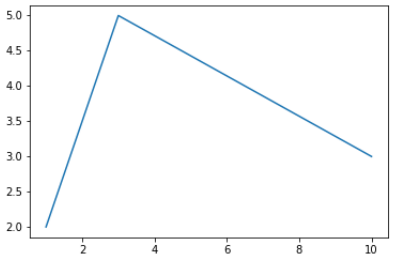
# TP 7 Python – Tracer des courbes

## 1. La base : `matplotlib.pyplot`, `plot` et `show`

Le module `matplotlib.pyplot` de Python permet de tracer des courbes. Vous le verrez toujours importé de la manière suivante : `import matplotlib.pyplot as plt`

La fonction principale à connaître de ce module est `plt.plot(X,Y)` où `X` et `Y` sont, respectivement, une liste d'abscisses et une liste d'ordonnées de points. Les points sont alors, *par défaut*, reliés par des segments de droite.

```
import matplotlib.pyplot as plt
X=[1, 3, 10]
Y=[2, 5, 3]
plt.plot(X, Y)
plt.show()
```

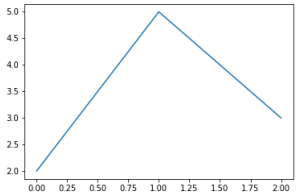



⚠ Les listes `X` et `Y` doivent faire la même longueur.
⚠ Ne pas oublier d'importer le module au début, et d'afficher le graphique à la fin à l'aide de la fonction `plt.show()` !

**Exercice 1.** Tracer et afficher un carré de côté 2, centré en l'origine du repère.

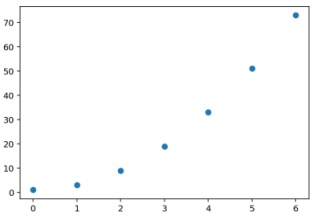
On peut également fournir une unique liste `Y` comme argument de la fonction `plot`. Cette liste sera alors la liste des ordonnées. Les abscisses par défaut sont $0, 1, 2, \ldots$.

```
plt.plot(Y)
```




Pour tracer les termes d'une suite numérique, il est souhaitable de **ne pas relier les points par des segments**. Dans ce cas, mieux vaut utiliser `scatter(X,Y)` au lieu de `plot(X,Y)` :

```
import matplotlib.pyplot as plt
X = [0, 1, 2, 3, 4, 5, 6]
Y = [2*n**2 + 1 for n in X]
plt.scatter(X, Y)
plt.show()
```




**Exercice 2.** Tracer les 30 premières valeurs de la suite $(u_n)_{n\in\mathbb{N}^*}$ définie par $u_n = 2 + \dfrac{(-1)^n}{n}$. On pourra utiliser l'instruction `list(range(k))` pour obtenir la liste des abscisses `X`.

## 2. Créer les listes d'abscisses et d'ordonnées avec `numpy`

Pour tracer la courbe représentative d'une fonction $f$ sur un segment $[a, b]$, on a donc besoin d'une liste d'abscisses régulièrement réparties sur $[a, b]$. Pour ce faire, on peut utiliser deux fonctions présentes dans le module `numpy`. Ce module permet entre autres de manipuler des listes, des tableaux *et des fonctions mathématiques*. Toutes les fonctions présentes dans le module `math` y sont présentes ! On l'importera de la façon suivante : `import numpy as np`

- La fonction `np.linspace(a, b, n)` renvoie une liste de $n$ valeurs régulièrement réparties sur l'intervalle $[a, b]$ (les bornes sont incluses).
- La fonction `np.arange(a, b, dx)` renvoie une liste de valeurs de l'intervalle $[a, b[$ ($b$ est exclu !), avec un pas de `dx`.

Le choix du nombre de valeurs (ici `n`) ou du pas (ici `dx`) est un compromis entre la précision du tracé et le temps de calcul. En général, une centaine de points est suffisante pour un bon rendu visuel.

**Exemple**

```
import numpy as np
X1 = np.linspace(0, 1, 11) #11 valeurs régulièrement réparties sur [0,1]
X2 = np.arange(0, 1, 0.1)  #[0, 0 + 0.1, 0 + 2*0.1, ...] jusqu'à 1 exclu
print(X1)
print(X2)
```

affiche

```
[ 0.   0.1  0.2  0.3  0.4  0.5  0.6  0.7  0.8  0.9  1. ]
[ 0.   0.1  0.2  0.3  0.4  0.5  0.6  0.7  0.8  0.9]
```

