Python - Cours 4

Représentations graphiques de fonctions et de suites

L'objectif de ce cours est de représenter graphiquement des fonctions et des suites sur Python. La bibliothèque à importer est `matplotlib.pyplot` que l'on importera ainsi :

```
import matplotlib.pyplot as plt
```

Quand on souhaite représenter graphiquement une fonction ou une suite à la main, il nous faut un tableau de valeurs ; autrement dit, les valeurs de $f(x)$ (ou u_n) pour un certain nombre de valeurs de x (ou de n). Commençons déjà par cela !

I Opérations sur les tableaux de valeurs...

La fonction `array`, disponible dans le bibliothèque `numpy`, permet de **transformer une liste en un tableau** sur lequel les opérations seront possibles...

En gros...
`list` est une liste d'objets mathématiques... alors que `numpy.array` est une liste de nombres.

Saisir les lignes suivantes dans la console :

```
>>> L=[1,2,3]
>>> type(L)
>>> L+L
>>> import numpy as np
>>> T=np.array(L)
>>> type(T)
>>> T+T
```

☞ Rappel...
Si L1 et L2 sont deux listes, alors L1+L2 est

Une fois une liste ainsi convertie en tableau, on peut effectuer un certain nombre d'opérations dont voici les résultats (`L1` est `array([a0,a1,...,an])` et `L2` est `array([b0,b1,...,bn])`).

Syntaxe Python	Résultat
`L1+L2`	
`L1+k`	
`k*L1`	
`L1*L2`	
`L1/L2`	
`L1**k`	
`L1**L2`	
`k**L1`	
`f(L1)`	

✘ Attention !
Pour effectuer des opérations entre L1 et L2, elles doivent faire la même taille !

Petite remarque
Où f désigne une fonction existante de `numpy`, ou une fonction créée.

Exécuter les lignes suivantes et préciser ce que contient `y` :

Python 1 – Opérations

```
1 import numpy as np
2
3 x=np.array([-2+k*0.5 for k in range(0,9)])
4 y=x**2-np.exp(x)
```

Commentaires :

II Représentations graphiques de fonctions

Pour commencer, il parait insensé de devoir saisir à la main la liste des abscisses, surtout si l'on souhaite un graphique précis, et donc une liste relativement grande...
On retiendra donc les deux commandes suivantes :

- pour $a, b \in \mathbb{R}$ et $n \in \mathbb{N}$, la commande `np.linspace(a,b,n)` crée un **tableau de n valeurs équiréparties de a à b inclus**.
- pour $a, b, p \in \mathbb{R}$, la commande `np.arange(a,b,p)` crée un **tableau de valeurs de a inclus à b exclu avec un pas de p**.

Voici la structure ainsi que la syntaxe pour obtenir la courbe d'une fonction :

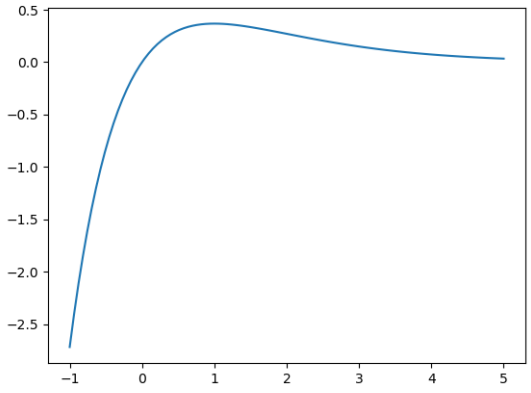
```
x = liste des abscisses
y = liste des ordonnées
plt.plot(x,y)
plt.show()
```

☞ Pour info...
Les listes des abscisses et des ordonnées peuvent être soit du type `numpy.array`, soit du type `list`...

