

Guide du créateur Astro Pi Mission Space Lab

Le Guide du créateur Mission Space Lab est conçu pour aider les équipes à écrire des programmes permettant de calculer la

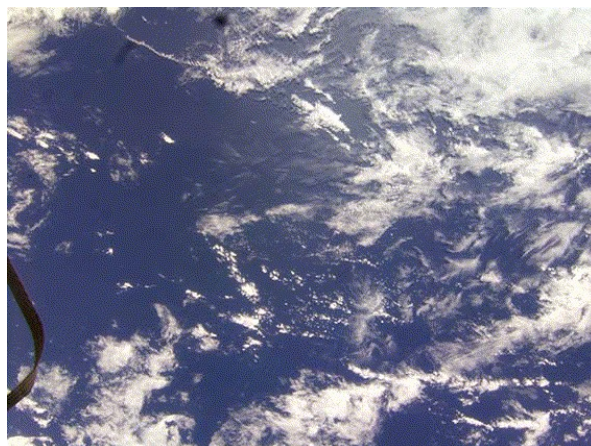
vitesse de l'ISS. Il présente la tâche et les exigences relatives aux programmes, indique d'autres guides pour le projet, et donne des conseils sur les éléments à soumettre et la manière de procéder.



Étape 1 Introduction

Ce guide est conçu pour vous aider à créer votre programme avec votre équipe pour Mission Space Lab 2023/24. Pour cette mission, votre tâche consiste à créer un programme qui recueillera des données au moyen des capteurs et de la caméra de l'un des ordinateurs Astro Pi et qui calculera, à partir de ces données, la vitesse à laquelle la Station spatiale internationale (ISS) se déplace. Nous vous fournirons de nombreux documents de référence pour vous aider à écrire et à développer votre programme, notamment un exemple de projet à partir de photos historiques. Ensuite, nous vous aiderons à adapter et à tester votre programme afin qu'il puisse fonctionner pendant 10 minutes à bord de l'ISS et produire une valeur en temps réel pour la vitesse de l'ISS.

Il ne s'agit pas d'un guide étape par étape complet expliquant comment créer un programme qui résoudra le problème posé dans cette mission. Votre équipe et vous-même devrez trouver des idées et des solutions et déterminer comment les mettre en œuvre.



Vous ne connaissez pas Mission Space Lab ? Ne vous inquiétez pas ! Parcourez le **site Astro Pi** (<https://astro-pi.org/mission-space-lab/>) pour obtenir plus d'informations.

Les réponses aux nombreuses questions qui vont se poser se trouveront probablement souvent en ligne, et nous vous invitons à faire des recherches et à essayer différentes solutions si vous êtes bloqués. Nous allons également programmer plusieurs webinaires où vous pourrez poser vos questions à l'équipe d'Astro Pi Mission Control. Vous pouvez également nous envoyer un message à enquiries@astro-pi.org (<mailto:enquiries@astro-pi.org>). N'hésitez pas à nous contacter. Nous serions ravis d'avoir de vos nouvelles.

De nombreuses ressources sont disponibles pour vous aider à réussir chaque étape de votre parcours Astro Pi.

Si vous êtes bloqués, n'hésitez pas à **nous contacter** (<mailto:enquiries@astro-pi.org>) et nous ferons de notre mieux pour vous aider !



Ce que vous aurez à faire

Votre tâche consiste à concevoir un programme qui fonctionnera pendant 10 minutes à bord de l'ISS. Pendant ce temps, il recueillera des données et les utilisera pour estimer la vitesse de l'ISS. À la fin des 10 minutes, votre programme doit avoir rédigé un fichier contenant votre estimation de la vitesse de l'ISS en kilomètres par seconde.

Vous pouvez utiliser notre plug-in Astro Pi Replay pour simuler l'exécution de votre code en direct sur les Astro Pi de l'ISS et modifier le programme pour qu'il fonctionne en temps réel.

Nous vous donnerons également des informations sur la façon d'améliorer votre programme afin de vous assurer qu'il fonctionne correctement sur l'ISS tout en respectant les règles de sécurité.

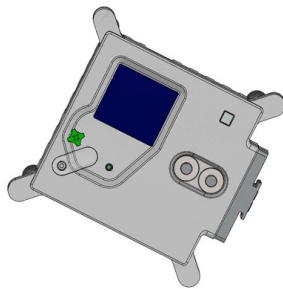
Ce dont vous aurez besoin

Pour ce projet, vous aurez besoin des éléments suivants :

- **Un ordinateur équipé de Python 3.9.2 ou d'une version ultérieure.** Vous pouvez utiliser n'importe quel ordinateur Microsoft Windows, macOS ou Linux. **Vous trouverez les instructions d'installation de Python ici** (<https://projects.raspberrypi.org/en/projects/generic-python-install-python3>). Les exigences complètes concernant Python pour Mission Space Lab sont décrites plus loin dans ce guide.
- **L'environnement de développement intégré (IDE) Thonny.** Nous vous recommandons d'utiliser Thonny, car il est facile à utiliser et il est disponible pour les systèmes d'exploitation Microsoft Windows, macOS, Linux et Raspberry Pi. Vous trouverez les instructions d'installation de Thonny **ici** (<https://thonny.org/>).
- **Une connexion à Internet.** Vous aurez besoin d'accéder à Internet pour installer les paquets Python, pour utiliser le plug-in Astro Pi Replay et pour soumettre votre programme.

Étape 2 Les ordinateurs Astro Pi

Les Astro Pi embarqués à bord de l'ISS sont deux ordinateurs Raspberry Pi 4 8 Go modifiés, équipés d'une carte d'extension Sense HAT et d'une caméra. Ils sont placés dans une malle renforcée sur mesure en aluminium. Le Sense HAT (V2) comprend notamment des capteurs de température et d'humidité, un gyroscope, un magnétomètre, un accéléromètre et des capteurs de lumière/couleur, qui permettent de mesurer le champ magnétique local et l'accélération, par exemple. Les ordinateurs sont équipés d'une puissante caméra haute qualité Raspberry Pi dotée d'un objectif de 5 mm capable de prendre des photos incroyables de la Terre. De plus, ces ordinateurs peuvent effectuer un apprentissage automatique en temps réel grâce aux accélérateurs Coral associés. Vous pouvez en **découvrir plus sur les ordinateurs et les capteurs ici** (<https://astro-pi.org/about/the-computers>).



Sachant ce que les capteurs disponibles sur les Astro Pi peuvent faire, trouvez une manière créative de les utiliser pour déterminer la vitesse de l'ISS. Ne vous inquiétez pas si tout n'est pas parfait au début. Essayez de réfléchir à différentes approches, même si elles semblent inhabituelles. Seul ou en équipe, combien de façons de calculer la vitesse à l'aide de ces outils pouvez-vous trouver ?

Proposez plusieurs manières différentes de calculer la vitesse de l'ISS à l'aide du matériel Astro Pi. Faites preuve de créativité et essayez de sortir des sentiers battus. Une fois que vous avez plusieurs options, discutez-en en équipe et choisissez celle qui, selon vous, donnera le résultat le plus juste.



Dans la section suivante, vous découvrirez les différentes bibliothèques Python disponibles pour vous aider dans votre projet, ainsi que celles que vous ne pouvez pas utiliser pour des raisons de sécurité.

L'environnement Python d'Astro Pi

Les ordinateurs Astro Pi de l'ISS disposent de la version 3.9.2 de Python, donc vous devrez utiliser cette version ou une version ultérieure. Si vous utilisez une version ultérieure, sachez qu'il peut y avoir de nouvelles fonctions qui fonctionnent sur votre ordinateur, mais pas sur les Astro Pi.

Les modules (parties) de la bibliothèque standard que vous pouvez utiliser sont soumis à certaines restrictions. Les modules suivants ne sont pas autorisés et, si vous les utilisez, votre programme ne sera pas accepté :

Bibliothèques interdites

(<https://docs.google.com/spreadsheets/u/0/d/1EoVzgA8gOiDXsJ1k9dQBdPyFC8U3bXFca2dRmdKNbcl/edit>)

En plus de l'environnement standard Python, les Astro Pi disposent de bibliothèques supplémentaires pour vous aider à accomplir la mission. Chacune d'elles est brièvement décrite ci-dessous avec des exemples. Des liens sont également fournis pour plus de détails si vous en avez besoin. N'oubliez pas de marquer cette page pour plus tard !



Skyfield

Utilisation

Skyfield est un programme d'astronomie qui calcule la position des étoiles, des planètes et des satellites qui sont en orbite autour de la Terre.

La rubrique **Comment localiser l'ISS** (6) explique comment utiliser Skyfield pour obtenir la position de la Station spatiale internationale au-dessus de la Terre et pour déterminer si l'ISS est éclairée par le Soleil.

