Étape 5 Localiser l'ISS

En utilisant la bibliothèque Python `skyfield` , tu peux calculer la position des objets spatiaux dans notre système solaire. Cela comprend le Soleil, la Lune, les planètes et de nombreux satellites terrestres comme l'ISS. Tu peux utiliser l'emplacement actuel de l'ISS au-dessus de la Terre pour déterminer si elle survole la terre ou la mer, ou savoir au-dessus de quel pays elle passe.

Qu'est-il arrivé à la bibliothèque Ephem ?

Si ton équipe a participé aux défis précédents, tu te souviens peut-être que la bibliothèque `ephem` était utilisée pour calculer la position de l'ISS. Cette bibliothèque est devenue obsolète et a été remplacée par son successeur, `skyfield`.

Des données de télémétrie à jour sont nécessaires pour calculer avec précision la position de l'ISS (ou de tout autre satellite en orbite autour de la Terre). Pour t'éviter de devoir obtenir et manipuler ces données, le système d'exploitation de Vol offre le package Python, qui utilise `skyfield` pour créer un objet `ISS` que tu peux importer dans ton programme :

Données de télémétrie

Pour des calculs précis, `skyfield` requiert l'ensemble d'éléments à deux lignes (TLE) le plus récent pour l'ISS. TLE est un format de données utilisé pour transmettre des ensembles de paramètres orbitaux qui décrivent les orbites des satellites terrestres.

Lorsque tu importes l'objet `ISS` à partir de la bibliothèque `orbit`, une tentative de récupération des données TLE à partir d'un fichier `iss.tle` dans le dossier `/home/pi` est effectué. Si le fichier n'est pas présent mais qu'un connexion Internet est disponible les dernières données seront téléchargées automatiquement dans le fichier `iss.tle`, tu n'as donc pas besoin de t'en soucier

Toutefois, si ton kit Astro Pi n'a pas accès à Internet, tu dois télécharger manuellement les dernières données TLE de l'ISS (http://www.celestrak.com/NORAD/elements/stations.txt), copier les trois lignes ISS dans un fichier appelé `iss.tle`, puis placer ce fichier dans le dossier `/home/pi`. Les données TLE ressembleront à ceci :

```
ISS (ZARYA)
1 25544U 98067A     21162.24455464 .00001369 00000-0 33046-4 0 9995
2 25544 51.6454 12.1174 0003601 83.6963 83.5732 15.48975526287678
```

Lorsque ton code sera exécuté sur la Station spatiale, nous nous assurerons d'utiliser les données de télémétrie les plus précises et les plus à jour.

Tu peux utiliser `ISS` comme toute autre objet `earthsatellite` dans `skyfield` (voir la référence (https://rhodesmill.org/skyfield/api-satellites.html#skyfield.sgp4lib.EarthSatellite) et exemples (https://rhodesmill.org/skyfield/earth-satellites.html)). Par exemple, voici comment calculer les coordonnées de l'emplacement de la Terre qui se trouve actuellement directement sous l'ISS :

```
from orbit import ISS
from skyfield.api import load

# Obtain the current time `t` t = load.timescale().now()
# Compute where the ISS is at time `t` position = ISS.at(t)
# Compute the coordinates of the Earth location directly beneath the ISS location =
position.subpoint()
print(location)
```

Si tu n'es pas intéressé par la définition ou l'enregistrement de l'heure `t`, alors l'objet `ISS` offre également une méthode `coordinates` pratique que tu peux utiliser comme alternative pour récupérer les coordonnées de l'emplacement de la Terre qui se trouve actuellement directement sous l'ISS :

```
from orbit import ISS
location = ISS.coordinates() # Equivalent to ISS.at(timescale.now()).subpoint()
print(location)
```

Remarque : la position actuelle de l'ISS est une estimation basée sur les données de télémétrie et l'heure actuelle. Par conséquent, lorsque tu testes ton programme sur la version de bureau du système d'exploitation de Vol, tu dois vérifier que l'heure du système a été correctement réglée.

