



# TP Python 6 : Représentation graphique de fonctions



## Préambule

Dans ce TP, nous allons nous intéresser à la **représentation graphique de fonctions**.

La bibliothèque `matplotlib.pyplot` contient une multitude de fonctions déjà prêtes à l'emploi pour représenter des fonctions en Python.

Ainsi, si l'on souhaite représenter le graphe d'une fonction, il nous faudra donc l'importer, en plus de la librairie `numpy`. Tout comme on abrège `numpy` par `np` pour aller plus vite, il est d'usage d'abrèger `matplotlib.pyplot` par `plt`.

```
Entrée[2]: 1 import numpy as np
           2 import matplotlib.pyplot as plt
```

## I. Un peu de théorie

Considérons  $f$  une fonction définie sur  $\mathbb{R}$

- La courbe représentative (ou graphe) de  $f$  est un ensemble de points du plan de la forme  $M(x, f(x))$ , où  $x \in \mathbb{R}$ .
- Ainsi, nous allons devoir nous munir d'outils en Python pour décrire le graphe de  $f$ . Pour ce faire, on va utiliser la notion de **liste** rapidement abordée au TP précédent.

L'objectif est le suivant :

On va se munir d'une **liste contenant les abscisses**  $x$ , et on va **construire** une **liste contenant les images**  $f(x)$  correspondantes

### I. 1) La fonction `linspace`

La fonction `linspace` est une fonction prête à l'emploi de la librairie `numpy`. Pour l'appeler, on utilisera donc la commande `np.linspace`. Cette fonction permet de **générer une liste de valeurs**. Dans notre cas, on l'utilisera pour *générer la liste des abscisses* que l'on souhaite considérer.

### Exemple :

Exécuter le programme suivant, et intuitiver le fonctionnement de la fonction `linspace`.

Entrée[2]:

```
1 a = np.linspace(0,10,3)
2 print(a)
3 b = np.linspace(0,10,5)
4 print(b)
5 c = np.linspace(0,10,11)
6 print(c, '\n')
7 d = np.linspace(3,7,4)
8 print(d)
```

```
[ 0.  5. 10.]
[ 0.  2.5  5.  7.5 10. ]
[ 0.  1.  2.  3.  4.  5.  6.  7.  8.  9. 10.]

[3.          4.33333333 5.66666667 7.          ]
```

La fonction `linspace` prends **trois variables en entrée**, via la commande `np.linspace(start, stop, num)`, où :

- La variable `start` correspond au **premier terme** de la liste. Il est **inclus** dans la liste.
- La variable `stop` correspond au **dernier terme** de la liste. Il est **inclus** dans la liste.
- La variable `num` correspond au **nombre de termes contenus dans la liste**.

### Exercice 1 :

1. Définir une liste `x` contenant 13 valeurs entre 0 et 5, puis l'afficher.
2. On décrit l'intervalle  $[0, 1]$  en partant de 0 et en avançant d'un **pas de discrétisation** de 0.25 jusqu'à arriver en 1. Définir la liste `y` d'abscisses correspondantes, puis l'afficher.
3. On décrit l'intervalle  $[2, 12]$  en partant de 2 et en avançant d'un **pas de discrétisation** de 1 jusqu'à arriver en 12. Définir la liste `z` d'abscisses correspondantes, puis l'afficher.

Entrée[ ]:

1

Vous l'aurez compris, on utilise `linspace` pour définir un ensemble d'abscisses de la manière suivante : on décrit un intervalle  $[a, b]$  ( $a < b$  deux réels) avec un **pas de discrétisation** fixé  $\delta_x$ .

Le risque est alors de **se tromper dans le choix de la variable `num`**.  
Un bon garde-fou est la commande `len`, qui renvoie la **longueur d'une liste donnée**

**Exemple :**

Exécuter le programme suivant.

Entrée[ ]:

```
1 a = np.linspace(0,1,4)
2 print('La liste a est de longueur ',len(a),'et a = ',a,'\n')
3 b = np.linspace(0,1,5)
4 print('La liste b est de longueur ',len(b),'et a = ',b,'\n')
```

**Remarque :**

Plus le pas de discrétisation (la troisième variable `num` de la fonction `linspace`) est **petit**, plus la liste obtenue est **longue**. On dit que le pas est **fin**.