Documentation

- rhodesmill.org/skyfield (<https://rhodesmill.org/skyfield/>)



picamera

La bibliothèque Python qui sert à contrôler le module caméra de Raspberry Pi s'appelle picamera. Pour commencer, consultez **ce guide du projet** (<https://projects.raspberrypi.org/fr-FR/projects/getting-started-with-picamera/4>) pour découvrir comment l'utiliser.

Utilisation

```
from picamera import PiCamera
from time import sleep

camera = PiCamera()
camera.resolution = (2592, 1944)

for i in range(3*60):
    camera.capture(f'image_{i:03d}.jpg') # Take a picture every minute for 3 hours
    sleep(60)
```

Documentation

- picamera.readthedocs.io (<https://picamera.readthedocs.io>)



colorzero

`colorzero` est une bibliothèque pour la manipulation des couleurs, conçue pour être simple à utiliser.

Utilisation

`colorzero` facilite la transition entre deux couleurs :

```
from colorzero import Color
from sense_hat import SenseHat
from time import sleep

sense = SenseHat()

start = Color('yellow')
end = Color('cyan')

# Slowly transition the Sense HAT from the `start` to the `end` color
for color in start.gradient(end, steps=100):
    sense.clear(color.rgb_bytes)
    sleep(0.1)
```

Documentation

- colorzero.readthedocs.io (<https://colorzero.readthedocs.io>)



GPIO Zero

GPIO Zero est une bibliothèque GPIO (General-Purpose Input/Output, c'est-à-dire Entrée-sortie à usage général) simple mais puissante. La plupart de ses fonctions sont limitées à bord de l'ISS. Par exemple, la seule broche à laquelle vous pouvez accéder est la broche GPIO 12, où le capteur de mouvement est connecté. Cependant, certaines de ses autres fonctionnalités peuvent être utiles pour votre expérience, comme le dispositif interne CPUtemperature.

Utilisation

Comparez les relevés de température du processeur du Raspberry Pi aux relevés de température du Sense HAT :

```
from sense_hat import SenseHat from
gpiozero import CPUtemperature

sense = SenseHat()
cpu = CPUtemperature()

while True:
    print(f'CPU: {cpu.temperature}')
    print(f'Sense HAT: {sense.temperature}')
```

Documentation

- gpiozero.readthedocs.io (<https://gpiozero.readthedocs.io>)



NumPy

numpy est une bibliothèque de traitement de tableau à usage général conçu pour manipuler efficacement de grands tableaux multidimensionnels (par exemple, des matrices) d'enregistrements arbitraires sans sacrifier trop de vitesse pour de petits tableaux multidimensionnels.

Utilisation

numpy est particulièrement utile pour capturer les données de caméra que vous souhaitez manipuler :

```
from picamera import PiCamera
from time import sleep import
numpy as np

camera = PiCamera()

camera.resolution = (320, 240)
camera.framerate = 24
output = np.empty((240, 320, 3), dtype=np.uint8)
sleep(2)
camera.capture(output, 'rgb')
```

Documentation

- docs.scipy.org/doc (<https://docs.scipy.org/doc/>)

SciPy

SciPy est une librairie Python libre et open source utilisée pour l'informatique scientifique et technique. SciPy contient des modules pour l'optimisation, l'algèbre linéaire, l'intégration, l'interpolation, les fonctions spéciales, la FFT (transformation de Fourier rapide), le traitement de signaux et d'images, les solveurs d'EDO (équations différentielles ordinaires) et d'autres tâches fréquentes en science et en ingénierie. Vous devrez peut-être utiliser cette bibliothèque pour résoudre une équation particulière.

Documentation

- docs.scipy.org/doc (<https://docs.scipy.org/doc/>)

TensorFlow Lite et PyCoral

TensorFlow Lite et la bibliothèque PyCoral permettent d'utiliser ou de ré-entraîner des modèles d'apprentissage automatique (ML) existants en cas d'inférence. Cette bibliothèque est construite sur TensorFlow Lite mais dispose d'une interface plus simple et de plus haut niveau, et permet d'utiliser facilement l'accélérateur Coral ML (Edge TPU). Notez que TensorFlow (contrairement à TensorFlow Lite) n'est pas pris en charge par le système d'exploitation du kit car TensorFlow nécessite un système d'exploitation 64 bits. Vous pouvez utiliser ces bibliothèques pour créer des classificateurs d'objets, par exemple. Pour plus d'informations, voir la section **Apprentissage automatique et vision numérique** (7).

Documentation

- [TensorFlow Lite](https://www.tensorflow.org/lite/api_docs/python/tf/lite) (https://www.tensorflow.org/lite/api_docs/python/tf/lite) **PyCoral**
- (<https://coral.ai/docs/edgetpu/tflite-python/>)

pandas

`pandas` est une bibliothèque open source offrant des structures de données et des outils d'analyse de données très performants et simples à utiliser. Vous pouvez l'utiliser lorsque vous analysez les résultats des essais de votre programme.

Utilisation

```
import pandas as pd

df = pd.read_csv("my_test_data.csv")
df.describe()
```

Documentation

- pandas.pydata.org (<https://pandas.pydata.org/>)

logzero

Logzero est une bibliothèque utilisée pour faciliter la journalisation. Les journaux sont les enregistrements de ce qui s'est passé pendant qu'un programme était en cours d'exécution, et peuvent être vraiment utiles pour le débogage.

Utilisation

Les journaux sont classés en différents niveaux en fonction de leur gravité. En utilisant les différents niveaux de manière appropriée, vous pourrez ajuster le nombre d'informations que vous obtenez sur votre programme en fonction de vos besoins de débogage.

```
from logzero import logger

logger.debug("hello")
logger.info("info")
logger.warning("warning")
logger.error("error")
```

Documentation

- [logzero.readthedocs.io](https://logzero.readthedocs.io/en/latest/) (<https://logzero.readthedocs.io/en/latest/>)



Matplotlib

`matplotlib` est une bibliothèque de traçage 2D qui produit des tracés de qualité publication dans une variété de formats papier et d'environnements interactifs. Vous pouvez l'utiliser pour analyser les résultats de vos essais.

Utilisation

```
from sense_hat import SenseHat from
gpiozero import CPUtemperature
import matplotlib.pyplot as plt from
time import sleep

sense = SenseHat()
cpu = CPUtemperature()

st, ct = [], []
for i in range(100):
    st.append(sense.temperature)
    ct.append(cpu.temperature)
    sleep(1)

plt.plot(st)
plt.plot(ct)
plt.legend(['Sense HAT temperature sensor', 'Raspberry Pi CPU temperature'], loc='upper left')
plt.show()
```



Le résultat du programme est une courbe de température générée avec `matplotlib`.

Documentation

- matplotlib.org (<https://matplotlib.org/>)



Pillow

`Pillow` est une bibliothèque de traitement d'images. Elle offre une prise en charge étendue des formats de fichier, une représentation interne efficace et des capacités de traitement d'image assez puissantes.

La bibliothèque d'images de base est conçue pour offrir un accès rapide aux données stockées dans quelques formats de pixel de base. Elle peut fournir une base solide pour un outil de traitement d'image général.

Documentation

- pillow.readthedocs.io (<https://pillow.readthedocs.io/>)

OpenCV

`opencv` est une bibliothèque de vision numérique open source. Plus particulièrement, c'est le package `opencv-contrib-python-headless` qui est installé sur les Astro Pi. Il inclut `opencv` en intégralité, ainsi que des modules supplémentaires (listés dans la **documentation OpenCV** (<https://docs.opencv.org/master/>)), et exclut toutes les fonctionnalités de l'interface utilisateur graphique. Vous pouvez utiliser OpenCV pour la **détection de contours** (<https://projects.raspberrypi.org/en/projects/astro-pi-iss-speed/3>), par exemple.

Documentation

- [docs.opencv.org \(https://docs.opencv.org/4.4.0/\)](https://docs.opencv.org/4.4.0/)

exif

`exif` vous permet de lire et de modifier les métadonnées Exif des images à l'aide de Python. Vous pouvez l'utiliser pour intégrer les données GPS **sur toutes les images que vous prenez, ou pour analyser les photos prises à bord de l'ISS** (<https://projects.raspberrypi.org/en/projects/astro-pi-iss-speed/1>).

Documentation

- [pypi.org/project/exif \(https://pypi.org/project/exif/\)](https://pypi.org/project/exif)

scikit-learn

`scikit-learn` est un ensemble d'outils simples et efficaces pour l'extraction et l'analyse des données accessibles à tous et réutilisables dans différents contextes. Il a été conçu pour interagir avec `numpy`, `scipy` et `matplotlib`.

