TP : Tracés

I) Tracés de courbes

Exercice 1 (Épicycloïde)
Une épicycloïde est une courbe plane trajectoire d'un point fixé à un cercle (de rayon r) qui roule sans glisser à l'extérieur un autre cercle (de rayon R) dit directeur. Avec $a = \dfrac{R+r}{r}$, en divisant par r, l'équation de la courbe est

$$\begin{cases} x(t) = a\cos(t) - \cos(at) \\ y(t) = a\sin(t) - \sin(at) \end{cases}$$

Pour $a = 2$ on obtient une cardioïde, pour $a = 3$ une néphroïde, pour $a = 4$, 3 pétales, etc.
Dans tout l'exercice, on s'intéresse au cas $a = 2$. On pourra tester les tracés pour les autres valeurs de a en fin d'exercice.
Toutes les fonctions doivent être testée au fur et à mesure, évidemment.

1) Définir une fonction f qui, à un flottant t associe le couple $(x(t), y(t))$.

2) Définir, à l'aide d'une fonction NumPy ad hoc, un tableau `t` de 100 valeurs régulièrement espacées entre 0 et 2π.

3) Appliquer f à ce tableau, en prédisant ce que va faire Python.
Ici, Python le fait spontanément car toutes les fonctions sont de la bibliothèque NumPy. Sinon, il faut écrire `fv = np.vectorize(f)` puis utiliser `fv`.

4) Comment récupérer $x = (x(t_0), \dots, x(t_N))$? Après avoir récupéré un tableau de valeurs de x et de y, tracer la courbe.
Pour un repère orthonormé : `plt.axis('equal')`.

5) Rappeler la définition de la normale (on pourra se contenter de donner un point, un vecteur).

6) Python ne sait tracer que des segments (il approche la courbe par une ligne brisée). Donc pour tracer une droite, il suffit de donner deux points. Tracer une normale en un point au choix.
Pour la dérivée, vous pouvez soit l'approcher numériquement, soit plus simplement la calculer formellement.

7) Tracer une famille de normale, pour $t \in [0, 2\pi]$. Que remarque-t-on ? Rappeler la définition de la courbe qui apparaît.

```
""" Construction du cadre du tracé : objet "figure", style des axes"""

ma_fig = plt.figure()  #figure vide, sans axes a priori.

ax = plt.gca()  #Axes : construction des axes (cf TP an dernier)
ax.spines['right'].set_color('none')  #l'axe de droite est effacé
ax.spines['top'].set_color('none')    #idem pour l'axe du haut
ax.xaxis.set_ticks_position('bottom')  #graduations : axe du bas uniquement
ax.yaxis.set_ticks_position('left')    #idem vertical
ax.spines['bottom'].set_position(('data', 0))  #l'axe du bas est placé en 0
ax.spines['left'].set_position(('data', 0))    #idem vertical

plt.axis('equal')       #repère orthonormé
plt.xlim(xmin, xmax)    #limites de la boite x
plt.ylim(ymin, ymax)    #limites de la boite y
```

Exercice 2 (Hypocycloïde)
Une hypocycloïde est une courbe plane trajectoire d'un point fixé à un cercle (de rayon r) qui roule sans glisser à l'intérieur d'un autre cercle (de rayon R) dit directeur. Avec $a = \dfrac{R-r}{r}$, en divisant par r, l'équation de la courbe est

$$\begin{cases} x = a\cos(t) + \cos(at) \\ y = a\sin(t) - \sin(at) \end{cases}$$

Pour $a = 2$ on obtient une deltoïde, pour $a = 3$ une astroïde, etc..
Adapter les fonctions de l'exercice précédent pour obtenir quelques tracés de courbes, ainsi que les développées comme enveloppe des normales.

Exercice 3 (Exercices 0, banque PT)
Exercice 11.

II) Arbre récursif

Exercice 4
Coder l'arbre récursif du premier TP. Pensez à la condition de sortie.
On tracera un segment $[A, B]$ à l'aide de la commande `plt.plot([xA, xB], [yA, yB], color='red')`.

III) Des sapins

Vous pouvez faires des arbres (de différentes tailles), placés aléatoirement, plus ou moins compliqués, avec des boules et des guirlandes ou pas, de la couleur ou pas, des flocons de neige (récursifs), etc... Libre à vous d'imaginer ce que vous voulez, suivant ce que vous vous sentez capable de faire.
Plan de travail :

1) Faire une liste des types de dessins que vous voulez faire : une fonction pour chacun. Spécifier les paramètres des fonctions (laissez un maximum de variables).

2) Faites des dessins tout simple pour commencer : vous ferez plus complexe ensuite. Au besoin, pour faire plus complexe, vous ferez de nouvelles fonctions. Toujours décomposer un problème complexe en sous-problèmes simples.

3) Gardez à l'esprit les principes généraux de la programmation : noms de variables et de fonctions explicites (mais pas trop longs), commentaires en début de fonction pour expliquer ce qu'elle fait.

4) Testez toujours vos fonction une par une.

5) Utilisez les raccourcis clavier. Vraiment.

6) N'hésitez pas à utiliser la documentation en ligne. Par exemple via google, avec des mots clés en anglais (quitte à passer par google traduction avant) et en rajoutant au besoin `docs.python.org` (documentation officielle) ou `stackoverflow` (forum de discussion) pour tomber sur des documents fiables. Partez de l'exemple donné, et adaptez le.

Pour rechercher dans une page : Ctrl-F (ou menu Edit -> rechercher).

Outils :

1) Pour effacer les axes : `plt.axis('off')`.

2) Pour un motif plein : `plt.fill(x, y, color='chartreuse')`