

## TP 05 – REPRÉSENTATIONS GRAPHIQUES

L'objectif de ce TP est d'apprendre à **représenter** graphiquement des **fonctions** et des **suites** à l'aide de Python. La bibliothèque nécessaire est `matplotlib.pyplot` que l'on importera de la manière suivante :

Entrée [1]: 

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

Quand on souhaite représenter graphiquement une fonction (ou une suite), il nous faut deux *tableaux* de valeurs.

- Un tableau d'*abscisses* contenant un certain nombre de valeurs de  $x$  (ou de  $n$ )

`abscisses=[x1, x2, ..., xp]`      (ou `abscisses=[n1, n2, ..., np]`)

- Un tableau d'*ordonnées* contenant les valeurs correspondantes de  $f(x)$  (ou de  $u_n$ )

`ordonnees=[f(x1), f(x2), ..., f(xp)]`      (ou `ordonnees=[un1, un2, ..., unp]`)

Alors, grâce aux commandes suivantes, Python affiche un graphe sur lequel il a placé dans le plan tous les points  $(x_i, f(x_i))$  pour  $i=1, \dots, p$  qu'il relie (par défaut) après entre eux. Ainsi, plus le nombre de points est important, plus la courbe sera précise.

Entrée [2]: 

```
plt.plot(abscisses, ordonnees)
plt.show()
```

Pour les suites, on devra forcer Python à ne pas relier les points entre eux (une suite n'est définie que sur des valeurs entières) grâce à l'ajout d'une option dans la première commande comme suit.

Entrée [3]: 

```
plt.plot(abscisses, ordonnees, '+')
plt.show()
```

Les commandes qui suivent peuvent être rajouter pour améliorer l'esthétisme du graphe. Elles ne sont cependant pas au programme.

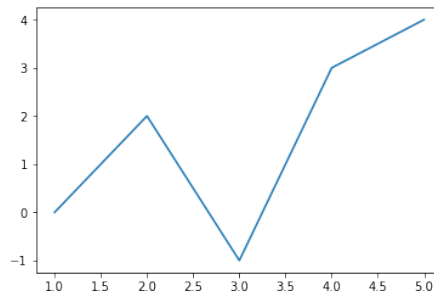
Commande	Effet
<code>plt.grid()</code>	Fait apparaître une grille sur le fond du repère
<code>plt.axis([a,b,c,d])</code>	restreint le repère entre les abscisses $a$ et $b$ et les ordonnées $c$ et $d$
<code>plt.title("nom")</code>	Permet de donner un nom au graphique
<code>plt.plot(abscisses, ordonnees, label='nom')</code>	Donne un nom à la courbe (dans la légende)
<code>plt.legend()</code>	Affiche la légende

## I Relier les points

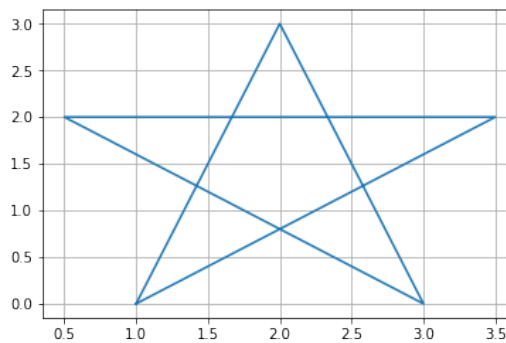
**Exercice 1** Quelle courbe Python affiche-t-il à la suite de ces instructions ?

Entrée [4]:

```
abscisses = [1, 2, 3, 4, 5]
ordonnées = [0, 2, -1, 3, 4]
plt.plot(abscisses, ordonnées)
plt.show()
```

