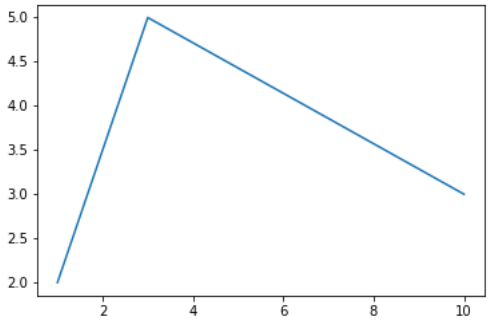
Cours 2 – Révisions de PTSI – Tracé de courbes, calculs d'intégrales

I Tracé de courbes en Python

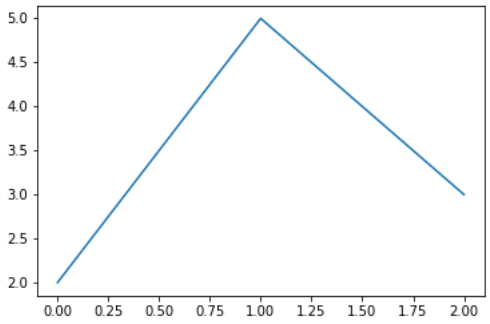
On utilise le module `matplotlib.pyplot` pour tracer des courbes. On utilise la fonction `plot(X,Y)` où `X` et `Y` sont, respectivement, une liste d'abscisses et une liste d'ordonnées de points. Les points sont alors reliés par des segments de droite.

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



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, ... (Cela peut être utile par exemple pour représenter les termes d'une suite).

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



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`.

— la fonction `linspace(a,b,n)` renvoie une liste de n valeurs régulièrement réparties sur $[a,b]$ (les bornes sont incluses)

— la fonction `arange(a,b,dx)` renvoie une liste de valeurs de $[a,b[$ séparées par un écart de `dx`. (*i.e.* $[a, a+dx, a+2dx, \ldots]$)

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.

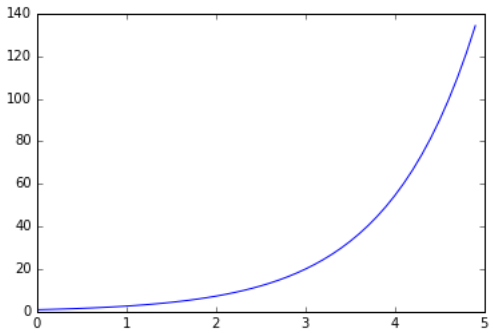
```
import numpy as np
X1=np.linspace(0,1,11) #11 valeurs régulièrement réparties
                       #sur [0,1]
X2=np.arange(0,1,0.1) #[0, 0+0.1, 0+2*0.1, ...] jusqu'à 1 exclu
print(X1)
print(X2)
```

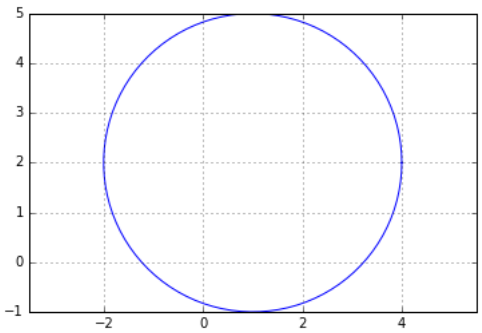
affiche

```
[ 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]
```

Exemples :

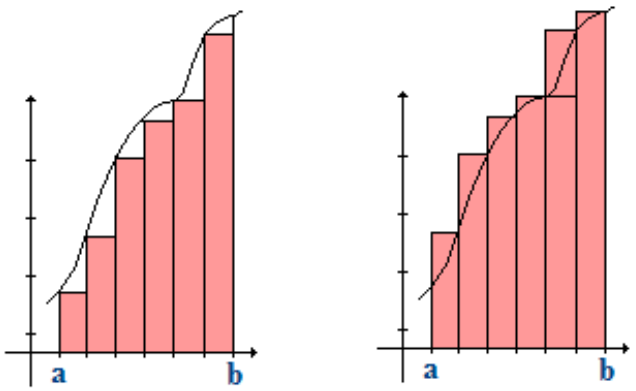
1. Tracer la courbe représentative de la fonction exponentielle sur [0,5]
2. Tracer le cercle de centre (1,2) et de rayon 3 en utilisant une représentation paramétrique.





II Calcul approché d'une intégrale sur un segment

II.1 Méthode des rectangles



Soit $f \in \mathcal{C}([a,b],\mathbb{R})$. On cherche à calculer $\displaystyle\int_a^b f(t)\mathrm{d}t$

On approxime la valeur de cette intégrale en utilisant les sommes de Riemann :

$$S_n(f) = \frac{b-a}{n}\sum_{k=0}^{n-1} f\left(a + k\frac{b-a}{n}\right)$$

On a vu dans le cours de math de PTSI que $\lim\limits_{n\to+\infty} S_n(f) = \int_a^b f(t)\mathrm{d}t$. On estime donc que $S_n(f)$ approxime bien la valeur de l'intégrale pour n suffisamment grand.