Exemple :

Python 2 – Courbe d'une fonction

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3
4 x=np.linspace(-1,5,100)
5 y=x*np.exp(-x)
6 plt.plot(x,y)
7 plt.show()
```



Petite remarque
Il est également possible de prédéfinir la fonction $x \longmapsto xe^{-x}$ dans une fonction Python, puis de l'utiliser dans la ligne 5...

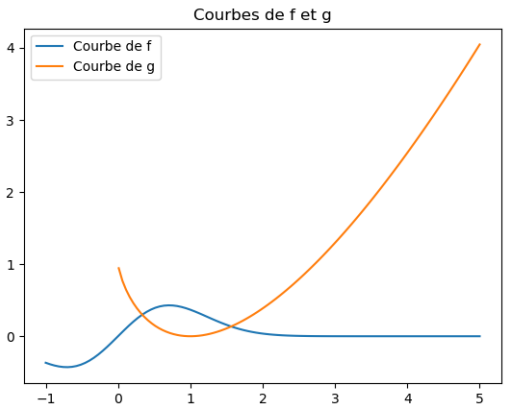
Les commandes qui suivent ne sont pas exigibles, mais elles peuvent parfois servir :

- `plt.grid()` : fait apparaître une grille sur le fond du repère
- `plt.axis('equal')` : rend le repère orthonormé
- `plt.axis([a,b,c,d])` : restreint le repère entre les abscisses a et b et les ordonnées c et d
- `plt.plot(x,y,label="nom de la courbe")`
- `plt.plot(x,y,'couleur')`, où `couleur` désigne la couleur voulue (ou son initiale)
- `plt.legend()` : affiche la légende

Voici comment représenter sur un même graphique les courbes des fonctions $f : x \longmapsto xe^{-x^2}$ et $g : x \longmapsto x\ln(x) - x + 1$:

Python 3 – Courbe de deux fonctions

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3
4 x=np.linspace(-1,5,100)
5 y=x*np.exp(-x**2)
6 plt.plot(x,y, label="Courbe de f")
7
8 x=np.linspace(0.01,5,100)
9 y=x*np.log(x)-x+1
10 plt.plot(x,y, label="Courbe de g")
11
12 plt.title("Courbes de f et g")
13 plt.legend()
14 plt.show()
```



III Représentations graphiques de suites

C'est sensiblement la même chose, à la différence majeure près que les abscisses sont des entiers naturels ; et que l'on représentera les points par des symboles + ou • par exemple.

- `plt.plot(x,y,'+')` : marque les points avec des +
- `plt.plot(x,y,'o')` : marque les points avec des •

Pour des suites définies explicitement, on procède comme pour les fonctions... Pour les suites récurrentes, voici un exemple :

On souhaite représenter les termes de la suite (u_n) définie par : $\begin{cases} u_0 = 1 \\ \forall n \in \mathbb{N},\ u_{n+1} = \sqrt{1+u_n} \end{cases}$.

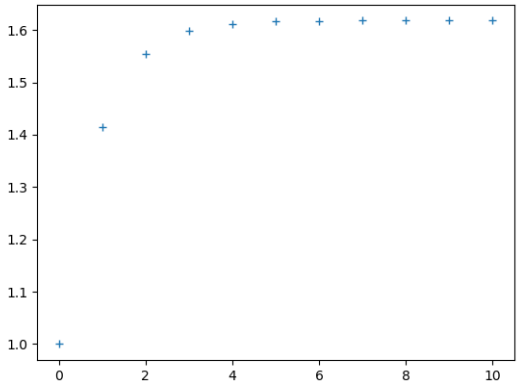
On peut commencer par créer une fonction permettant les calculs des termes de u_n... puis créer une liste d'abscisses et une liste d'ordonnées :

Python 4 – Suite

```
import numpy as np
import matplotlib.pyplot as plt

def u(n):
    U=1
    for k in range(1,n+1):
        U=np.sqrt(1+U)
    return U

x=range(0,11)
y=[u(n) for n in range(0,11)]
plt.plot(x,y,'+')
plt.show()
```



Inconvénient

Cette méthode fonctionne, mais

On pourrait également procéder avec les deux autres méthodes ci-dessous :

Python 5 – On place un point après l'autre

```
import numpy as np
import matplotlib.pyplot as plt

n=0
u=1
plt.plot(n,u,'b+')
for k in range(1,11):
    u=np.sqrt(1+u)
    plt.plot(k,u,'b+')

plt.show()
```

Petite remarque

On précise 'b+' pour que tous les points soient représentés avec le même style (ici des croix bleues), sinon, ils seraient tous avec un style différent.

