Exercice 1 : Vrai ou Faux

1) Pour importer deux fonction $f1$ et $f2$ d'un module *mod*, on écrit `from mod import f1 and f2 ?`
2) Pour importer une fonction $f1$ d'un module *mod*, on écrit `import f1 from mod ?`
3) Dans l'écriture `mystere.myst, mystere` est une fonction et `myst` un module ?
4) Afin de construire les points de coordonnées (1 ;2) et (3 ;4), on écrit l'instruction `plt.plot([1,2],[3,4])` ?
5) Pour ouvrir en lecture un fichier nommé fichier.txt, on peut écrire `f=open(fichier.txt,r)`

1)Non and est une fonction logique, on écrit `from mod import f1,f2`

2)Non, on écrit `from mod import f1`

3)Non, mystère est le module et myst la fonction

4)Non, `plt.plot([1,3],[2,4])`

5)Non, `f=open('fichier.txt','r')`

Exercice 2 :

Pour importer dans un fichier la fonction *sqrt* du module *math*, on peut écrire :

```
a) import sqrt from math
b) insert sqrt from math
c) from math import sqrt
d) from math insert sqrt
```

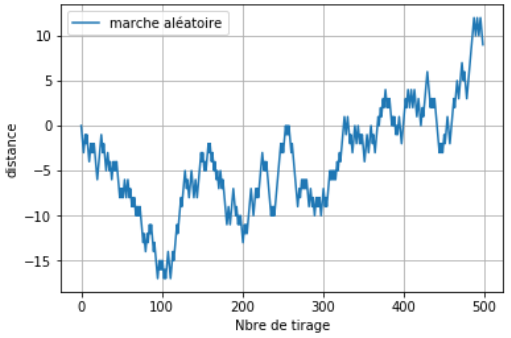
C'est la réponse c

Exercice 3 :

On simule une marche aléatoire. Le point de départ est l'origine du repère. Les abscisses représentent les numéros (entiers) des n étapes. La distance à l'origine est représentée en ordonnée. A chaque étape, cette distance augmente de 1 ou diminue de 1 de manière équiprobable.

Tracer la figure avec Matplolib en prenant par exemple $n = 500$ (n est un paramètre). On rappelle que `rd.randint(0,2)` tire au hasard 0 ou 1.

```
from numpy import random as rd
import matplotlib.pyplot as plt
def hasard(n):
    x=[0]
    y=[0]
    for i in range(1,n):
       x.append(i)
       y.append(y[i-1]+2*rd.randint(0,2)-1)#borne finale
exclue
    plt.plot(x,y,label="marche aléatoire")
    plt.legend()
    plt.grid
    plt.xlabel("Nbre de tirage")
    plt.ylabel("distance")
    plt.show()
```



Exercice 4 :

1) Ecrire une fonction *maximum* qui renvoie le maximum de deux nombres et une fonction *minimum* qui renvoie le minimum de deux nombres, sans utiliser les fonction *min* et *max*. Ces deux fonctions sont écrites dans un fichier *exo4.py* et constitue un module.
2) Ecrire dans un autre fichier, après avoir importer le module précédent, une fonction qui renvoie le maximum de 4 nombres

```
from exo4 import maximum
def quatre(nbr1,nbr2,nbr3,nbr4):
    m1=maximum(nbr1,nbr2)
    m2=maximum(nbr3,nbr4)
    return maximum(m1,m2)
```

Exercice 5 :

Créer un module *stats* qui contient deux fonctions *somme* et *moyenne*. Ces deux fonctions prennent une liste de nombres non vide en paramètre. La fonction *somme* renvoie la somme des éléments de la liste et la fonction moyenne renvoie la moyenne des éléments de la liste.

Tester ce module en l'important dans un autre fichier où les deux fonctions sont utilisées.

```
from exo5 import moyenne,somme
L=[0,1,2,3,4]
print(somme(L))
print(moyenne(L))
```

Exercice 6 :

Ecrire une fonction *variance* qui prend en argument une liste de nombre et renvoie la variance de ces nombres. Utiliser la fonction *moyenne* du module *stats*. On rappelle que la variance d'une liste est donnée par :

$$< \sum_{i=0}^{n-1} L_i^2 > - < \sum_{i=0}^{n-1} L_i >^2$$

```
from exo5 import moyenne
def variance(L):
    L_carre=[]
    for i in range(len(L)):
        L_carre.append(L[i]**2)
    return moyenne(L_carre)-moyenne(L)**2
from exo5 import moyenne
def variance2(L):
    L_carre=[i**2 for i in L]
    return moyenne(L_carre)-moyenne(L)**2
```

Exercice 7 :

1) Construire, sous python, un fichier nommé *data.csv* en reportant les données du tableau ci-dessous

largeur	longueur	Hauteur
1.1	2.3	3.6
4.4	5.6	6.0

2) Ecrire un code permettant de récupérer les lignes contenant des valeurs numériques et de stocker l'ensemble de ces lignes dans une liste dont chaque élément est une liste constituée par les trois valeurs de type float d'une ligne.

```
fichier=open("data.csv","w")
fichier.write("largeur"+";"+"longueur"+";"+"hauteur"+"\n")
fichier.write(str(1.1)+";"+str(2.3)+";"+str(3.6)+"\n")
fichier.write(str(4.4)+";"+str(5.6)+";"+str(6.0)+"\n")
fichier.close()

fichier=open("data.csv","r")
tab=fichier.readlines()
fichier.close()
n=len(tab)
liste_finale=[]
for i in range(1,n):
    ligne=tab[i]
    ligne=ligne.split(";")
    liste_finale.append([float(j) for j in ligne])
print(liste_finale)
```

Exercice 8 :

