Exploiter, à l'aide d'un langage de programmation, des données astronomiques ou satellitaires pour tester les deuxième et troisième lois de Kepler

Il est possible d'utiliser des données astronomiques récupérées sur internet. Cependant, la bibliothèque python **jplephem** associée à la bibliothèque **de423** donne accès directement aux données des éphémérides du laboratoire JPL de la NASA. Pour les télécharger, il suffit d'utiliser les commandes **pip install --upgrade pip** (pour mettre à jour la commande pip) puis **pip install jplephem** pour la bibliothèque et **pip install de423** pour les données. L'utilisation de ces commandes varie en fonction de l'installation de python mais dans la plupart des cas, il suffit de les taper dans le shell de votre IDE et d'attendre qu'elle soit terminée avant de taper la suivante. **Attention de423 fait presque 40 MB**.

On charge ensuite les bibliothèques utiles

```
import de423
from jplephem import Ephemeris
import matplotlib.pyplot as plt
from scipy.optimize import curve_fit
from math import pi
from matplotlib.patches import Polygon
from mpl_toolkits.mplot3d import Axes3D
```

L'accès aux données peut se faire à partir du module **Ephemeris** de la bibliothèque **jplephem**. L'objet qui sera utilisé est stocké dans une variable, ici **eph**.

```
eph = Ephemeris(de423)
```

Les différents corps présents dans cette base de donnée sont listés ci-dessous. Pour la Terre, on utilisera **earthmoon** qui donne la position et la vitesse du centre de gravité du système Terre-Lune. La position de tous ces corps est donnée à partir du centre de gravité du système solaire (donc le Soleil se déplace dans ce référentiel).

```
eph.names
```

On peut visualiser facilement les trajectoires des planètes autour du soleil en utilisant la méthode **position** qui prend comme argument le nom de l'objet et le numéro du jour dans le calendrier julien et qui renvoie les trois composantes de la position de l'objet sous forme d'une liste.

On peut ainsi balayer les jours sur une période afin de récupérer toutes les positions successives et tracer la trajectoire. Ceci peut se faire pour l'ensemble des planètes du système solaire afin d'étudier la première loi de Kepler par exemple.

La méthode **position_and_velocity** permet quant à elle d'obtenir la position et la vitesse du corps considéré. Les positions sont données en km et les vitesses en km/jour.

```
fig = plt.figure("système solaire interne")
# objets internes
objets = ["sun","mercury","venus","earthmoon","mars"]
couleurs = ["orange","gray","brown","blue","red"]
periodes = [5000,89,226,366,688]    # Le soleil se déplace par rapport au centre de gravité du système solaire.
                                    #  On  peut  le  visualiser  aven  utilisant  une période assez longue
for j in range(len(objets)):
    x = []
    y = []
    for i in range(2378556,2378556+periodes[j],1): #1er mars 1800 à 1er mars 1800 + période
        a,b,c = eph.position(objets[j],i) # Position de la planète sur les axes x, y et z
        x.append(a)
        y.append(b)
    plt.plot(x,y,color=couleurs[j],label = objets[j]) # On ne trace ici que le projeté des trajectoires sur un plan
plt.axis("equal") # Repère orthonormé pour visualiser des trajectoires non déformées par les échelles
plt.legend()
plt.show()
```

```
fig = plt.figure("système solaire interne 3D")
ax = Axes3D(fig)
# objets internes
objets = ["sun","mercury","venus","earthmoon","mars"]
couleurs = ["orange","gray","brown","blue","red"]
periodes = [5000,89,226,366,688]
for j in range(len(objets)):
    x = []
    y = []
    z = []
    for i in range(2378556,2378556+periodes[j],3): #1er mars 1800 à 1er mars 1800 + période
```

```
        a,b,c = eph.position(objets[j],i) # Position de la planète sur les axes x, y
et z
        x.append(a)
        y.append(b)
        z.append(c)
    ax.scatter(x, y, z,label = objets[j]) # La même chose en 3 dimensions
#plt.axis("equal") # Repère orthonormé pour visualiser des trajectoires non déformées
ax.set_xlim3d([-2e8,2e8])
ax.set_ylim3d([-2e8,2e8])
ax.set_zlim3d([-2e8,2e8])
plt.legend()
plt.show()
```

On peut faire la même chose pour le système solaire externe (4 planètes plus Pluton intéressante car sa trajectoire est bien elliptique et elle est inclinée par rapport au plan de l'écliptique) mais en utilisant une base de temps un peu plus longue.

```
fig = plt.figure("système solaire externe")
# système solaire externe
objets = ["sun","jupiter","saturn","uranus","neptune","pluto"]
periodes = [90638,4333,10759,30685,60266,90638] # La période pour le soleil peut être
modifiée en fonction des besoins

for j in range(len(objets)):
    x = []
    y = []
    for i in range(2378556,2378556+periodes[j],50): #1er mars 1800 à 1er mars 1800 +
période
        a,b,c = eph.position(objets[j],i)
        x.append(a)
        y.append(b)
    plt.plot(x,y, label = objets[j]) # Trajectoires projetées dans un plan

plt.axis("equal")
plt.legend()
plt.show()
```

```
fig = plt.figure("système solaire externe 3D")
ax = Axes3D(fig)

# système solaire externe
objets = ["sun","jupiter","saturn","uranus","neptune","pluto"]
periodes = [90638,4333,10759,30685,60266,90638]
for j in range(len(objets)):
    x = []
    y = []
    z = []
    for i in range(2378556,2378556+periodes[j],100): #1er mars 1800 à 1er mars 1800
+ période
        a,b,c = eph.position(objets[j],i) # Position de la planète sur les axes x, y
et z
        x.append(a)
        y.append(b)
        z.append(c)
    ax.scatter(x,y,z,label = objets[j]) # trajectoires en 3D

ax.set_xlim3d([-7e9,7e9])
ax.set_ylim3d([-7e9,7e9])
ax.set_zlim3d([-7e9,7e9])
plt.legend()
plt.show()
```

Activité à réaliser

1. Utiliser des morceaux des codes proposés ci-dessus puis les modifier afin de tracer sur une même figure le mouvement des 8 planètes du système solaire sur une durée de 80 jours.