Documentation

- [scikit-learn.org \(https://scikit-learn.org\)](https://scikit-learn.org)

scikit-image

`scikit-image` est une bibliothèque de traitement d'images open source. Elle comprend des algorithmes pour la segmentation, les transformations géométriques, la manipulation de l'espace colorimétrique, l'analyse, le filtrage, la morphologie, la détection de caractéristiques, et bien plus encore.

Documentation

- [scikit-image.org \(https://scikit-image.org/\)](https://scikit-image.org/)

reverse-geocoder

`reverse-geocoder` prend une coordonnée latitude/longitude et donne le village ou la ville la plus proche.

Utilisation

Lorsqu'il est utilisé avec `skyfield`, `reverse-geocoder` peut déterminer la position actuelle de l'ISS :


```
import reverse_geocoder
from orbit import ISS

coordinates = ISS.coordinates()
coordinate_pair = (
    coordinates.latitude.degrees,
    coordinates.longitude.degrees)
location = reverse_geocoder.search(coordinate_pair)
print(location)
```

Selon ce résultat, l'ISS se trouve actuellement au-dessus de Hamilton, dans l'état de New York :

```
[OrderedDict([
  ('lat', '42.82701'),
  ('lon', '-75.54462'),
  ('name', 'Hamilton'),
  ('admin1', 'New York'),
  ('admin2', 'Madison County'),
  ('cc', 'US')
]])]
```

Documentation

- github.com/thampiman/reverse-geocoder (<https://github.com/thampiman/reverse-geocoder>)



sense_hat

La bibliothèque `sense_hat` est la principale bibliothèque utilisée pour recueillir des données à l'aide du Sense HAT d'Astro Pi. **Consultez ce guide de projet** (<https://projects.raspberrypi.org/en/projects/getting-started-with-the-sense-hat>) pour commencer.

Utilisation

Vous pouvez consigner l'humidité par rapport à l'affichage en utilisant le code ci-dessous :

```
from sense_hat import SenseHat
sense = SenseHat()
sense.show_message(str(sense.get_humidity()))
```

Documentation

- <https://sense-hat.readthedocs.io/en/latest/> (<https://sense-hat.readthedocs.io/en/latest/>)
- **Documentation supplémentaire pour le capteur de couleur** (<https://gist.github.com/boukeas/e46ab3558b33d2f554192a9b4265b85f>)



pisense

`pisense` est une interface alternative au Sense HAT de Raspberry Pi. La principale différence avec `sense_hat` est que dans `pisense` les différents composants du Sense HAT (écran, joystick, capteurs environnementaux, etc.) sont représentés par des classes distinctes qui peuvent être utilisées individuellement ou par la classe principale qui les comprend toutes.

L'écran comporte quelques atouts supplémentaires, dont la prise en charge de toutes les polices de PIL, la représentation sous forme de tableau numpy (ce qui rend très simple le défilement en assignant des sections d'une image plus grande), et plusieurs fonctions d'animation rudimentaires. Le joystick et l'ensemble des capteurs disposent également d'une interface itérable.

Utilisation

```

from pisense import SenseHAT, array
from colorzero import Color

hat = SenseHAT(emulate=True)
hat.screen.clear()

B = Color('black')
r = Color('red') w
= Color('white') b
= Color('blue')

black_line = [B, B, B, B, B, B, B, B]
flag_line = [B, b, b, w, w, r, r, B]
flag = array(black_line * 2 + flag_line * 4 + black_line * 2)

hat.screen.fade_to(flag)

```

Documentation

- **pisense.readthedocs.io** (<https://pisense.readthedocs.io/en/latest/>)

Compte tenu des nombreuses restrictions de sécurité lors de l'exécution d'un programme à bord de l'ISS, ce sont les seules bibliothèques tierces que vous pourrez utiliser si votre programme est exécuté sur les Astro Pi. N'hésitez pas à **nous contacter** (enquiries@astro-pi.org) si vous pensez qu'il manque quelque chose ou si vous avez des suggestions.

Configurer votre environnement de programmation

Nous vous recommandons d'utiliser Thonny pour créer votre programme.

Installer Thonny

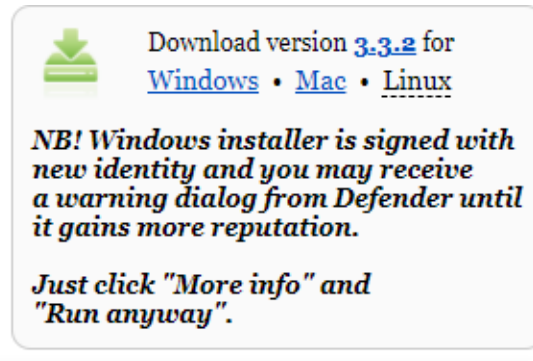
Installer Thonny sur un Raspberry Pi

- Thonny est déjà installé sur Raspberry Pi OS, mais peut avoir besoin d'être mis à jour pour avoir la dernière version.
- Ouvrez une fenêtre du terminal en cliquant sur l'icône située en haut à gauche de l'écran ou en appuyant simultanément sur les touches Ctrl+Alt+T.
- Dans la fenêtre, tapez la commande suivante pour mettre à jour votre système d'exploitation et Thonny :

```
sudo apt update && sudo apt upgrade -y
```

Installer Thonny sur d'autres systèmes d'exploitation

- Sous Windows, macOS et Linux, vous pouvez installer la dernière version de l'IDE Thonny ou mettre à jour la version existante. Dans un navigateur Web, accédez à **thonny.org** (<https://thonny.org/>)
- Dans le coin supérieur droit de la fenêtre du navigateur, vous verrez des liens de téléchargement pour Windows et macOS, et les instructions pour Linux.
- Téléchargez les fichiers appropriés et exécutez-les pour installer Thonny.



Ouvrir Thonny

Ouvrez Thonny depuis votre lanceur d'application. Vous devriez obtenir une fenêtre qui ressemble à ceci :



Vous pouvez utiliser Thonny pour écrire du code Python standard. Saisissez ce qui suit dans la fenêtre principale, puis cliquez sur le bouton **Run** (Exécuter) (il vous sera demandé d'enregistrer le fichier).

```
print('Hello World!')
```

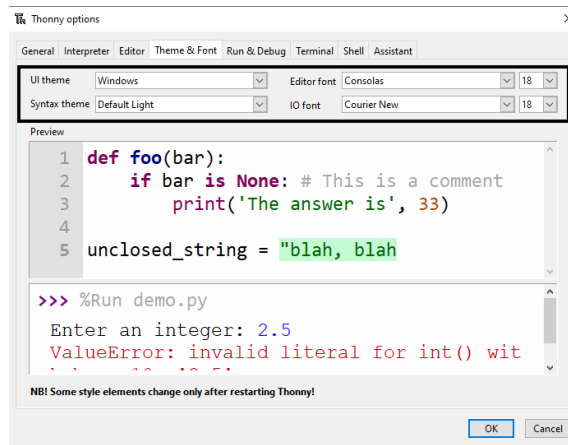


Modifier le thème et la police dans Thonny

Thonny vous permet de modifier le thème et la police du logiciel. Cette fonctionnalité permet d'augmenter la taille de la police et de modifier les couleurs de l'arrière-plan et du texte en fonction de vos besoins.

Pour modifier le thème et la police :

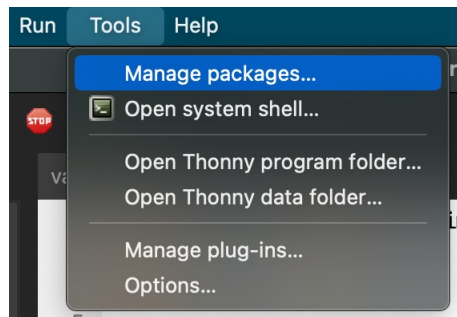
- Cliquez sur Tools (Outils) -> Options.
- Cliquez sur l'onglet Theme & Font (Thème et Police).
- Cliquez sur les cases déroulantes pour chaque option jusqu'à ce que vous trouviez les paramètres qui correspondent le mieux à vos besoins.



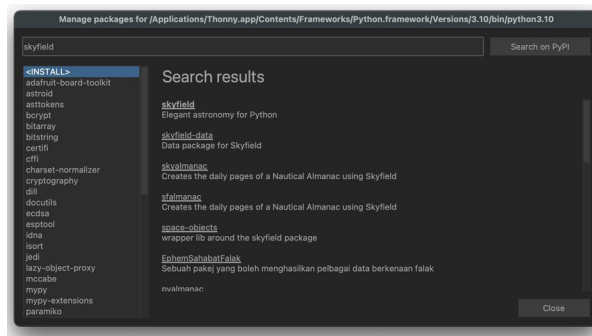
- Cliquez sur OK lorsque vous avez terminé.