En outre, `location` est une `GeographicPosition`, tu peux donc consulter la documentation et voir comment accéder à ses éléments individuels : (https:/rhodesmill.org/skyfield/api-topos.html#skyfield.toposlib.GeographicPositio n) :

```
print(f'Latitude: {location.latitude}')
print(f'Longitude: {location.longitude}')
print(f'Elevation: {location.elevation.km}')
```

Note que la latitude et la longitude sont des `Angles` et que l'altitude est une `Distance`. La documentation explique comment passer d'une représentation d'`Angle` à une autre (https:/rhodesmill.org/skyfield/api-units.html#skyfield.units.Angle) ou comment exprimer la html#skyfield.units.Distance) :dans différentes unités (https:/rhodesmill.org/skyfield/api-units.

```
print(f'Lat: {location.latitude.degrees:.1f}, Long: {location.longitude.degrees:.1f}')
```

Il existe plusieurs façons de représenter la latitude et la longitude, et il est important de choisir la méthode appropriée, surtout lorsque l'on travaille avec des logiciels et des bibliothèques qui attendent un format de données spécifique.

Le code ci-dessus donne la latitude et la longitude en utilisant le format des degrés décimaux (DD), où l'unité de mesure des coordonnées est le degré (°). Il y a 180° de latitude : 90° au nord et 90° au sud de l'équateur. Il y a 360° de longitude : 180° à l'est et 180° à l'ouest du premier méridien (avec une latitude égale à zéro et défini comme un point situé à Greenwich, en Angleterre). Pour spécifier précisément un emplacement, chaque degré peut être exprimé par un nombre décimal, par exemple (-28.277777, 71.5841666).

Une autre approche est le format degrés:minutes:secondes (DMS), où chaque degré est divisé en 60 minutes (') et chaque minute est divisée en 60 secondes ("). Pour une précision encore plus fine, on utilise des fractions de seconde indiquées par un point décimal. Le signe de l'angle indique si le point auquel se réfère la

coordonnée est au nord ou au sud de l'équateur (pour la latitude) et à l'est ou à l'ouest du méridien (pour la longitude).

```
print(f'Lat: {location.latitude.signed_dms()}, Long: {location.longitude.signed_dms()}')
```

Exemple : quel hémisphère ?

Si tu souhaites que ton expérience se déroule lorsque l'ISS se trouve au-dessus d'un emplacement terrestre particulier, tu peux utiliser les valeurs de latitude et de longitude pour déclencher une autre action. N'oublie pas que l'orbite de l'ISS ne survole pas toute la Terre, et que la plus grande partie de la surface de notre planète est constituée d'eau et non de terres. Ainsi, dans ta fenêtre expérimentale de 3 heures, les chances de passer au-dessus d'une ville ou d'un emplacement très précis seront faibles.

Pour savoir comment cela pourrait être utile dans ton programme, modifie le code ci-dessus afin qu'il affiche un message lorsque l'ISS se trouve au-dessus de l'hémisphère sud.

J'ai besoin d'un indice

Ton code doit ressembler à cela :

```
from orbit import ISS

location = ISS.coordinates() latitude = location.latitude.degrees if latitude < 0:
   print("In Southern hemisphere") else:
print("In Northern hemisphere")
```

Exemple : ISS éclairée par la lumière du Soleil

Le comportement de ton code peut différer selon que l'ISS est éclairée ou non par la lumière du Soleil. La bibliothèque `skyfield` permet d'obtenir facilement ces informations pour n'importe quel objet `EarthSatellite`. Peux-tu consulter la documentation et écrire un programme qui affiche toutes les 30 secondes si l'ISS est éclairée ou non par la lumière du Soleil ?

J'ai besoin d'un indice

Ton code doit ressembler à cela :

```
from time import sleep from orbit import ISS
from skyfield.api import load

ephemeris = load('de421.bsp') timescale = load.timescale()

while True:
t = timescale.now()
ISS.at(t).is_sunlit(ephemeris): print("In sunlight")
else:
   print("In darkness") sleep(30)
```

Remarque : en raison de l'altitude de l'ISS, le soleil se lève un peu plus tôt sur cette dernière que sur la surface terrestre, qui se trouve en dessous de la Station spatiale. De même, le soleil se couche sur l'ISS un peu plus tard que sur la surface terrestre, qui se trouve directement en dessous de l'ISS.