Travaux Pratiques 7 : Représentations graphiques

A retenir : python construit une courbe point par point et interpole entre ces points (c'est-à-dire relie ces points). Pour représenter une courbe dans python, il faut donc préciser un vecteur d'abscisses `x`*, le vecteur des ordonnées correspondantes* `y`*, puis utiliser les syntaxes*

`plt.plot(x,y)` *# crée le graphique*
`plt.show()` *# affiche le graphique (inutile avec certains éditeurs)*

Exercice 1:

Comprendre ce que fait le programme suivant :

```
import numpy as np
import matplotlib.pyplot as plt
x=np.array([1,3,4,6])
y=np.array([2,3,5,1])
plt.plot(x,y)
plt.show()
```

Changer alors la ligne 5 par : `plt.plot(x,y,'+')`. Commenter.

Exercice 2: *Première représentation graphique*

1. Dessiner sur votre feuille, l'allure de l'exponentielle sur $[0, 1]$.
2. Taper les instructions suivantes dans un script, puis l'exécuter. Bien comprendre toutes les lignes !

   ```
   import numpy as np
   import matplotlib.pyplot as plt
   x=np.linspace(0,1,10)
   y=np.exp(x)
   plt.plot(x,y)
   plt.show()
   ```

3. Comparer votre graphique avec celui de python. Remarquer que le repère de ce dernier n'est pas orthonormé (ce qui "fausse" l'allure), et que le point d'intersection des axes n'est pas l'origine.
   En effet, les ordonnées sont comprises entre 1 et e, donc l'axe des y part de 1.
4. Rajouter dans le script les commandes suivantes, puis constater :
   `plt.axis("equal")` `plt.xlim(0,3)`
   `plt.ylim(0,3)`
5. Remplacer la ligne `x=np.linspace(0,1,10)` par la ligne `x=np.arange(0,1.1,0.1)`.
   Le vecteur x est-il changé ? Le graphique est-il changé ? Quelle syntaxe est la plus simple ?
6. Taper les instructions nécessaires pour faire afficher la droite y=x+1 sur [0,1].
   Réaliser que tous les graphiques doivent être effectués avant de marquer la commande `plt.show()`. On ne peut pas rajouter un graphique plus tard. D'où l'intérêt d'écrire les commandes dans l'éditeur pour pouvoir les modifier puis les exécuter toutes d'un coup.

A retenir : il existe plusieurs commandes pour créer le vecteur des abscisses `x`*. Cette année nous utiliserons essentiellement* `linspace()` *(plus simple pour un vecteur de réels) et* `arange()` *(plus simple pour un vecteur d'entiers).*

Exercice 3:

Tracer la fonction inverse sur $]0, 2]$ puis sur $[-2, 2] - \{0\}$.

Exercice 4: *Vérification d'inégalités*

Le but est de vérifier graphiquement l'encadrement suivant : $\forall x \in \mathbb{R}_+,\ x - \frac{1}{2}x^2 \leq \ln(1 + x) \leq x$.

1. Définir *le* vecteur `x` correspondant aux abscisses choisies, puis représenter les courbes nécessaires à l'étude.
2. Pour plus de lisibilité, vous pouvez rajouter une légende via l'argument `label=" -----"` dans `plt.plot`, puis rajouter la commande `plt.legend()` avant la commande `plt.show()`.

Exercice 5: *Résolution d'une équation*

Le but est de montrer graphiquement que l'équation $\cos(x) = x$ admet une unique solution dans $\mathbb{R}$, et d'avoir une valeur approchée de celle-ci.

1. Définir le vecteur `x` correspondant aux abscisses choisies et justifier le choix de cette fenêtre.
2. Représenter les courbes nécessaires à l'étude de l'équation $\cos(x) = x$.
3. Prendre des initiatives pour obtenir une valeur approchée à 0.05 près de la solution.

Exercice 6: *Méthode des rectangles*

1. Représenter graphiquement la fonction $f : x \mapsto \frac{x}{e^x+1}$ sur [0,1] à l'aide de python.
   Interpréter $\int_0^1 f(x)dx$.
2. Le but de cette question est de déterminer une valeur approchée de $\int_0^1 f(x)dx$, à l'aide de la méthode des rectangles.
   (a) Compléter le théorème suivant : Soit f une fonction continue sur $[0,1]$.
   Alors $S_n(f) = \frac{1}{n}\sum_{k=0}^{n-1}$ $\underset{n\to+\infty}{\longrightarrow} \int_0^1 f(t)dt$.
   (b) Créer une fonction d'en-tête `def S(n):` qui au paramètre d'entrée n renvoie la valeur de $S_n(f)$.
   (c) Exécuter la fonction avec plusieurs n et conclure.

Exercice 7: *Représentation graphique d'une suite*
Soit la suite u définie par $u_0 = 1$ et $u_{n+1} = \sqrt{2+u_n}$.

1. Ecrire une fonction `python` d'en-tête `def suite(n):` qui à un entier n renvoie le réel u_n.
2. Modifier la fonction précédente afin qu'elle donne en sortie le vecteur $[u_0, ..., u_n]$.
3. Compléter le script suivant afin d'obtenir la représentation graphique de $(u_0, u_1, ..., u_{20})$. On suppose la fonction python `suite()` chargée, et les bibliothèques importées.

```
x=np.arange(
y=
plt.plot(x,y,'x')
plt.show()
```

4. Exécuter alors le script : qu'en déduit-on sur la suite u ?
5. Que se passe-t-il si $u_0 = 2$? et si $u_0 = 3$?
6. Aurait-on pu définir facilement le vecteur des abscisses `x` via la syntaxe `np.linspace()` ?

Exercice 8: `inspiré d'ecricome E 2015`
Soit la suite u définie par $u_0 = 1$ et $u_{n+1} = 1 - e^{-u_n}$.

1. Compléter le programme afin d'obtenir la représentation graphique des 100 premiers termes de cette suite.

```
U = np.zeros(100)
.............
for .........:
    .............
N=np.arange(100)
plt.plot(N,U,"+")
plt.show()
```

2. Exécuter alors le programme : que peut-on conjecturer sur la suite ?
3. ** Modifier le programme précédent en remplaçant les 2 dernières lignes par :

```
S = np.cumsum(U)
Y = np.log(N)
plt.plot(N,S,"-.", label="S")
plt.plot(N,Y, label="ln")
plt.legend()
plt.show()
```

   Que représente le vecteur-ligne `S` ?
   Si besoin : taper dans la console `k=np.array([1,3,6,8]), np.cumsum(k)` *pour comprendre l'instruction* `np.cumsum`.
   Quelle conjecture pouvez-vous émettre sur la limite de la somme $\sum_{k=1}^{n} u_k$?