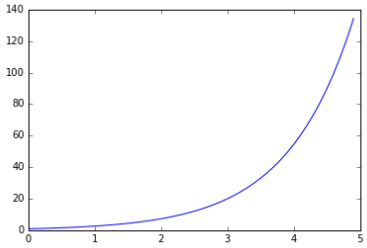
**Exercice 3.** Pour chacune des listes en progression arithmétique données ci-dessous, créer cette liste dans Python de deux manières différentes, à l'aide de chacune des fonctions `linspace` et `arange`.

a) $L_1 = [1, 1.01, \dots, 2]$ b) $L_2 = [-3, -2.9, \dots, 3]$ c) $L_3 = [0, 0.05, \dots, 3.95]$ d) $L_4 = [0.001, 0.002, \dots, 9.999]$

**Exemple**

1. Pour tracer la courbe représentative de la fonction exponentielle sur $[0, 5]$.

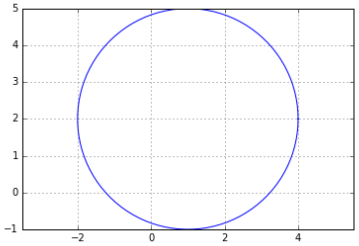
```
X = np.linspace(0, 5, 100) #100 valeurs réparties
    sur [0,5]
Y = [np.exp(x) for x in X]  #images de toutes les
    valeurs de X
plt.plot(X, Y)
plt.show()
```




2. Pour tracer le cercle de centre $(1, 2)$ et de rayon 3 en utilisant sa représentation paramétrique :

$$\begin{cases} x(t) = 1 + 3\cos(t) \\ y(t) = 2 + 3\sin(t) \end{cases}, \quad t \in [0, 2\pi]$$

```
T = np.linspace(0, 2 * np.pi, 100)
X = [1+3*np.cos(t) for t in T]
Y = [2+3*np.sin(t) for t in T]
plt.axis("equal")       #repère orthonormé
plt.grid()              #quadrillage
plt.plot(X, Y)
plt.show()
```




**Exercice 4.** Tracer le graphe de la fonction logarithme népérien sur l'intervalle $\left[\frac{1}{100}, 50\right]$.

# 3. Amélioration des tracés

## 3.A) L'essentiel

Pour rendre un graphique lisible, on donne des noms aux axes, aux courbes et à la figure, on choisit une échelle pertinente, etc.
Tout ça se fait très facilement grâce à quelques fonctions du module `matplotlib.pyplot`.

```
plt.title("titre")     # donne un titre à la figure
plt.xlabel("nom_abs")  # donne un nom à l'axe des abscisses
plt.ylabel("nom_ord")  # donne un nom à l'axe des ordonnées
plt.legend(["nom_courbe"]) # donne une légende pour les différentes courbes
```

Pour forcer à avoir un **repère orthonormé** :

```
plt.axis('equal')
```

Pour afficher un **quadrillage** en fond :

```
plt.grid()
```

Pour fixer les **limites du repère** en abscisses/ordonnées :

```
plt.xlim(0,2) #les abscisses sont limitées à [0,2]
plt.ylim(-1,3) #les ordonnées sont limitées à [-1,3]
```

**Exercice 5.** Améliorez votre tracé du logarithme népérien en affichant le quadrillage et en donnant un titre à la figure, aux axes et à la courbe.

## 3.B) Style du tracé

On peut également modifier la couleur, le type de tracé, le type de marquage des points, etc.

- Options de couleur : `'b'` : bleu, `'r'` : rouge, `'g'` : vert, `'k'` : noir, …
- Types de ligne : `'-'` : trait plein, `'--'` : tirets, `'-.'` : alterné, …
- Marque : `'o'` : rond, `'x'` : croix, `'*'` : étoile, …

**Exemple**

```
plt.plot(X,Y,"r")   #courbe tracée en rouge
plt.plot(X,Y,"--")  #courbe tracée en tirets
plt.plot(X,Y,":")   #courbe tracée en pointillés
plt.plot(X,Y,"-b")  #trait plein, en bleu
plt.plot(X,Y,"xg")  #les points sont marqués d'une croix verte, non reliés
plt.plot(X,Y,"*k-") #les points sont marqués d'une étoile noire et reliés par un
    trait plein
```

## 3.C) Tracé dans un même repère

Pour tracer plusieurs courbes dans un même repère, il suffit d'utiliser plusieurs fois la fonction `plt.plot` avant de tout afficher à l'aide d'un unique `plt.show`
Pour la légende, il faut alors donner une liste de nom, avec autant de noms que de courbes tracées : `plt.legend(["nom_1", "nom_2", …])`

**Exercice 6.**
1. Rappelez l'équation de la tangente en 1 à la courbe du logarithme népérien.
2. Ajoutez cette droite à votre graphe, en prenant des abscisses allant de $-5$ à 5.
3. Légendez votre graphique.