Écrivons une fonction `rectangles(f,a,b,n)` prenant en arguments une fonction `f`, les bornes `a` et `b` de l'intervalle ainsi qu'une valeur de `n` et calculant une valeur approchée de $\int_a^b f(t)\mathrm{d}t$ par la méthode des rectangles.

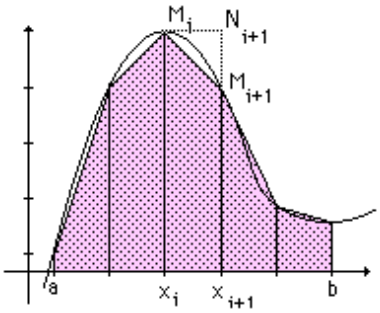
Exemple : Testons la fonction `rectangles` sur la fonction $f : x \mapsto x^2$ sur le segment $[0,1]$.

```
print(rectangles(f,0,1,100))
```

affiche

```
0.32835000000000003
```

II.2 Méthode des trapèzes



Pour obtenir une meilleure approximation de l'intégrale, on va calculer, au lieu des aires des rectangles, les aires des trapèzes obtenus avec les points de la courbe d'abscisses $a + k\dfrac{b-a}{n}$.

On calcule donc $T_n(f) =$

On en déduire une fonction `trapezes(f, a, b, n)` qui approxime $\int_a^b f(t)\mathrm{d}t$ par la méthode des trapèzes.

Comparons les résultats de `trapezes` et de `rectangles` sur la même fonction, avec le même nombre d'itérations.

```
print(trapezes(f,0,1,100))
```

affiche

```
0.3333499999999999
```

Conclusions :

— Les deux méthodes demandent n (ou $n+1$) évaluations de la fonction f

— Le choix de n influence ainsi le temps de calcul et la précision : plus n est grand, plus il y a de calculs et plus le résultat est précis

— La méthode des trapèzes est plus précise que celle des rectangles

II.3 Résolution avec scipy.integrate

```
In [11]: import scipy.integrate as spi
```

La fonction `spi.quad(f, a, b)` calcule une valeur approchée de l'intégrale de f sur $[a, b]$. Elle renvoie un couple de valeurs :

— la valeur approchée de l'intégrale

— une estimation de l'erreur commise

Exemple : calcul de $\displaystyle\int_0^1 (x^3 - 1)\mathrm{d}x$

III Exercices

1. On considère la fonction g définie sur $[0, 2[$ par

$$g(x) = \begin{cases} x & \text{pour } 0 \leq x < 1 \\ 1 & \text{pour } 1 \leq x < 2 \end{cases}$$

   Définir la fonction g en Python, puis tracer sa courbe représentative sur $[0, 2[$ (avec un pas de 0.01 en abscisse).

2. Tracer la courbe de représentation paramétrique $\begin{cases} x(t) = t - \sin(t) \\ y(t) = 1 - \cos(t) \end{cases}$, $t \in [-2\pi, 6\pi]$

3. Améliorer les tracés précédents en donnant des titres à la figure, aux axes et à la courbe (cf. annexe au dos)

4. Tracer la courbe représentative de la fonction cos sur $[-\pi/2, 3\pi/2]$. Tracer les tangentes horizontales à la courbe.

5. Tracer la courbe représentative de la fonction exp sur $[-2, 2]$ ainsi que la tangente à la courbe au point d'abscisse 0.

6. Tracer un rectangle de longueur 5 et de largeur 3

7. Tracer un polygone régulier à 10 côtés inscrit dans le cercle unité.

8. Tracer une étoile à cinq branches.

IV Pour améliorer les tracés

Pour créer une nouvelle figure *(mettre un nom de figure à la place de « titre »)* :

```
plt.figure("titre")
```

Pour forcer à avoir un repère orthonormé :

```
plt.axis("equal")
```

Pour afficher un quadrillage en fond :

```
plt.grid()
```

Pour fixer des limites au 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]
```

Pour choisir des couleurs, un style de trait pour le tracé,...

```
plt.plot(X,Y,"r") #courbe tracée en rouge
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
```

— Options de couleur : `'b'` : bleu, `'r'` : rouge, `'g'` : vert, `'k'` : noir, ...

— Types de ligne : `'-'` : trait plein, `'--'` : pointillés, `'-.'` : alterné, ...

— Marque : `'o'` : rond, `'x'` : croix, `'*'` : étoile, ...

Pour donner des titres, une légende :

```
plt.xlabel("nom") #mettre un nom à l'axe des abscisses
plt.ylabel("nom") #mettre un nom à l'axe des ordonnées
plt.title("titre") #mettre un titre à la figure
plt.legend(["nom1","nom2","nom3"]) #donne une légende au graphique
#nom1 est le nom de la 1ère courbe, #nom2 de la 2ème...
```

Pour sauvegarder une figure

```
plt.savefig("image.png") #enregistre la figure dans le fichier image.png
```