TP 4 - Visualisation d'objets mathématiques

En Python, les fonctionnalités graphiques sont fournies par le module `matplotlib`. Pour l'utiliser, on le charge au début du programme : `import matplotlib.pyplot as plt`. On aura aussi besoin du module `numpy` (`import numpy as np`).

Exercice 1. *Visualiser une suite*

En Python, on peut visualiser une suite à l'aide de la commande `plot`, cf. l'aide mémoire.

1. Visualiser la suite Fibonacci $u_5 \ldots, u_{25}$.

   Attention: La fenêtre graphique peut se cacher derrière celle de pyzo.

2. Visualiser la même suite en utilisant l'échelle logarithmique sur l'axe des ordonnées (la commande `plt.yscale('log')`). Que peut-on dire de la croissance de cette suite ?

3. Donner un équivalent simple (approché) de la suite.

Exercice 2. *Courbe représentative d'une fonction*

On rappelle que la courbe représentative d'une fonction f est l'ensemble de points $(x, f(x))$ dans le plan muni d'un repère orthogonal. En Python on peut visualiser une fonction à l'aide de deux *tableaux* `x, y` comme dans l'exemple suivant (à exécuter) :

```
x = np.linspace(-2, 2, 400)
y = x**2
plt.plot(x, y)
plt.plot(x, 4 - x**2)
plt.axis("equal")      # orthonormalisation du repere
```

La commande `linspace` (du module `numpy`) crée un *tableau* de points équidistants entre deux bornes (ci-dessus, on crée 400 points entre -2 et 2). Techniquement, un tableau (ang. *array*) représente un vecteur ou une matrice. Il s'agit d'un type spécial, apporté par le module `numpy`) et différent d'une liste. Son comportement est spécifique : par exemple, si `x` est un tableau, l'expression simple `y = x**2` applique la fonction puissance à tout élément de `x`. Par conséquent, `y` est le tableau de carrés des valeurs dans `x`, ce qui convient pour le graphique.

La commande `plt.plot` construit la courbe définie par le couple `(x,y)` et l'affiche dans une fenêtre (externe à Pyzo). Plusieurs commandes `plot` consécutives superposent des éléments graphiques dans un même cadre. Si on veut séparer deux graphiques, la commande `plt.figure()` crée un nouveau cadre. L'orthonormalisation `plt.axis("equal")` est optionnelle : certains graphiques sont meilleures si les échelles ne sont pas égales.

Visualiser, sur deux graphiques différents, les deux fonctions suivantes (on rappelle que les fonctions usuelles se trouvent dans le module `numpy`) :

$$y = \frac{\sin(x)}{x}, \quad x \in [-10, 10]; \qquad\qquad y = \sin(x^2), \quad x \in [0, 20].$$

Exercice 3. *Fonction(s) puissance*

Soit $f_t : x \mapsto x^t$, $x \in [0, 1]$ où t est un paramètre fixé, $t > 0$.

1. Visualiser, dans un même graphique, une suite de courbes représentatives de fonctions f_t, où t varie de 0.3 à 2 par un pas de 0.2. On pourra faire une boucle à l'aide de la commande `np.arange(0.3, 2, 0.2)`; celle dernière produit une suite arithmétique de flottants (`range` fonctionne seulement avec des entiers).

2. En Python, on peut faire la légende d'un graphique en utilisant le paramètre `label` et la commande `legende`, par exemple:
   ```
   x = np.linspace(0, 1, 200)
   plt.plot(x, x**2, label="carre")
   plt.plot(x, x**3, label="cube")
   plt.legend(loc=0)
   ```
   L'option `loc=0` positionne la légende automatiquement.

   Refaire la question précédente en rajoutant cette fois une description qui indique la valeur de paramètre t pour chaque courbe. Rajouter aussi un titre (`plt.title("...")`)

3. Visualiser, sur un même graphique, les courbes représentatives de f_t sur $[0, 1]$ pour t variant de 0.4 à 3 par un pas de 0.01 (cette fois sans légende).