Attention : Privilégiez les polices simples et claires. Si vous utilisez une police de type écriture manuelle, cela peut rendre la lecture et le débogage difficiles.

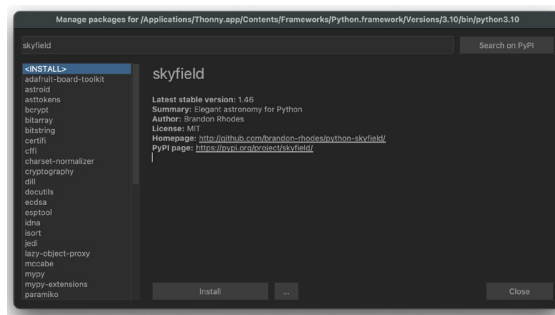
Pour installer l'une des bibliothèques Python, ouvrez Thonny et cliquez sur **Tools(Outils) > Manage packages (Gérer les paquets)**.



Recherchez la bibliothèque qui vous intéresse en saisissant son nom dans la barre de recherche.



Sélectionnez le fichier approprié dans les résultats de recherche, puis cliquez sur **Install (Installer)**.



Le plug-in Astro Pi Replay

Le plug-in Astro Pi Replay fonctionne comme un simulateur que vous pouvez utiliser sur Terre et qui fera en sorte que votre programme agisse comme s'il fonctionnait sur un Astro Pi à bord de l'ISS. Il vous permet de tester votre code avant de l'envoyer dans l'espace sans avoir besoin d'un Raspberry Pi, d'une caméra ou d'un Sense HAT. Toutefois, la simulation n'est pas parfaite et ne produira que des photos et des données de capteur issues de son propre ensemble de données, mais cela devrait tout de même vous permettre de vérifier que votre programme fonctionnerait à bord de l'ISS.

Vous trouverez des instructions pour télécharger et utiliser l'outil Astro Pi Replay plus loin dans ce guide du créateur.

Anticiper

C'est le moment de déterminer de quelle manière votre équipe va aborder cette mission. Discutez de la façon dont vous allez choisir votre méthode, divisez les tâches et planifiez votre programme. Parlez avec votre mentor de vos idées, de vos progrès et de tous les obstacles en cours de route. Il ou elle aura beaucoup d'idées pour vous aider dans la planification.

Étape 3 Écriture de votre programme et ressources pour vous aider

Cette section vous aidera à commencer l'écriture de votre programme. Elle contient des liens vers d'autres guides de projet qui vous donneront certaines des compétences de codage dont vous avez besoin. Vous pouvez choisir les guides de projet que vous souhaitez consulter en fonction des capteurs ou des caméras que vous allez utiliser dans votre programme. À ce stade, vous devez déjà avoir passé du temps avec votre équipe pour planifier votre programme avec votre mentor, et avoir décidé quelles données vous allez recueillir pour faire vos calculs.

Pour commencer

Nous vous recommandons de commencer à écrire votre programme par petites étapes et de ne pas essayer de tout faire à la fois.

Afin de rester organisés, créez un dossier pour stocker tous les fichiers de votre projet. Pour le nom du dossier, vous pouvez utiliser le nom de votre équipe.



Le fichier main.py

Chaque contribution doit inclure un fichier intitulé `main.py`. Il s'agit du fichier à partir duquel votre programme sera exécuté et qui sera testé par Astro Pi Mission Control. Lorsque vous exécuterez le programme complet, il devra faire tout ce dont vous avez besoin pour estimer la vitesse de l'ISS. Commencez par créer un fichier pour votre programme principal et ajoutez le code au fur et à mesure que vous parvenez à le faire fonctionner.

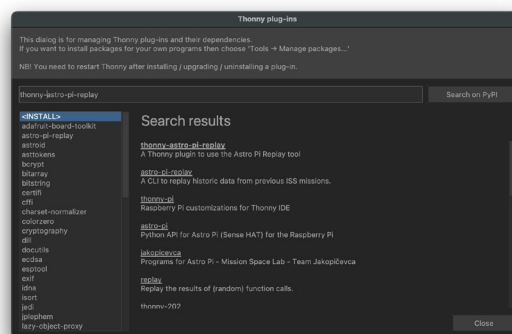
Créez un nouveau fichier dans Thonny et **Enregistrez sous** `main.py` dans votre dossier de projet.

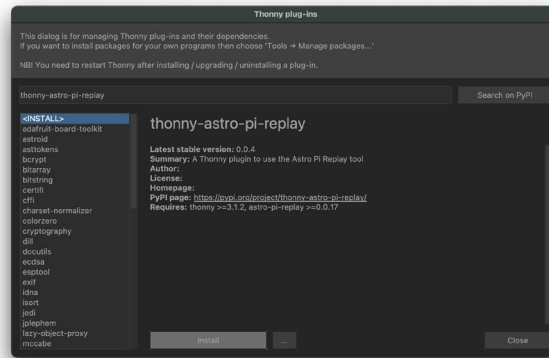


Installer Astro Pi Replay

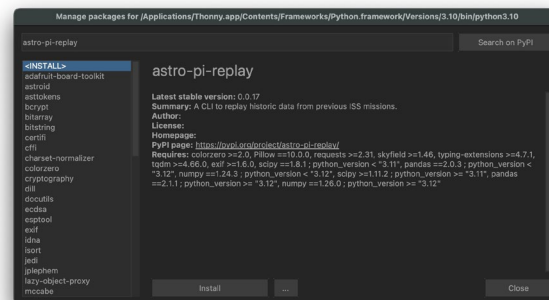
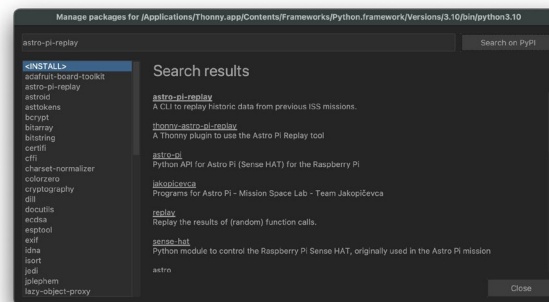
Ensuite, vous allez installer l'outil Astro Pi Replay, qui vous permet de simuler l'utilisation d'un Sense HAT ou d'une caméra d'Astro Pi pour recueillir des données depuis l'espace.

Pour installer l'outil Astro Pi Replay, ouvrez Thonny, puis cliquez sur **Tools (Outils) > Manage plug-in** (Gérer les plug-ins), et recherchez `-thonny astro-pi-replay`. Sélectionnez le plug-in approprié, puis cliquez sur Install (Installer).



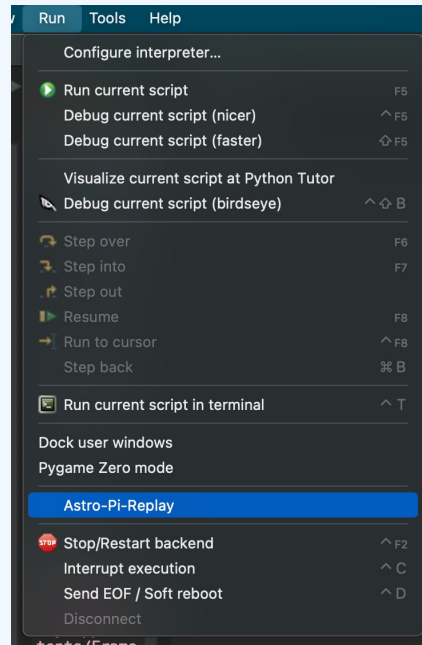


Ensuite, cliquez sur **Tools (Outils) > Manage packages (Gérer les paquets)**, et recherchez **astro-pi-replay**. Sélectionnez le paquet approprié, puis cliquez sur **Install (Installer)**.



Vous devrez fermer et redémarrer Thonny pour terminer l'installation.

L'outil Astro Pi Replay fonctionne en réutilisant un ensemble d'anciennes photos prises sur l'ISS. Lorsque votre code demande de prendre une photo, au lieu d'accéder à une caméra, la bibliothèque sélectionne une photo pour la réutiliser comme si elle venait d'être capturée « en direct ».



Utilisation du plug-in Astro Pi Replay Pour exécuter votre code à l'aide du plug-in Astro Pi Replay, il ne faut **pas** cliquer sur le bouton vert **Run** (Exécuter). Au lieu de cela, cliquez sur le menu **Run** (Exécuter) puis sur **Astro-Pi-Replay**. Votre code est alors exécuté comme s'il fonctionnait sur l'Astro Pi. Bien que toutes les fonctions de la bibliothèque `picamera` soient disponibles, nombre des réglages et paramètres de `picamera` qui entraîneraient normalement la capture d'une image différente sont ignorés lorsque le code est exécuté avec Astro Pi Replay. De plus, la plupart des attributs de l'objet `PiCamera` sont ignorés. Par exemple, un réglage de l'attribut de résolution autre que (4056, 3040) n'a aucun effet lors de la simulation sur Astro Pi Replay, mais il modifie la résolution lorsqu'il est exécuté sur un Astro Pi dans l'espace.

