ON 1 - Tracés de graphiques

I. Tracer une courbe

1. La fonction plot

Le module `matplotlib` est chargé de tracer les courbes. Il peut être nécessaire de charger la librairie : `import matplotlib.pyplot as plt`.

L'instruction `plot(x,y)`, où x et y sont des listes ou des tableaux de données, permet de tracer y(x).

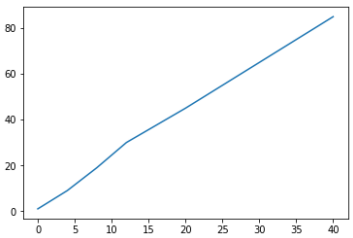
2. Tracé d'une courbe de données

On stocke les abscisses et les ordonnées dans des listes x et y :
`x = [x1,...,xn] et y = [y1...,yn]`

On lance l'instruction `plot(x,y)` : cela trace l'ensemble des points de cordonnées (x_k,y_k) en les reliant entre eux.

```
import matplotlib.pyplot as plt

x=[0,2,4,8,12,20,30,40]
y=[1,5,9,19,30,45,65,85]
plt.plot(x,y)
```



3. Tracé d'une fonction

Par défaut, **`plot`**`()` relie l'ensemble des points donnés, on obtiendra une courbe correcte en travaillant avec un nombre de points suffisants.

On utilise généralement des tableaux (type `array`) pour pouvoir faire des opérations membres à membres telle que des additions ou des multiplications.

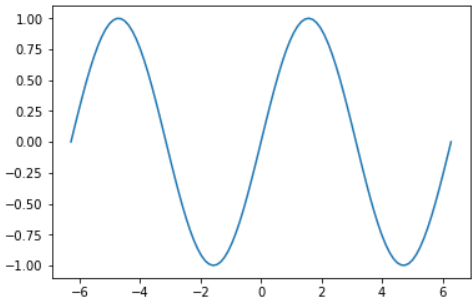
→ **A l'aide de la fonction `linspace` de `numpy` :** On donne, dans l'ordre, les bornes inférieures et supérieures (inclues) et le nombre de valeurs souhaitées.

→ **A l'aide de la fonction `arange` de `numpy` :** On donne, dans l'ordre, la borne inférieure (inclue), la borne supérieures (exclue) et le pas.

→ Pour générer le tableau de valeurs y, il suffit d'écrire l'instruction y = f(x) où f est une fonction prédéfinie (sinus, cosinus…) ou à définir.