4. En Python, on peut contrôler la couleur d'un graphe à l'aide d'un *jeu de couleurs* à paramètre. Par exemple :
   ```
   plt.plot(x, y, color=plt.cm.hsv(c))
   ```
   où `c` a une valeur entre 1 et 255.

   Visualiser le suite de courbes représentatives de f_t pour t variant entre 0.4 et 3 avec un pas de 0.01, en utilisant pour la k-ième courbe la couleur numéro k.

   On pourra essayer des jeux de couleur différents : `cm.hsv`, `cm.hot`, `cm.jet`, `cm.winter`, `cm.autumn` (le prefixe `cm` vient de l'anglais *colormap*).

Exercice 4. *Tangente à une courbe représentative*

Soit f une fonction à valeurs réelles. On rappelle que l'équation de la tangente au point d'abscisse x_0 est

$$y = f'(x_0)(x - x_0) + f(x_0).$$

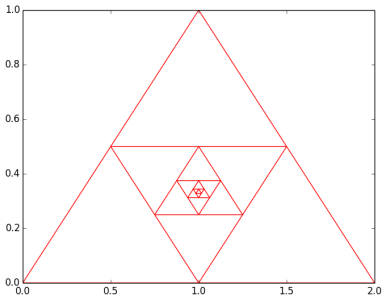
Visualiser sur un même graphique la courbe représentative de $f(x) = \dfrac{1}{x}$, $x > 0$, et la tangente en $x_0 = 2$ (on pourra calculer la dérivée sur papier). Penser à orthonormaliser les axes.

Exercice 5. *Courbes paramétriques*

On rappelle qu'un point sur le cercle unité à pour coordonnées $x = \cos(t)$, $y = \sin(t)$ avec certain $t \in [0, 2\pi[$. Visualiser, sur un même graphique, le cercle unité et un triangle équilatéral inscrit (pour le triangle, voir le cours ou l'exemple "une suite de points dans le plan" dans la section Graphisme de l'aide mémoire).

Exercice 6. *Triangles imbriqués*

Reproduire, *sans récursivité*, le dessin suivant :



Indication : on peut représenter un point par une liste, par exemple `p = [0, 0]`. On peut représenter un triangle par trois points `p1, p2, p3`. Chaque triangle consécutif a ses sommets au milieu des côtés de son "parent".

Exercice 7. *Diagramme de Feigenbaum (extrait du CC 2016)*

Soit a une constante dans $]0, 4]$ et $x_0 \in [0, 1]$. Posons $x_{n+1} = ax_n(1 - x_n)$ pour $n \geq 0$.

1. Etant donnés $a = 2.1$ et $x_0 = 0.2$ (de type `float`), calculer x_{200} puis une liste `L` de 10 éléments $[x_{201}, x_{202}, \ldots, x_{210}]$.

   Afficher cette liste sous forme numérique.

   Que peut-on constater? Votre programme affichera une courte réponse.

2. Refaire la question 1 avec $a = 3.1$ et $x_0 = 0.2$ puis visualiser la liste obtenue ($[x_{201}, x_{202}, \ldots, x_{210}]$) sous forme graphique. Que peut-on observer ? Votre programme affichera une courte réponse.

3. Soit maintenant $a = 3.46$ et $x_0 = 0.2$. Visualiser l'ensemble des points $X_i(a, x_i) \in \mathbb{R}^2$ pour $i = 201, 202, \ldots, 210$ sans les relier (cf. l'aide mémoire). Combien de points peut-on y voir ?

4. Pour $x_0 = 0.2$ et toute valeur de a variant entre 2.5 et 4.0 par pas de 0.01, visualiser, sur *un même graphique* et en couleur rouge, l'ensemble des points non liés $X_i(a, x_i) \in \mathbb{R}^2$, $i = 200, 201, ..., 300$.