## 4. Bilan

Les commandes de la bibliothèque de `matplotlib` **ne sont pas connaître par coeur**. Agro-Véto les rappelle sous cette forme :

**Matplotlib.pyplot**

`import matplotlib.pyplot as plt`

`plt.plot(X,Y,'+-r')` ---- Génère la courbe des points définis par les listes `X` et `Y` (abscisses et ordonnées) avec les options :
- symbole : `'.'` point, `'o'` rond, `'h'` hexagone, `'+'` plus, `'x'` croix, `'*'` étoile, ...
- ligne : `'-'` trait plein, `'--'` pointillé, `'-.'` alterné, ...
- couleur : `'b'` **bleu** , `'r'` **rouge** , `'g'` **vert** , `'c'` **cyan** , `'m'` **magenta** , `'k'` **noir** , ...

`plt.bar(X,Y)` ---------- Génère l'histogramme des points définis par les listes `X` et `Y` (abscisses et ordonnées)

`plt.axis('equal')` ------ Rend le repère orthonormé

`plt.xlim(xmin,xmax)` ---- Fixe les bornes de l'axe des abscisses

`plt.ylim(ymin,ymax)` ---- Fixe les bornes de l'axe des ordonnées

`plt.show()` ------------ Affiche le graphique

Dans ce qui suit, tous les tracés sont bien évidemment à réaliser dans Python. On prendra garde à toujours donner un nom à la figure, aux axes et aux différentes courbes (légende).

**Exercice 7.** On considère la fonction $g$ définie sur $[-1,2[$ par $g(x) = \begin{cases} 2x^2 & \text{pour } -1 \leq x < 0 \\ x & \text{pour } 0 \leq x < 1 \\ \ln(x)+1 & \text{pour } 1 \leq x < 2 \end{cases}$

1. Définir la fonction $g$ en Python.
2. Tracer la courbe représentative de $g$ sur $[-1,2[$.

**Exercice 8.** Tracer dans un même repère orthonormé les graphe de plusieurs fonctions $x \in \mathbb{R}_+^* \mapsto x^\alpha$ avec $\alpha \in \mathbb{R}$. On fera apparaître tous les cas de figures vus en cours de maths.

**Exercice 9.** On considère les fonctions $f_a : x \mapsto a^x$ et $g_a : x \mapsto \log_a(x)$.

1. Rappeler les valeurs de $a$ pour lesquelles ces fonctions existent, leur domaine de définition respectif et le lien entre $f_a$ et $g_a$.
2. Quel est le lien entre les courbes représentatives de $f_a$ et de $g_a$ ?
3. Tracer dans un même repère orthonormé $f_2$ et $g_2$, en faisant aussi apparaître l'axe de symétrie. On légendera soigneusement.
4. Recommencer la question précédente avec 2 ou 3 autres valeurs de $a$ pertinentes, pour faire apparaître tous les cas de figure possible.

**Exercice 10.**
1. Tracer la courbe représentative de la fonction tangente en se restreignant aux $x \in [-2\pi, 2\pi]$ et $y \in [-4,4]$.
2. Faire apparaître sur le même graphique les asymptotes verticales, ainsi que la tangente en 0.

**Exercice 11.** Avec les outils actuellement à notre disposition, peut-on tracer le graphe de la fonction partie entière ? Justifier.

**Exercice 12.** Quelle jolie figure est cachée derrière la courbe paramétrée suivante ?

$$\begin{cases} x(t) = \sin^3(t) \\ y(t) = \cos(t) - \cos^4(t) \end{cases}, \qquad t \in [0, 2\pi]$$

**Exercice 13.** Tracer sur l'intervalle $]0,1]$, sur un même graphique, les courbes représentatives de la famille de fonctions $f_n : x \to \dfrac{e^{\frac{-1}{nx}}}{nx+2}$ pour $n \in \{1, \ldots, 20\}$.

**Exercice 14.** On souhaite représenter l'ensemble des nombres complexes de la forme $te^{it}$ avec $t \in \mathbb{R}$. On note $E$ cet ensemble.

1. Quel nom pertinent pourrait-on donner aux axes du graphique ?
2. Soit $t \in \mathbb{R}$. Mettre $te^{it}$ sous forme algébrique.
3. En déduire une représentation paramétrique de $E$.
4. Tracer cet ensemble pour $t \in [-20, 20]$.

**Exercice 15** (*)**.** Écrire une fonction `polygone(n)`, qui prend en argument un entier naturel $n \geqslant 3$ et qui réalise et affiche le tracé d'un polygone régulier à $n$ côtés inscrit dans le cercle unité.

**Exercice 16** (*)**.** Tracer un pentacle.