## 1. 2) Définir une liste d'images

Soit  $f$  une fonction définie sur un intervalle  $[a, b] \subset \mathbb{R}$ . On sait maintenant définir une liste d'abscisses contenues dans cet intervalle  $[a, b]$ .

*Comment dès lors définir la liste des images associées?*

Un exemple est cette fois-ci plus parlant !

### Exercice 2 :

On va se donner une fonction affine  $f : x \mapsto 2x + 1$  définie sur  $\mathbb{R}$ , et on va s'intéresser à sa restriction à l'intervalle  $[0, 10]$ .

1. Définir  $X$  la liste contenant les valeurs obtenues en parcourant l'intervalle  $[0, 10]$  avec un pas de 1.
2. Définir la fonction `func_affine` qui étant donnée un réel  $x$ , renvoie  $2x + 1$ .
3. À l'aide d'une boucle `for` et de la commande `len`, définir une liste  $Y$  qui contient, pour chaque valeur de la liste  $X$ , l'image correspondante par  $f$  de cette valeur.

### Rappel :

- On initialise une liste vide à l'aide de la commande `liste = []`.
- Si  $k$  est un entier, la commande `liste[k]` renvoie la  $k + 1$ ème valeur de la liste (*array starts at 0!*).
- La commande `liste.append(truc)` ajoute à la liste **déjà existante** `liste` la variable `truc`.

Entrée[ ]: 1

## 1. 3) Première utilisation de `matplotlib.pyplot`

On sait désormais se munir d'une liste d'antécédents, et construire la liste des images par une fonction  $f$  correspondantes. Il ne reste plus qu'à **représenter graphiquement les points de coordonnées**  $(x, f(x))$ . Dans le jargon, on dit aussi **plotter** la fonction  $f$ ...

On va en effet utiliser la **fonction** `plot` de la bibliothèque `matplotlib.pyplot`, via la commande `plt.plot(x,y)`, où :

- la variable `x` est la liste des antécédents,
- la variable `y` est la liste des images correspondantes.

### Remarques :

- **Produire l'image** (au sens informatique du terme) du graphe de  $f$  est fait par `plt.plot`, mais **afficher cette image dans une fenêtre graphique** relève d'une autre fonction : `plt.show()` qui ne prends pas d'argument!
- On utilise le raccourci `plt` pour désigner `matplotlib.pyplot`. Sans raccourci, l'appel à la fonction `plot` sera : `matplotlib.pyplot.plot` !

**Exemple :**

On cherche à tracer le graphe de la fonction  $f : x \mapsto 2x + 1$  sur l'intervalle  $[0, 10]$ . Vu que cette fonction est déjà codée par la fonction `func_aff`, pas besoin de la recoder, il suffit de l'appeler! On pourrait argumenter qu'il en va de même pour les listes d'images et d'antécédents, mais puisque les variables `x` et `y` ne cessent d'être redéfinies au cours de ce TP, ne prenons pas de risques.

Entrée[14]:

```
1 # Liste des antécédents
2 X = np.linspace(0,10,11)
3
4 # On construit les images
5 Y = []
6 for k in range(len(X)):
7     Y.append(func_affine(X[k]))
8
9 # On représente graphiquement la fonction affine
10
11 plt.plot(X,Y)
12 plt.show()
```

**Exercice 3 :**

Tracer la fonction  $f : x \mapsto \frac{1}{2}e^x$  sur l'intervalle  $[-1, 1]$ , en choisissant le pas de discrétisation.

**Remarque :**

En Python, si l'on demande à plotter plusieurs fonctions, la **fenêtre graphique** reste la même! Cela peut nous empêcher d'obtenir un seul graphique à la fois. Pour éviter ce problème, on utilise la commande `plt.clf()` qui permet de **réinitialiser la fenêtre graphique**.