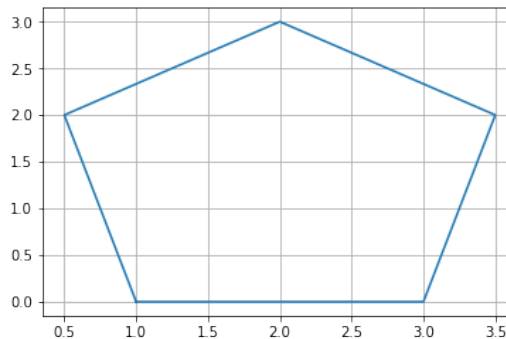


**Exercice 2** 1. Réaliser cette figure grâce à Python (sans la grille).



```
abscisses = [1, 3.5, 0.5, 3, 2, 1]
ordonnées = [0, 2, 2, 0, 3, 0]
plt.plot(abscisses, ordonnées)
plt.show()
```

2. Réaliser cette figure (sans la grille).



```
abscisses = [1, 3, 3.5, 2, 0.5, 1]
ordonnées = [0, 0, 2, 3, 2, 0]
plt.plot(abscisses, ordonnées)
plt.show()
```

## II Opérations sur les tableaux de valeurs

De manière générale, les listes *abscisses* et *ordonnées* contiennent beaucoup de valeurs. Il faut donc apprendre à les définir et les manipuler de manière efficace (sans rentrer les valeurs une par une "à la main"). Plutôt que de travailler avec des listes (comme précédemment), il est plus judicieux de travailler avec *tableaux* sur lesquels plus d'opérations sont possibles. Le module `array` de `numpy` permet de définir des tableaux.

```
Entrée [5]: import numpy as np
            L1 = np.array([1,2,3]) #definition explicite
            L2 = np.array([k for k in range(4,7)]) #definition en comprehension
```

```
Entrée [6]: print(L2)
```

```
Out [6]: [4, 5, 6]
```

Commande	Effet	Résultat
$L1+L2$	Addition terme à terme	<code>array([5,7,9])</code>
$L1+2$	Addition de la constante à chaque terme	<code>array([3,4,5])</code>
$3*L1$	Multiplication par la constante de chaque terme	<code>array([3,6,9])</code>
$L1*L2$	Multiplication terme à terme	<code>array([4,10,18])</code>
$L1/L2$	Division terme à terme	<code>array([0.25,0.4,0.5])</code>
$L1**2$	Puissance terme à terme	<code>array([1,4,9])</code>
$f(L1)$	Application de la fonction $f$ à chacun des termes	

- Pour effectuer les opérations entre  $L1$  et  $L2$ , les deux listes doivent être de la même taille.
- Dans la dernière ligne du tableau, la fonction  $f$  peut désigner une fonction déjà existante de `numpy` (comme `np.exp`, `np.sqrt`,...) ou une fonction créée (à l'aide de l'environnement `def`).

**Exercice 3** Donner l'affichage du programme suivant.

```
Entrée [7]: def f(x):
            return(x+1)

            x = np.array( [-2+k*0.5 for k in range(0,9)])
            print(x)
            y = 2*x -f(x)
            print(y)
```

```
Out [7]: [-2.  -1.5 -1.  -0.5  0.   0.5  1.   1.5  2. ]
        [-3.  -2.5 -2.  -1.5 -1.  -0.5  0.   0.5  1. ]
```

### III Représentation graphique d'une fonction

Pour représenter une fonction, il nous faut deux tableaux de valeurs : un contenant les abscisses souhaitées et l'autre l'image de ces abscisses par l'application qu'on souhaite représenter.

- Pour avoir un graphique précis, la *liste des abscisses* doit être grande. On ne rentre donc pas les valeurs à la main mais on peut utiliser les deux commandes suivantes.
  - La commande `np.linspace(a,b,n)` qui crée un tableau de  $n$  valeurs réparties de manière uniforme entre  $a$  et  $b$  (inclu).

Entrée [8]: `np.linspace(0,1,3)`

Out [8]: `array([0. , 0.5, 1. ])`

- La commande `np.arange(a,b,p)` qui crée un tableau de valeurs de  $a$  à  $b$  (exclu) avec un pas de  $p$ .

Entrée [9]: `np.arange(0,1,0.1)`

Out [9]: `array([0. , 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9])`

- La *liste des ordonnées* est obtenue par manipulation sur les listes (expliquée à la Section 2).