Calcul à partir de données historiques

Vous pouvez commencer par apprendre comment écrire un programme pour estimer la vitesse de l'ISS en utilisant le module `caméra` avec notre guide de projet **Calculer la vitesse de l'ISS à partir de photos** (<https://projects.raspberrypi.org/en/projects/astropi-iss-speed/0>). Une fois que vous avez écrit un programme, vous pouvez le tester en utilisant différentes images ou ensembles de données afin d'améliorer l'exactitude de votre estimation. Voici quelques exemples d'images et de données que vous pouvez utiliser :

- **Photos Astro Pi Mission Space Lab 2022/23** (<https://www.flickr.com/photos/raspberrypi/collections/72157722152451877/>)
- **Données Astro Pi Mission Space Lab 2022/23** (<https://docs.google.com/spreadsheets/d/1RjPEp2IHVB6For65wuUQdWntsg1H5sHWpYUtLzK9LCM/edit?usp=sharing>)

Simuler l'exécution de votre programme en temps réel

Vous pouvez commencer en utilisant les `sense_hat` et `picamera` bibliothèques et en simulant l'exécution de votre programme en temps réel. Pour simuler la lecture des données du Sense HAT et la capture des photos depuis la caméra, utilisez l'outil Astro Pi Replay. L'outil est simple à utiliser : au lieu d'exécuter votre programme en cliquant sur le bouton vert **Run** (Exécuter), ouvrez le menu Run (Exécuter), puis cliquez sur **Astro-Pi-Replay**.

Prendre des mesures avec le Sense HAT

Pour calculer la vitesse de l'ISS, vous devez recueillir des données à partir des capteurs du Sense HAT. Consultez notre guide de projet **Premiers pas avec le Sense HAT** (<https://projects.raspberrypi.org/en/projects/getting-started-with-the-sense-hat>) pour découvrir comment faire.

N'oubliez pas que vous devrez exécuter votre code à l'aide du plug-in **Astro Pi Replay** sur Thonny.

Prendre des photos avec la caméra

Vous pouvez également utiliser la caméra pour prendre des photos de la Terre à utiliser dans votre programme. Vous pouvez utiliser notre guide de projet **Premiers pas avec le module caméra** (<https://projects.raspberrypi.org/fr-FR/projects/getting-started-with-picamera>) pour découvrir comment faire. Cependant, si vous ne disposez pas d'un module Raspberry Pi avec caméra pour tester votre code, vous pouvez toujours l'exécuter avec le plug-in Astro Pi Replay.

Voici un exemple de programme simple pour tester le module Astro Pi Replay :

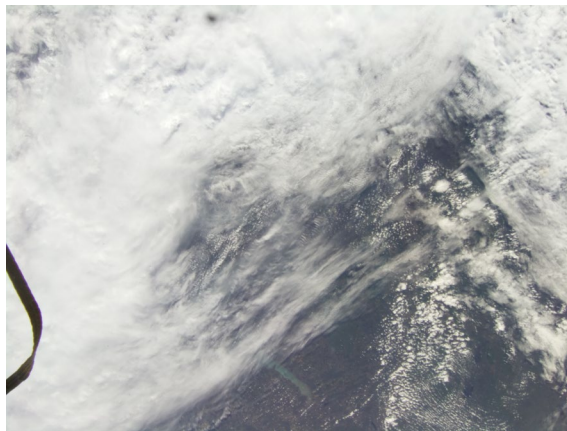
```
# Import the PiCamera class from the picamera module
from picamera import PiCamera

# Create an instance of the PiCamera class
cam = PiCamera()

# Set the resolution of the camera to 4056x3040 pixels
cam.resolution = (4056, 3040)

# Capture an image
cam.capture("image1.png")
```

Il simule la prise d'une photo sur l'ISS et l'enregistre dans un fichier appelé `image1.jpg`. Si vous ouvrez ce fichier, vous devriez voir exactement la photo ci-dessous.



La `picamera` bibliothèque offre un grand nombre de fonctions et de réglages de caméra. Vous pouvez voir des exemples plus avancés sur la page « **Recettes de base** » (<https://picamera.readthedocs.io/en/release-1.13/recipes1.html>) du site web de picamera, mais attention : si votre code est exécuté sur l'ISS, il prendra des photos dans diverses conditions météorologiques avec différents nuages, paysages et degrés de luminosité.

Toutes les fonctions de la `picamera` bibliothèque seront disponibles sur l'Astro Pi dans l'espace, mais toutes ne peuvent pas être simulées par le plug-in Astro Pi Replay. De plus amples informations sont disponibles ici.

Capture de séquences

Avec une boucle `for`, il est très simple de prendre une séquence d'images en appelant plusieurs fois la fonction de capture. L'exemple ci-dessous prend trois photos successivement. Il enregistre également les images sous forme de fichiers PNG au lieu de fichiers JPEG.

Créez un nouveau fichier appelé `camera-sequence.py` et, dans ce fichier, tapez les lignes suivantes :

```
# Import the PiCamera class from the picamera module
from picamera import PiCamera

# Create an instance of the PiCamera class
cam = PiCamera()

# Set the resolution of the camera to 4056x3040 pixels
cam.resolution = (4056, 3040)

# Capture three images using a loop
for i in range(3):
    # Capture an image and save it
    # with a file name like
    # "image0.png", "image1.png", etc.
    cam.capture(f"image{i}.png")
```

Exécutez ce code à l'aide du plug-in Astro Pi Replay en cliquant sur **Run (Exécuter)**

> **Astro-Pi-Replay.** Plans de numérotation des images et des fichiers

En présence de nombreux fichiers du même type, il est conseillé de suivre une convention de nommage. Dans l'exemple ci-dessus, nous utilisons une numérotation évidente (`image1.jpg`, `image2.jpg`, etc.) pour organiser nos fichiers.

Si vous avez besoin de plus d'aide pour utiliser la caméra, consultez l'étape « **Capturer des images fixes avec du code Python** » (<https://projects.raspberrypi.org/en/projects/getting-started-with-picamera/5>) de notre guide de projet « Premiers pas avec le module caméra ».

Mettez votre fichier `main.py` à jour pour capturer des images ou des données Sense HAT en temps réel. 

Localiser l'ISS

Vous pourrez télécharger jusqu'à 42 photos que vous prendrez sur l'ISS. Il peut être bon de savoir exactement où une

image a été prise, et c'est quelque chose que vous pouvez faire `orbit` et `exif` disponibles sur les Astro Pi. facilement avec les bibliothèques

Voici un exemple de programme qui, lorsqu'il sera exécuté avec le plug-in Astro Pi Replay, créera une nouvelle image appelée `gps_image1.jpg`. La fonction `custom_capture` aura défini les métadonnées Exif de l'image pour inclure la **latitude** et la **longitude** de l'ISS au moment de la photo. Il existe plusieurs façons de formater les angles de latitude et de longitude (<https://www.britannica.com/science/latitude>) et d'utiliser la fonction `custom_capture`. Vous devrez adapter ce code en fonction de votre propre programme.

```

from orbit import ISS
from picamera import PiCamera

cam = PiCamera()
cam.resolution = (4056,3040)

def convert(angle):

    # Convert a `skyfield` Angle to an Exif-appropriate #
    # representation (positive rationals)
    # e.g. 98° 34' 58.7 to "98/1,34/1,587/10"

    # Return a tuple containing a Boolean and the converted angle, #
    # with the Boolean indicating if the angle is negative

    sign, degrees, minutes, seconds = angle.signed_dms()
    exif_angle = f'{degrees:.0f}/1,{minutes:.0f}/1,{seconds*10:.0f}/10'
    return sign < 0, exif_angle

def custom_capture(iss, camera, image):
    # Use `camera` to capture an `image` file with lat/long Exif data
    point = iss.coordinates()

    # Convert the latitude and longitude to Exif-appropriate #
    # representations
    south, exif_latitude = convert(point.latitude)
    west, exif_longitude = convert(point.longitude)

    # Set the Exif tags specifying the current location
    camera.exif_tags['GPS.GPSLatitude'] = exif_latitude
    camera.exif_tags['GPS.GPSLatitudeRef'] = "S" if south else "N"
    camera.exif_tags['GPS.GPSLongitude'] = exif_longitude
    camera.exif_tags['GPS.GPSLongitudeRef'] = "W" if west else "E"

    # Capture the image
    camera.capture(image)

capture(ISS(), cam, "gps_image1.jpg")

```

Notez que la latitude et la longitude sont des Angl^es tandis que l'altitude est une Distance. La documentation Skyfield explique comment passer d'une représentation d'angle à une autre (<https://rhodesmill.org/skyfield/api-units.html#skyfield.units.Angle>) et comment exprimer une distance en différentes unités (<https://rhodesmill.org/skyfield/api-units.html#skyfield.units.Distance>).