Entrée[ ]:

1

## ***1. 4) Pour aller plus loin avec matplotlib.pyplot (HP)***

Tracer le graphe d'une fonction demande d'annoter les axes, et de donner un titre à notre fonction. Tout cela est faisable avec `matplotlib.pyplot`, mais n'est pas exigible au programme :

- En **troisième argument** de `plot`, on peut modifier la couleur du graphique (`r` pour rouge, `b` pour bleu, `g` pour vert...).
- En accolant certains symboles à la couleur, on peut modifier l'aspect des points composant le graphique (`+` pour des croix, `*` pour des astérisques...).
- La fonction `title` permet de donner un titre à son graphique.

- La fonction `legend` permet de donner un titre à un graphique contenant plusieurs représentations graphiques.
- La fonction `xlabel` permet de donner un titre à l'axe des abscisses.
- La fonction `ylabel` permet de donner un titre à l'axe des ordonnées.

**Exemple :**

On trace la courbe représentative de la fonction  $f : x \mapsto -3x + 4$  sur  $[0.5, 2]$ .  
On change la couleur du graphique et donne des titres aux axes et à la figure.

Entrée[ ]:

```

1 def f1(x):
2     y = -3*x+4
3     return y
4
5 X = np.linspace(0.5,2,100)
6 Y = []
7 for k in range(len(X)):
8     Y.append(f1(X[k]))
9
10 plt.clf()
11 plt.plot(X,Y, 'r')
12 plt.xlabel('x')
13 plt.ylabel('y')
14 plt.title('Graphe de f(x)=-3x+4')
15 plt.show()

```

**Exercice 4 :**

Tracer la courbe représentative de la fonction  $f : x \mapsto \cos(x)$  sur  $[-10, 10]$ .  
Mettre la couleur du graphique en vert, changer l'aspect des points pour obtenir des croix et donne des titres aux axes et à la figure.

Entrée[ ]:

1

**Exemple :**

Que fait le programme suivant?

```

Entrée[ ]: 1 X = np.linspace(0,1,100)
           2 Y2 = []
           3 Y3 = []
           4 Y4 = []
           5
           6 for k in range(len(X)):
           7     Y2.append(X[k]**2)
           8     Y3.append(X[k]**3)
           9     Y4.append(X[k]**4)
          10
          11 plt.clf()
          12 plt.plot(X,X)
          13 plt.plot(X,Y2)
          14 plt.plot(X,Y3)
          15 plt.plot(X,Y4)
          16 plt.xlabel('antécédents')
          17 plt.ylabel('images')
          18 plt.legend(['y=x', 'y=x^2', 'y=x^3', 'y=x^4'])
          19 plt.show()

```

## II. À vous de jouer

### Exercice 5 :

Tracer la courbe représentative de la fonction  $f : x \mapsto \sin(x)$  sur  $[-2\pi, 2\pi]$ .

Mettre la couleur du graphique en rouge, changer l'aspect des points pour obtenir des étoiles et donne des titres aux axes et à la figure.

On pourra utiliser la commande `np.pi` pour utiliser une valeur approchée du nombre  $\pi$ .

Entrée[ ]: 1

### Exercice 6 :

Tracer la courbe représentative de la fonction  $g : x \mapsto \frac{(x-3)^2}{(x-1)(x+5)}$  sur  $[2, 10]$ .

Entrée[ ]: 1

### Exercice 7 :



Représenter sur un même graphe au voisinage de 0 (disons, sur  $[-2, 2]$ ) les fonctions suivantes :

$$\begin{cases} g_0(x) = e^x \\ g_1(x) = 1 + x \\ g_2(x) = 1 + x + \frac{x^2}{2} \\ g_3(x) = 1 + x + \frac{x^2}{2} + \frac{x^3}{6} \end{cases}$$

Les fonctions  $g_1$ ,  $g_2$  et  $g_3$  sont appelées *approximations de la fonction exponentielle au voisinage de 0*. Nous aborderons ce thème au second semestre.

Entrée[ ]: 1

### III. Fin du TP

- **Effectuez** une sauvegarde personnelle du document obtenu :
  - utilisez le menu Fichier / Enregistrer sous... .
  - Le fichier sera alors téléchargé dans votre répertoire Téléchargements .
  - Charge à vous de le placer dans votre répertoire.
- **Rendez** votre travail en utilisant le bouton  en haut à droite.
- **Fermez** ce document :
  - Cliquez sur le bouton  en haut à droite.
  - Ou fermez simplement le navigateur.