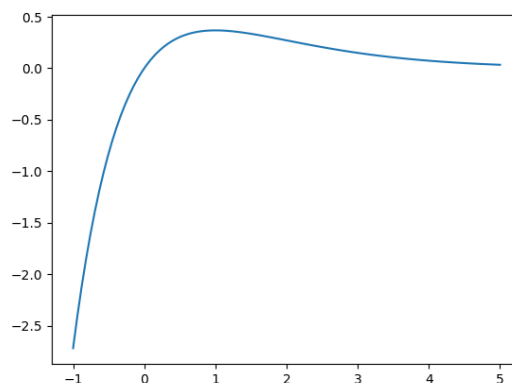
Entrée [10]: `#Représentation de f(x)=x**2 sur [0,1]`  
`abscisses = np.linspace(0,1,1000)`  
`ordonnes = abscisses**2`

Enfin, la structure complète pour obtenir la courbe d'une fonction est la suivante.

```
abscisses = liste des abscisses #obtenue avec linspace ou arange
ordonnees = liste des abscisses #obtenue par operations
plt.plot(x,y)
plt.show()
```

**Exercice 4** Représenter la fonction  $x \mapsto x \exp(-x)$  sur l'intervalle  $[-1, 5]$ .

Entrée [11]: `abscisses = np.linspace(-1,5,100)`  
`ordonnees = abscisses*np.exp(-abscisses)`  
`plt.plot(abscisses,ordonnees)`  
`plt.show()`



## IV Exercices

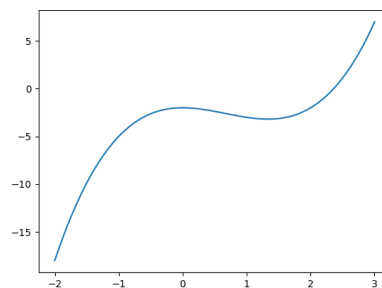
### IV.1 Sur les fonctions

**Exercice 5** Représenter sur la fonction polynomiale suivante

$$f : x \mapsto x^3 + 2x^2 - x - 2$$

sur l'intervalle  $[-2, 3]$ .

```
Entrée [12]: abscisses = np.linspace(-2,3,100)
ordonnees = abscisses**3-2*abscisses**2-2
plt.plot(abscisses,ordonnees)
plt.show()
```



**Exercice 6** Représenter sur le même graphique, les fonctions

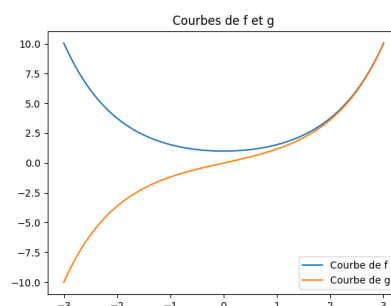
$$f : x \mapsto \frac{e^x + e^{-x}}{2} \quad \text{et} \quad g : x \mapsto \frac{e^x - e^{-x}}{2}$$

sur l'intervalle  $[-3, 3]$ .

```
Entrée [13]: #Courbe de f
abscisses = np.linspace(-3,3,100)
ordonnees = (1/2)*(np.exp(abscisses) + np.exp(-abscisses))
plt.plot(abscisses,ordonnees, label='Courbe de f')

#Courbe de g
abscisses = np.linspace(-3,3,100)
ordonnees = (1/2)*(np.exp(abscisses) - np.exp(-abscisses))
plt.plot(abscisses,ordonnees, label='Courbe de g')

plt.title('Courbes de f et g')
plt.legend()
plt.show()
```



**Exercice 7** Représenter sur le même graphique, les fonctions  $x \mapsto x^n$  pour  $n \in \{-2, -1, 0.5, 1, 1.5\}$  sur l'intervalle  $]0, 5]$ .

Entrée [14]: `abscisses = np.linspace(0.001, 5, 100)`

```
for n in [-2, -1, 0.5, 1, 1.5]:  
    ordonnees = abscisses**n  
    plt.plot(abscisses, ordonnees)  
  
plt.axis([0, 5, 0, 5])  
plt.title('Courbes des puissances')  
plt.show()
```