Apprentissage automatique avec l'accélérateur Coral

Si vous avez accès à un accélérateur Coral pour l'apprentissage automatique `tensorflow_lite`, consultez notre guide de projet **Classification des images** (<https://projects.raspberrypi.org/en/projects/image-id-coral/2>). Vous suivrez l'entraînement d'un modèle d'apprentissage automatique pour le classement des images et vous testerez l'utilisation de la bibliothèque. Vous pouvez ensuite adopter une approche similaire pour classer les images utilisées lorsque vous exécuterez votre programme avec Astro Pi Replay, ou sur l'ISS.

Une fois ce projet terminé, vous pouvez consulter la **page d'exemples de Coral** (<https://coral.ai/examples/>) et **cette page de GitHub** (<https://github.com/robmarkcole/satellite-image-deep-learning#datasets>) afin de vous en inspirer pour appliquer les techniques d'apprentissage automatique à votre propre expérience.

Écrire votre fichier de résultats

Pour que votre contribution réussisse l'essai d'Astro Pi Mission Control, votre programme doit écrire un fichier appelé `result.txt` qui contient votre estimation de la vitesse de l'ISS. Ce fichier doit être au format texte (.txt) et contenir votre estimation jusqu'à 5 décimales. Merci de ne pas inclure d'autres données dans ce fichier.

```
7.12345
```

Exemple de fichier `result.txt` pour une estimation de vitesse moyenne.

Voici un exemple de programme qui va écrire un fichier `.txt` appelé `result.txt` et contenant une estimation de vitesse à 5 décimales en kilomètres par seconde (km/s). Vous devrez adapter ce code en fonction de votre propre programme.


```
estimate_kmps = 7.1234567890 # Replace with your estimate

# Format the estimate_kmps to have a precision
# of 5 significant figures
estimate_kmps_formatted = "{:.5f}".format(estimate_kmps)

# Create a string to write to the file
output_string = estimate_kmps_formatted

# Write to the file
file_path = "result.txt" # Replace with your desired file path
with open(file_path, 'w') as file:
    file.write(output_string)

print("Data written to", file_path)
```

Mettez à jour votre fichier `main.py` pour qu'il écrive un fichier appelé `result.txt` lorsqu'il est exécuté. 

N'oubliez pas de consulter les règles concernant les fichiers et le nommage des fichiers dans le **Règlement de Mission Space Lab** (<https://astro-pi.org/mission-space-lab/rulebook>).

Étape 4 Optimiser votre programme pour l'ISS

Pour les astronautes, travailler dans l'espace signifie travailler sous des contraintes très strictes, et il en va de même pour vous ! Cette section explique comment s'assurer que votre code se comportera comme prévu lors de son exécution sur l'ISS et comment gérer des éléments comme les ressources et les erreurs.

Exécution d'une expérience pendant 10 minutes

Chaque programme exécuté sur les Astro Pi dispose d'un créneau de 10 minutes pour estimer la vitesse de l'ISS. Votre programme devra surveiller l'heure et s'arrêter correctement avant la fin des 10 minutes pour s'assurer qu'aucune donnée ne soit perdue.

Pour arrêter un programme Python après une durée spécifique, vous pouvez utiliser la bibliothèque Python `datetime`. Cette bibliothèque permet de travailler facilement avec les heures et de les comparer. Effectuer cette action sans recourir à cette bibliothèque n'est pas toujours simple : il est facile de se tromper en utilisant les mathématiques normales.

En enregistrant et en stockant l'heure du début de l'expérience, nous pouvons vérifier plusieurs fois si l'heure actuelle est supérieure à l'heure de début, plus un certain nombre de minutes, de secondes ou d'heures. Dans le programme ci-dessous, cette technique est utilisée pour imprimer « Bonjour de l'ISS » toutes les secondes pendant 1 minute :

```
from datetime import datetime, timedelta
from time import sleep

# Create a variable to store the start time
start_time = datetime.now()
# Create a variable to store the current time #
# (these will be almost the same at the start)
now_time = datetime.now()
# Run a loop for 1 minute
while (now_time < start_time + timedelta(minutes=1)):
    print("Hello from the ISS")
    sleep(1)
    # Update the current time
    now_time = datetime.now()
# out of the loop - stopping
```

Mettez à jour votre fichier `main.py` pour utiliser la bibliothèque `datetime` afin d'arrêter votre programme avant la fin du créneau de 10 minutes.



Remarque : lorsque vous définissez la durée d'exécution de votre programme, tenez compte de la durée de réalisation d'un cycle par votre boucle. Ainsi, si vous voulez exploiter l'intégralité du créneau de 10 minutes disponible et que chaque boucle de votre code dure 2 minutes, alors votre `timedelta` doit être $10 - 2 = 8$ minutes pour s'assurer que votre programme se termine avant la fin des 10 minutes.

Utilisation de chemins relatifs

Votre programme sera stocké à un autre endroit lorsqu'il sera déployé sur l'ISS. Il est donc très important d'éviter d'utiliser des chemins d'accès absolus lors de l'écriture de votre fichier (ou de tout autre fichier que vous pourriez écrire). Utilisez le code ci-dessous pour déterminer dans quel dossier le fichier `main.py` est actuellement stocké, lequel est appelé `base_folder` :

```
from pathlib import Path
base_folder = Path( file ).parent.resolve()
```

Ensuite, vous pouvez enregistrer vos données dans un fichier situé dans ce « `base_folder` » :

```
data_file = base_folder / "data.csv"
for i in range(10):
    with open(data_file, "w", buffering=1) as f:
        f.write(f"Some data: {i}")
```

N'oubliez pas de consulter les règles concernant les fichiers et le nommage des fichiers dans le **Règlement de Mission Space Lab** (<https://astro-pi.org/mission-space-lab/rulebook>).

Fermer les ressources

À la fin de l'expérience, il est conseillé de refermer toutes les ressources que vous avez ouvertes. Il peut s'agir de fermer les fichiers que vous avez ouverts :


```
file = open(file)
file.close()
```

ou de fermer la caméra :

```
from picamera import PiCamera

cam = PiCamera()
cam.close()
```

Évitez de fermer et de rouvrir la caméra en boucle : cela risque de saturer la mémoire de Raspberry Pi et votre programme risque de ne pas être autorisé à fonctionner sur l'ISS. Fermez la caméra seulement lorsque vous avez terminé de prendre des photos.

Relisez votre fichier `main.py` et mettez-le à jour afin qu'il ferme toutes les ressources de façon appropriée. 

Se préparer à l'imprévu

Un programme peut échouer pour de nombreuses raisons, mais avec un peu d'anticipation et de planification, il est possible de gérer ces défaillances pour éviter le plantage et la perte de toute chance de capturer des données et des images à bord de l'ISS. Dans cette section, nous allons essayer de trouver des moyens d'améliorer votre programme afin qu'il ait la meilleure chance de fonctionner comme prévu en cas d'imprévu.

Traitement des exceptions

Une exception est un événement inattendu qui se produit pendant qu'un programme est en cours d'exécution et auquel il ne sait pas comment réagir. Cela peut faire planter le programme, à moins qu'il ait une procédure à suivre en cas de problème.


Consultez le tutoriel d'Ada Computer Science sur le **Traitement des exceptions** (https://adacomputerscience.org/concepts/design_exception?examBoard=all&stage=gcse).

Mettre le fichier en mémoire tampon

Normalement, lorsque vous écrivez dans un fichier à l'aide de la fonction `Open` (Ouvrir), Python ne sauvegarde pas le fichier immédiatement sur le disque. Au lieu de cela, il conserve le contenu du fichier pour l'enregistrer dans une zone de stockage temporaire de la mémoire de l'ordinateur, appelée mémoire tampon. Python fait cela afin de pouvoir choisir le meilleur moment pour écrire sur le disque, ce qui n'a normalement pas d'importance pour nous. Mais tant que les données sont dans la mémoire tampon et pas encore enregistrées sur le disque, il y a un risque qu'elles soient perdues en cas d'erreur. Pour éviter cela, nous pouvons dire à Python d'enregistrer la mémoire tampon sur le disque à la fin de chaque ligne de texte en définissant l'argument `buffering` sur 1 :

```
with open("some_file.txt", "w", buffering=1) as f:
    f.write("example data")
```

Remarque : si vous écrivez des octets dans un fichier (avec l'argument `"wb"`), alors vous devez dire à Python de ne pas du tout utiliser la mémoire tampon et d'écrire les données immédiatement sur le disque. Vous pouvez le faire en paramétrant l'argument `buffering` sur 0.