Python 6 – On crée une liste

```
import numpy as np
import matplotlib.pyplot as plt

u=1
U=[u]
for k in range(1,11):
    u=np.sqrt(1+u)
    U.append(u)

plt.plot(range(0,11),U,'+')
plt.show()
```

●●○○ Exercice 1 - Représentations graphiques de fonctions et de suites

1. Sur un même graphique, tracer les courbes des fonctions $f : x \longmapsto \dfrac{e^x + e^{-x}}{2}$ et $g : x \longmapsto \dfrac{e^x - e^{-x}}{2}$ sur la fenêtre $[-3;3] \times [-4;4]$. Ajouter un titre et une légende.
2. Représenter les premiers termes de la suite $(u_n)_{n\in\mathbb{N}^*}$ définie par : $\forall n \in \mathbb{N}^*,\ u_n = \dfrac{n!}{n^n}$.
3. Représenter les premiers termes de la suite (u_n) définie par : $\begin{cases} u_0 = 0 \\ \forall n \in \mathbb{N},\ u_{n+1} = e^{-u_n} \end{cases}$.
4. Représenter les premiers termes de la suite (u_n) définie par : $\begin{cases} u_0 = 0,\ \ u_1 = 1 \\ \forall n \in \mathbb{N},\ u_{n+2} = u_{n+1} + 2u_n \end{cases}$.
5. Représenter les premiers termes des suites (u_n) et (v_n) définies par $u_0 = 0$, $v_0 = 1$ et :
$$\forall n \in \mathbb{N},\ \begin{cases} u_{n+1} = 0,2u_n + 0,7v_n \\ v_{n+1} = 0,8u_n + 0,3v_n \end{cases}$$
6. Représenter sur un même graphique la courbe de la fonction ln ainsi que les 30 premiers termes de la somme partielle de la série harmonique.

Petite remarque
Si besoin, créer une fonction permettant le calcul de $n!$...

Petite remarque
Ne pas hésiter à remettre le nez dans le TP1...

●●○○ Exercice 2 - Courbes des fonctions puissances

Sur un même graphique, tracer les courbes des fonctions $f_n : x \longmapsto x^n$, pour $n \in \{-2, -1, 0.5, 0.7, 1, 1.5, 2, 3\}$.

●●●○ Exercice 3 - Petit mélange...

Pour $n \in \mathbb{N}^*$, on considère la fonction $f_n : x \longmapsto x^n + x - 1$. *On a vu, dans un précédent exercice, que pour tout $n \in \mathbb{N}^*$, l'équation $f_n(x) = 0$ possède une unique solution dans $[0;1]$, que l'on notera α_n.*

1. Écrire un programme `Python` qui demande une valeur de $n \in \mathbb{N}^*$ à l'utilisateur et affiche un graphique représentant la courbe de f_n, en précisant avec une croix rouge le point de coordonnées $(\alpha_n, 0)$.
2. Écrire maintenant un programme qui demande une valeur de $n \in \mathbb{N}^*$ à l'utilisateur et affiche un graphique représentant toutes les courbes des fonctions f_k, en précisant avec une croix rouge le point de coordonnées $(\alpha_k, 0)$, pour $k \in [\![1;n]\!]$.

●●●○ Exercice 4 - Suites récurrentes...

Écrire un programme `Python` permettant d'obtenir la représentation en "escalier", en "spirale" (ou autre ?) de la suite (u_n) définie par :
$$\begin{cases} u_0 \in \mathbb{R} \\ \forall n \in \mathbb{N},\ u_{n+1} = f(u_n) \end{cases}$$
où f est une fonction donnée.
On testera le programme dans les cas suivants :

1. $u_0 = 0$ et : $\forall n \in \mathbb{N},\ u_{n+1} = \sqrt{1 + u_n}$
2. $u_0 = 0$ et : $\forall n \in \mathbb{N},\ u_{n+1} = e^{-u_n}$
3. d'autres cas au choix !