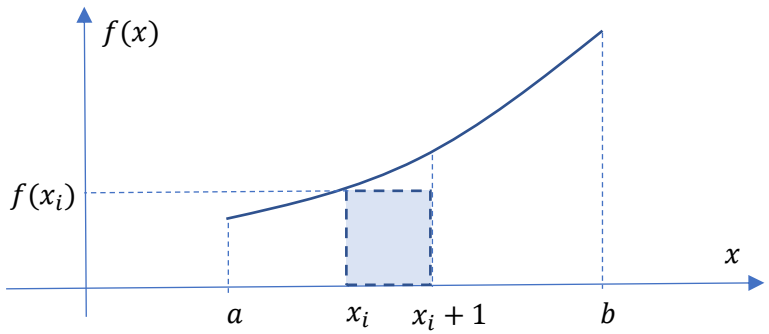
L'objectif est d'écrire une fonction qui trace, à l'aide de la bibliothèque Matplotlib, la courbe représentant une fonction f sur un intervalle $[a, b]$, en utilisant n points. On importe au préalable le module *pyplot* de Matplotlib.

Ecrire une fonction *trace* qui prend en argument deux nombre a et b, une fonction f et un entier n. L'appel de $trace(a, b, f, n)$ permet d'obtenir le tracé de la courbe.

```
import matplotlib.pyplot as plt
def trace(a,b,f,n):
    pas=(b-a)/(n-1)
    x=[i*pas for i in range(n)]
    y=[f(i) for i in x]
    plt.plot(x,y,label="$f(x)$")
    plt.legend()
    plt.grid()
    plt.xlabel("x")
    plt.ylabel("y")
    plt.show()
def f(x):
    return x**2
trace(0,10,f,1000)
```

Exercice 9 :

Une manière d'approximer le calcul d'intégration d'une fonction définie de manière discrète est d'estimer la somme des aires des rectangles $\frac{f(x_i)}{x_{i+1}-x_i}$:



Ecrire une fonction *rectange_gauche*( ) permettant de calculer l'intégrale d'une fonction f, entre a et b et prenant pour argument f, a, b, N. L'argument N étant le nombre d'intervalles utilisés pour cette méthode des rectangles à gauche. Tester votre fonction pour $f(x) = 3x^2$ pour $x \epsilon [0,10]$ et vérifier que l'erreur de quadrature est linéaire avec N.

On pourra noter que ce calcul permet aussi d'obtenir le calcul de valeur moyenne :

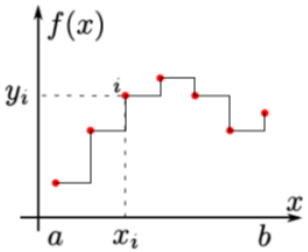
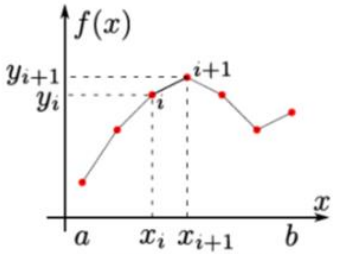
$$f_{moy} = \frac{\int_a^b f(x)dx}{(b-a)} \approx \frac{h\sum_i f(x_i)}{nh} \approx \frac{\sum_i f(x_i)}{n}$$

Exercice 10

Reprendre votre calcul d'intégrale avec la méthode des trapèzes et vérifier que l'erreur est en N^2.

```
def rectangle_gauche(N,a,b,f):
    h=(b-a)/N
    integrale =0
    for i in range(N):#la méthode range ne prend pas de pas
décimaux
        x=a+i*h
        integrale = integrale + h*f(x)
    return integrale
def f(x):
    return 3*x**2
print(1000-rectangle_gauche(100,0,10,f))
print(1000-rectangle_gauche(1000,0,10,f))
def trapeze(N,a,b,f):
    h=(b-a)/N
    integrale =0
    for i in range(N):#la méthode range ne prend pas de pas
décimaux
        x=a+i*h
        integrale = integrale + 0.5*h*(f(x)+f(x+h))
    return integrale

print(1000-trapeze(100,0,10,f))
print(1000-trapeze(1000,0,10,f))
```

Interpolation par morceaux de degré 0	Interpolation par morceaux de degré 1
 Interpolation de degré 0	 Interpolation de degré 1

Approximation de l'aire (rectangle à gauche): $\int_a^b f(x)dx \approx \sum_{i=0}^{n-1} y_i h$	Approximation de l'aire : $\int_a^b f(x)dx \approx \sum_{i=0}^{n-1} (y_i + y_{i+1}) \frac{h}{2}$
Erreur quadratique : $\int_a^{a+h} f(x)dx = F(a+h) - F(a)$ Avec un DL à l'ordre 1 : $\int_a^{a+h} f(x)dx = f(a)h + \frac{f'(\xi)h^2}{2}$ Donc $\int_a^b f(x)dx \approx \sum_{i=0}^{n-1} y_i h$ si on néglige $n\frac{h^2}{2} f'(\xi)_{max} \approx O(h)$	Erreur quadratique : $\int_a^{a+h} f(x)dx = F(a+h) - F(a)$ Avec un DL à l'ordre 2 : $\int_a^{a+h} f(x)dx = f(a)h + \frac{f'(a)h^2}{2} + \frac{f''(\xi)h^3}{6}$ Donc $\int_a^b f(x)dx \approx \sum_{i=0}^{n-1} y_i h + \frac{(y_{i+1} - y_i)h^2}{2h}$ $\int_a^b f(x)dx \approx \sum_{i=0}^{n-1} (y_i + y_{i+1}) \frac{h}{2}$ Si on néglige $n\frac{h^3}{6} f''(\xi)_{max} \approx O(h^2)$