Relisez votre programme et réfléchissez à la nécessité de définir le mode tampon lors de l'écriture dans un fichier. 

Journalisation

Si votre programme échoue, il est toujours utile d'avoir un enregistrement de ce qui s'est passé, afin que vous puissiez le corriger pour la prochaine fois. La bibliothèque Python `Logzero` (documentation disponible ici (<https://logzero.readthedocs.io/en/latest/>)) permet de consigner facilement des notes sur ce qui se passe dans votre programme. Vous pouvez enregistrer autant d'informations que vous le souhaitez sur ce qui se passe dans votre programme (chaque itération de boucle, chaque fois qu'une fonction importante est appelée) et si vous avez des éléments conditionnels dans votre programme, `Logzero` enregistrera le chemin parcouru par le programme (`if` ou `else`).

Voici un exemple basique sur la façon dont `logzero` peut être utilisé pour suivre les itérations de boucle :

```
from logzero import logger, logfile
from time import sleep

logfile("events.log")

for i in range(10):
    logger.info(f"Loop number {i+1} started")
    ...
    sleep(60)
```

Les deux principaux types d'entrée de journal que vous pouvez utiliser sont : `logger.info()` pour enregistrer une information et `logger.error()` si vous rencontrez une erreur inattendue ou que vous gérez une exception. Il existe aussi `logger.warning()` et `logger.debug()`.

Nous vous recommandons de toujours utiliser la bibliothèque `Logzero` (pour enregistrer les événements importants qui se déroulent au cours de votre expérience), même si vous écrivez également des données de capteur dans un fichier.

Lorsque vous avez terminé d'écrire votre programme et que vous pensez qu'il fournit l'estimation de la vitesse de l'ISS au format approprié et qu'il suit les meilleures pratiques comme la journalisation et le traitement des erreurs, il est essentiel de tester votre programme minutieusement à l'aide d'Astro Pi Replay.

Étape 5 Améliorer la précision de votre programme

La vitesse moyenne de l'ISS n'est pas un secret, mais elle n'est pas vraiment constante et peut être affectée par plusieurs facteurs, comme l'altitude. Si l'altitude a changé, l'ISS doit avoir allumé ses propulseurs et, par conséquent, la vitesse doit avoir changé. Pour augmenter vos chances d'exécuter votre programme sur l'ISS, demandez-vous si votre programme est suffisamment sensible aux changements subtils qui affecteront sa vitesse de déplacement.

Sur le site **N2YO.com** (<https://www.n2yo.com/?s=25544>), vous pouvez voir des données concernant l'ISS en direct, et celles-ci montrent que son altitude change pendant sa rotation autour de la Terre.

Vous pouvez également voir quand l'ISS passera au-dessus de vous.

Moyennes

Si votre programme calcule plusieurs estimations de la vitesse de l'ISS (par exemple, en calculant la vitesse à partir de séquences de deux photos), vous devrez décider comment réduire ces estimations en un seul nombre lorsque vous écrivez votre fichier `result.txt`. Si vous avez utilisé une simple moyenne (définition de la **moyenne** : <https://fr.wikipedia.org/wiki/Moyenne>), pouvez-vous explorer la précision d'autres mesures statistiques, comme la médiane et d'autres centiles ?

L'amélioration de la précision de vos estimations laisse une grande place à la créativité. Une méthode consiste à sélectionner les photos ou données que vous utilisez pour calculer votre estimation. Si vous arrivez à déterminer qu'une séquence de données spécifique est la plus fiable, vous pouvez donner plus de poids à ces données dans votre estimation finale.

Faites attention à ce que votre programme ne soit pas trop sensible à la séquence exacte illustrée par Astro Pi Replay : la séquence sur l'ISS sera différente, et c'est avant tout sur l'ISS que votre programme doit donner des résultats précis !

Si votre méthode de calcul de la vitesse est basée sur le projet **Calculer la vitesse de l'ISS à partir de photos** (<https://projects.raspberrypi.org/en/projects/astropi-iss-speed/0>), vous pouvez peut-être utiliser des techniques issues de la vision numérique ou de l'apprentissage automatique pour classer les photos qui permettent de calculer l'estimation plus facilement. Par exemple, vous pouvez effectuer une inférence par apprentissage automatique en temps réel afin d'évaluer la précision de votre estimation ou les conditions sous l'ISS.

Évaluez la précision de l'estimation de la vitesse de l'ISS produite par votre programme en utilisant Astro-Pi-Replay.



Étape 6 Tester votre programme

Vous devez maintenant le tester en utilisant Astro Pi Replay. Cela permet de maximiser les chances de réussite de votre contribution et d'être sûrs qu'elle fonctionnera à bord de l'ISS. Lorsqu'Astro Pi Mission Control recevra votre programme, il sera testé et évalué à l'aide d'Astro Pi Replay, et si l'essai est réussi, sur un véritable Astro Pi. Des centaines d'équipes soumettent des programmes chaque année pour le challenge et, malheureusement, il n'y a pas assez de temps pour vérifier les maladresses ou corriger les erreurs de code complexes. Si votre programme présente des erreurs lors de notre test, il ne sera pas admissible pour l'exécution sur l'ISS.

Si vous avez suivi ce guide dès le début, vous devriez déjà avoir installé Astro Pi Replay. Les instructions d'installation figurent plus haut dans ce guide et dans la documentation d'Astro Pi Replay.

Pour tester votre programme et simuler son fonctionnement à bord de l'ISS, exécutez votre `main.py` code avec le plug-in Astro-Pi en cliquant **sur Run (Exécuter) > Astro-Pi-Replay**.

Votre code doit fonctionner pendant moins de 10 minutes, puis s'arrêter.

Une fois terminé, vérifiez qu'il a créé un fichier `result.txt` avec une structure valide dans votre dossier de projet. Observez aussi tous les autres fichiers de sortie créés par votre projet. Vos fichiers enregistrés dépassent-ils la limite de 250 Mo ou incluent-ils des types de fichiers qui ne sont pas autorisés dans les règles ? Enfin, vérifiez que vos journaux ne contiennent pas d'erreur.

Si vous constatez des erreurs, ou si le programme ne fait pas ce que vous aviez prévu, vous devez résoudre ces problèmes avant de soumettre votre code, afin de vous donner toutes les chances d'atteindre la « phase de vol ». Vous pouvez exécuter `Astro-Pi-Replay` pour refaire l'expérience autant de fois que nécessaire jusqu'à ce que vous soyez certains que votre programme fonctionne.

Testez votre programme avec `Astro-Pi-Replay` et vérifiez le résultat pour détecter tout problème ou comportement inattendu.



Liste de contrôle du programme

Votre programme sera également analysé afin de s'assurer qu'il est conforme au règlement. Prenez le temps de les lire maintenant et de relire à nouveau votre programme pour vous assurer qu'il correspond bien aux critères.

Vérifiez que votre programme est conforme au **Règlement de Mission Space Lab** (<https://astro-pi.org/mission-space-lab/rulebook>).



De nombreuses équipes Mission Space Lab n'ont pas pu exécuter leur programme sur l'ISS en raison de maladresses ou d'erreurs courantes dans leurs programmes. Vous trouverez ci-dessous une liste d'erreurs courantes avec la description des raisons pour lesquelles elles affectent les programmes exécutés sur l'ISS.

Erreurs courantes



Ouverture et fermeture répétées de la caméra

Si vous fermez et rouvrez la caméra à plusieurs reprises, par exemple dans une boucle, vous risquez de saturer la mémoire du Raspberry Pi et d'être recalés par Mission Control.

Ne pas utiliser les images en résolution intégrale

Attention : la distance d'échantillonnage au sol, ou GSD (*ground sampling distance*), peut changer avec la résolution finale de l'image.

La valeur utilisée dans le projet Calculer la vitesse de l'ISS à partir de photos (<https://projects.raspberrypi.org/en/projects/astropi-iss-speed/0>) est valable pour les images en résolution intégrale (4056x3040) mais pas nécessairement pour les résolutions inférieures. Pour cette raison, nous vous recommandons de capturer des images en résolution intégrale.

Stockage de plus de 42 images

Votre programme n'est pas autorisé à conserver plus de 42 images à la fin des 10 minutes, mais il peut en stocker davantage pendant l'exécution.

