heurter une autre personne
- Des données personnelles telles que des numéros de téléphone, des liens vers les réseaux sociaux, et des adresses e-mail
- Des images obscènes
- Des caractères spéciaux ou des émojis
- Des gros mots ou des injures

Saisis ton code de classe et le nom de ton équipe dans la case en bas : ton mentor te dira quel est ton code.



Les notes destinées aux mentors se trouvent dans l'étape **Introduction** (https://projects.raspberrypi.org/fr-FR/projects/astro-pi-mission-zero/0).

Appuie sur le bouton **Ajouter ton équipe** pour saisir ton code. Sache qu'un programme ne peut pas être modifié après avoir été soumis.

Ton mentor recevra un e-mail pour confirmer ton inscription.

Si tu veux, tu peux partager le lien vers ton code sur les réseaux sociaux pour informer les gens que le code que tu as écrit sera exécuté dans l'espace !

Étape 6 Et ensuite — d'autres projets Astro Pi

Maintenant que tu as terminé ta mission, pourquoi ne pas essayer d'autres projets en utilisant les autres capteurs de l'Astro Pi ?

Si tu te sens d'attaque, tu peux participer à **Mission Space Lab** (https://astro-pi.org/missions/space-lab/) ! Forme une équipe de deux à six personnes et travaille ensemble comme de vrais scientifiques de l'espace pour concevoir ta propre expérience. Les meilleures idées soumises recevront un kit Astro Pi à utiliser pour t'aider dans ta mission.

Tu peux aussi essayer l'un de nos autres projets Sense HAT :

- Apprendre **plus sur le Sense HAT** (https://projects.raspberrypi.org/fr-FR/projects/getting-started-with-the-sense-hat) et les autres choses qu'il peut faire
- Créer de beaux **scintillements aléatoires** (https://projects.raspberrypi.org/fr-FR/projects/sense-hat-random-sparkles) sur l'écran LED du Sense HAT
- Créer un jeu d'**Astronaute Flappy** (https://projects.raspberrypi.org/fr-FR/projects/flappy-astronaut)
- Défier tes amis avec un jeu de **labyrinthe de billes** (https://projects.raspberrypi.org/fr-FR/projects/sense-hat-marble-maze)
- Recréer le jeu classique **Pong** (https://projects.raspberrypi.org/fr-FR/projects/sense-hat-pong)

Ce projet a été traduit par des bénévoles:

Jonathan Vannieuwkerke
Michel Arnols

Grâce aux bénévoles, nous pouvons donner aux gens du monde entier la chance d'apprendre dans leur propre langue. Vous pouvez nous aider à atteindre plus de personnes en vous portant volontaire pour la traduction - plus d'informations sur **rpf.io/translate** (https://rpf.io/translate).