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

x=np.linspace(-2*np.pi,2*np.pi,100)
y=np.sin(x)
plt.plot(x,y)
```



II. Fonctions principales

Fonctions	Actions réalisées
`plt.plot(x, y, 'styleDuGraphe', linewidth=1, label = 'y = f(x)')`	Tracer la courbe représentant y en fonction de x avec le style `styleDuGraphe`, l'épaisseur `linewidth`, le nom de la courbe à afficher dans la légende
`plt.xlabel('axe des abscisses ')` `plt.ylabel('axe des ordonnées')`	Ajouter des libellés sur les axes

`plt.axis([xmin,xmax,ymin,ymax])` ou
`plt.xlim(xmin,xmax)`
`plt.ylim(ymin,ymax)`
	Définir des valeurs minimales et maximales pour les abscisses (`xmin` et `xmax`) et les ordonnées (`ymin` et `ymax`)

`plt.title('titre du graphe')`	Ajouter un titre au graphique
`plt.grid()`	Ajouter une grille au graphique
`plt.text(valeur x, valeur y, 'texte')`	Ajouter du texte dans le graphe à la position souhaitée
`plt.legend()`	Ajouter une légende avec le nom des courbes
`plt.show()`	Afficher le graphe
`plt.clf()`	Supprime les graphes précédents

III. Pour améliorer les tracés

De nombreuses fonctions de `pyplot` (`plt.plot()`, `plt.grid()`, `plt.xlabel()`, `plt.ylabel()`, `plt.title()`,...) peuvent utiliser des mots clé comme `color = ''` permettant l'usage de toute la palette des couleurs, `linewidth = ''` permettant de choisir l'épaisseur du tracé (abrégé `lw`) ou encore `marker='...'` permettant d'ajouter des marqueurs pour chaque point du tracé etc...

Exemple : `plt.plot(x, y, 'styleDuGraphe')` *trace la courbe y(x) où* `styleDuGraphe` *est une chaîne de caractères qui regroupe la couleur de la courbe, le marqueur de point et le style de liaison entre les points.*

Ci-dessous des listes d'options possibles et le symbole correspondant

Caractère	Marqueur de point
.	Point
o	rond
v	triangle pointe en bas
^	triangle pointe en haut
<	triangle pointe à gauche
>	triangle pointe à droite
1	croix à 3 branches vers le bas
2	croix à 3 branches vers le haut
3	croix à 3 branches vers la gauche
4	croix à 3 branches vers la droite
s	carré
p	pentagone
*	étoile
h	hexagone
+	plus
P	plus plein
x	croix
X	croix pleine
d	carreau

Caractère	Style de ligne
-	ligne continue
--	tirets
:	ligne en pointillé
-.	tirets points

Caractère	Couleur
b	bleu
g	vert
r	rouge
c	cyan
m	magenta
y	jaune
k	noir
w	blanc

Exemple : `plt.plot(x, y, 'r+:')` *trace un graphe y(x) dont les points sont rouges, en forme de + et reliés par des lignes en pointillé.*

IV. Tracé de plusieurs graphes

1. Superposer des courbes

Il suffit de faire une succession d'instructions `plot`.

2. Affichage de plusieurs graphes dans différentes fenêtres

L'instruction `plt.figure(2)` ouvre une seconde fenêtre et les instructions qui suivent s'adresseront à cette seconde figure.

3. Affichage de plusieurs graphes dans une même fenêtre

L'instruction `subplot()` permet de juxtaposer différents graphiques. On donne successivement le **nombre de lignes**, le **nombre de colonnes**, le **numéro de la figure** (la numérotation se fait de gauche à droite et de haut en bas) et les éventuelles options (fond coloré par exemple).

Exemple : *Oscillateur harmonique non amorti ou en régime pseudopériodique avec une grille de 1 colonne et 2 lignes avec la couleur de fond du second graphe qui sera cyan*

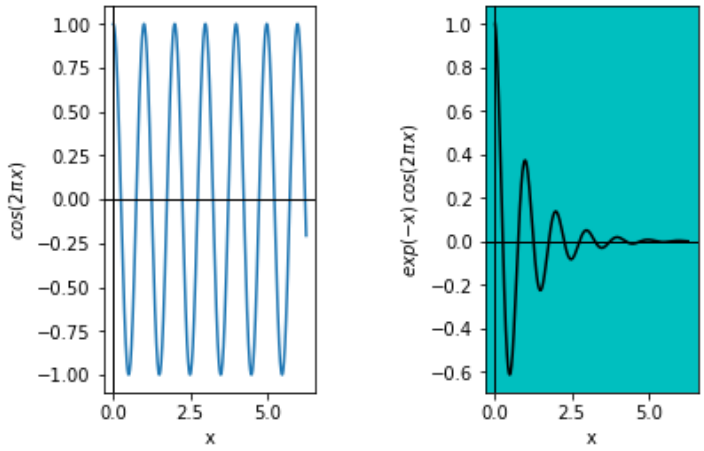
```
import matplotlib.pyplot as plt
import numpy as np

x=np.linspace(0,2*np.pi,1000)
y=np.exp(-x)*np.cos(2*np.pi*x)

plt.subplot(1,2,1)
plt.plot(x, np.cos(2*np.pi*x))
plt.ylabel('$cos(2\pi\,x)$')
plt.xlabel('x')
plt.axhline(lw=1,color='k')
plt.axvline(lw=1,color='k')

plt.subplot(1,2,2, facecolor='c')
plt.plot(x,y,'k')
plt.ylabel('$exp(-x)\,cos(2\pi\,x)$')
plt.xlabel('x')
plt.axhline(lw=1,color='k')
plt.axvline(lw=1,color='k')

# pour espacer les graphes
plt.subplots_adjust(wspace = 0.8)
```


1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
cos(2πx)
0.0
2.5
5.0
x
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
exp(−x) cos(2πx)
0.0
2.5
5.0
x