Intervention de l'utilisateur

Votre programme ne peut pas s'appuyer sur une interaction avec un astronaute pour fonctionner.

Utilisation de la matrice LED

Votre programme n'est pas autorisé à utiliser la matrice LED.

Mauvaise documentation

Lorsque vous avez créé un logiciel utile et que vous voulez le partager avec d'autres personnes, il est essentiel de produire une documentation qui aide les gens à comprendre ce que fait le programme, comment il fonctionne et comment l'utiliser. Assurez-vous que votre programme contient des commentaires et que la méthode utilisée pour déterminer la vitesse de l'ISS est bien expliquée.

Ce projet (<https://projects.raspberrypi.org/en/projects/documenting-your-code/>) vous explique comment ajouter des commentaires utiles dans votre programme.

Remarque : toute tentative de masquer ou de rendre difficile à comprendre ce que fait une partie du code peut entraîner la disqualification du projet. Et bien sûr, votre code ne doit pas contenir de langage grossier ou injurieux.

Suradaptation aux données de réexécution

Votre code doit réagir aux images et aux données du capteur embarqué sur l'ISS et ne doit pas dépendre des repères spécifiques ou des zones géographiques présentés dans la ou les séquence(s) utilisée(s) dans Astro Pi Replay.

Utilisation de chemins d'accès absolus

Assurez-vous de ne pas utiliser de chemins spécifiques pour vos fichiers de données. Utilisez la variable file.

Ne pas enregistrer les données immédiatement

Assurez-vous que toutes les données expérimentales sont écrites dans un fichier dès qu'elles sont enregistrées. Évitez d'enregistrer des données dans une liste ou un dictionnaire interne au fur et à mesure de votre progression, puis de les écrire dans un fichier à la fin de l'expérience. En effet, si votre expérience s'arrête brusquement en raison d'une erreur ou du dépassement du délai de 10 minutes, vous n'obtiendrez aucune donnée.

Manque d'espace

Vous pouvez générer jusqu'à 250 Mo de données. Souvenez-vous que la taille d'un fichier image dépend non seulement de la résolution, mais également de la quantité de détails présents dans l'image : une photo d'un mur blanc et nu sera plus petite qu'une photo de paysage. L'utilisation d'Astro Pi Replay vous donnera une bonne idée du nombre d'images que vous pourrez prendre.

Oublier d'appeler votre fonction

Nous avons vu des cas où des équipes avaient écrit une fonction, mais avaient oublié de l'appeler dans leur `main.py`. Oups !

Enregistrement dans des répertoires qui n'existent pas

Un certain nombre d'équipes souhaitent ranger leurs données dans des répertoires, par exemple pour les données, les images, etc. En soi, c'est une très bonne chose, mais il est facile d'oublier de créer ces répertoires avant d'y écrire quelque chose.

Travail en réseau

Pour des raisons de sécurité, votre programme n'est pas autorisé à accéder au réseau de l'ISS. Il ne doit pas essayer d'ouvrir une prise, d'accéder à Internet, ou d'établir une quelconque connexion réseau. Cela inclut les connexions réseau locales vers l'Astro Pi lui-même. Quand vous testez votre programme, vous devez désactiver la connexion réseau pour vérifier que votre programme s'exécute correctement sans connexion Internet.

Essayer d'exécuter un autre programme

En plus de ne pas pouvoir utiliser de réseau, votre programme n'est pas autorisé à exécuter un autre programme ou une commande que vous saisissez normalement dans la fenêtre du terminal du Raspberry Pi, comme `vcgencmd`.

Multithreading (multifil)

Si vous avez besoin de faire plus d'une chose à la fois, vous pouvez utiliser un processus multithread. Plusieurs bibliothèques Python permettent d'inclure ce type de mode multitâche dans votre code. Cependant, pour le faire sur les Astro Pi, vous avez le droit d'utiliser uniquement la bibliothèque `threading`.

Utilisez la bibliothèque `threading` seulement si cela est absolument nécessaire. Gérer des threads peut être complexe, et comme votre programme sera exécuté dans une séquence qui contiendra beaucoup d'autres programmes, nous devons nous assurer que le programme précédent s'est terminé sans problème avant de commencer le suivant. Les threads indésirables peuvent se comporter de manière inattendue et utiliser trop de ressources du système. Si vous décidez d'utiliser des threads dans votre code, vous devez vérifier qu'ils sont tous gérés soigneusement et fermés proprement à la fin du programme. En outre, vous devez veiller à ce que les commentaires inclus dans votre code expliquent clairement comment cela est réalisé.

Réglage du temps d'exécution du programme sur une valeur trop faible

Certaines équipes règlent le temps d'exécution de leur programme sur une valeur faible (par exemple, 1 minute) pour les essais, puis oublient de revenir à une valeur appropriée. Assurez-vous d'exploiter votre créneau temporel au maximum.

Relisez à nouveau votre programme. Pouvez-vous repérer l'une des erreurs courantes dans votre programme ?



Étape 7 Déposer votre travail

Si vous êtes satisfaits de votre programme, que vous l'avez exécuté avec Astro Pi Replay, que vous avez lu le règlement sur le site Web d'Astro Pi et que vous avez vérifié et revérifié la liste de contrôle des programmes Mission Space Lab, il ne vous reste plus qu'à zipper votre travail et à demander à votre mentor de nous le soumettre.

Préparer un fichier Zip

Si vous ne savez pas comment compresser votre dossier de projet dans un fichier zip, adressez-vous à votre mentor, qui sera en mesure de vous indiquer la méthode appropriée pour votre système d'exploitation.

Éliminer les fichiers inutiles

La taille de votre fichier zip ne doit pas dépasser 3 Mo, sauf si le fichier comprend un modèle .tflight, auquel cas elle peut aller jusqu'à 7 Mo pour tenir compte de la taille du modèle. Cela signifie, par exemple, que vous ne pouvez pas fournir vos propres fichiers d'éphémérides (par exemple **de421.bsp** : https://naif.jpl.nasa.gov/pub/naif/generic_kernels/spk/planets/a_old_versions/de421.bsp ou **de440s.bsp** : https://naif.jpl.nasa.gov/pub/naif/generic_kernels/spk/planets/de440s.bsp) étant donné qu'ils dépassent la taille limite.

Les deux fichiers **de421.bsp** et **de440s.bsp** sont disponibles sur les Astro Pi. Si votre programme en a besoin, vous pouvez y accéder dans la bibliothèque **orbit**, comme indiqué dans le code suivant :

```
from orbit import de421, de440s
print(de440)
print(de440s)
```

Générez un fichier zip pour votre projet et demandez à votre mentor de nous le soumettre avant la date limite.

Autres ressources

De nombreuses ressources sont disponibles pour vous aider à réussir chaque étape de votre parcours Astro Pi.

Vous êtes bloqués ? N'hésitez pas à **nous contacter** (enquiries@astro-pi.org) et nous ferons de notre mieux pour vous aider !

- Les **exemples de projets Coral** (<https://coral.ai/examples/>) et les **exemples de caméras Coral** (<https://github.com/google-coral/examples-camera>) donnent un bon aperçu de ce que l'on peut faire avec l'accélérateur Coral ML. Les **tutoriels OpenCV Python** (https://docs.opencv.org/4.x/d6/d00/tutorial_py_root.html) expliquent comment faire toutes sortes de choses comme l'apprentissage automatique, la détection de contours et le suivi des objets.
- La **galerie d'exemples EarthPy** (https://earthpy.readthedocs.io/en/latest/gallery_vignettes/index.html) montre comment traiter les images satellitaires.
- Cette **page GitHub** (<https://github.com/orbitalindex/awesome-space>) contient une liste de ressources liées à l'espace soigneusement choisies.
- Cette **liste d'ensembles de données satellitaires annotés** (https://github.com/Seyed-Ali-Ahmadi/Awesome_Satellite_Benchmark_Datasets) peut être utilisée pour entraîner les modèles d'apprentissage automatique.
- Les équipes qui souhaitent corser le challenge trouveront une ressource complète sur l'apprentissage automatique aérien et satellitaire **ici** (<https://github.com/robmarkcole/satellite-image-deep-learning#datasets>). Attention, c'est plutôt pointu.

Enfin, n'oubliez pas que des liens vers la documentation de chaque bibliothèque sont disponibles dans la section 23-24 de Mission Space Lab si vous avez besoin d'informations spécifiques sur l'utilisation d'une bibliothèque particulière.

Bonne chance !

Publié par la Fondation Raspberry Pi (<https://www.raspberrypi.org>) sous licence Creative Commons (<https://creativecommons.org/licenses/by-sa/4.0/deed.fr>).

Consulter les projets et les licences sur le site de GitHub (<https://github.com/RaspberryPiLearning/mission-space-lab-creator-guide>)