Programmer en Python

1. Les importations de bibliothèques

Les commandes **«from math import *» et « from random import *** » importent toutes les fonctions (c'est le sens de *) de ces bibliothèques. Les principales fonctions utilisées en lycée sont :

- De la bibliothèque « math » :sqrt,pi, sin, cos, tan, abs
- De la bibliothèque « random » : randint, random

```
1   from random import *
2   from math import *
3
4   # racine de 2, pi, sin(pi/4)
5   print(sqrt(2),pi,sin(pi/4))
6
7   # Un nombre aléatoire entre 3 et 7
8   print(randint(3,7))
9
10  # Un nombre décimal entre 0 et 1
11  print(random())
```

Console

```
1.4142135623730951 3.141592653589793 0.7071067811865475
3
0.1355034133436589
```

Remarques :

- En python, **les commentaires** (lignes non exécutées) commencent par le symbole « **#** ». Le code est indenté, ce qui implique en particulier, que tout « copier-coller »de code peut amener à des problèmes de lecture des tabulations.
- Les copies d'écran ont été réalisées à partir du laboratoire Python du site « Le Livre Scolaire » : https://www.lelivrescolaire.fr/console-python

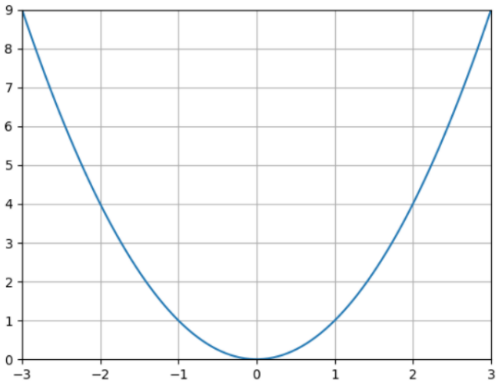
2. Construire une courbe représentative de fonction

Dans le programme suivant, on importe des bibliothèques et fonctions graphiques que l'on renomme

```
import matplotlib.pyplot as plt
import numpy as np
# Définition des axes [xmin,xmax,ymin,ymax]
plt.axis([-3,3,0,9])
# Création de la grille
plt.grid()
# Définition des coordonnées des points avec 100 abscisses allant de -3 à 3
x=np.linespace(-3,3,100)
y=x**2
# Création des points
plt.plot(x,y)
# Affichage du graphique
plt.show()
```

```python
import matplotlib.pyplot as plt
import numpy as np
plt.axis([-3,3,0,9])
plt.grid()
x=np.linspace(-3,3,100)
y=x**2
plt.plot(x,y)
plt.show()
```

Affichage graphique



3. Utiliser une fonction(au sens de la programmation)

Le programme précédent est modifié pour utiliser une fonction.

```
def f(x) :
   return x**2
```

```python
import matplotlib.pyplot as plt
import numpy as np

def f(x):
  return x**2

plt.axis([-3,3,0,9])
plt.grid()
x=np.linspace(-3,3,100)
y=f(x)
plt.plot(x,y)
plt.show()
```

4. L'instruction conditionnelle « if elif else »

```
def je_suis_perdu(x):
# Test "if elif else"
 if x==2:
        print("x=2")
 elif x<3:
        print("x<3")
 else :
  print("je suis perdu")

 #Test unique "if"
 if x<2:
        print("mais que fait donc ce programme?")

 print("-----------------------------------")

#  Lancement de la fonction avec trois arguments différents
je_suis_perdu(2)
je_suis_perdu(1)
je_suis_perdu(5)
```

```
def je_suis_perdu(x):
# Test "if elif else"
  if x==2:
    print("x=2")
  elif x<3:
    print("x<3")
  else :
    print("je suis perdu")

  #Test unique "if"
  if x<2:
    print("mais que fait donc ce programme?")

  print("----------------------------------")

#  Lancement de la fonction avec trois arguments différents
je_suis_perdu(2)
je_suis_perdu(1)
je_suis_perdu(5)
```

Console

```
x=2
----------------------------------
x<3
mais que fait donc ce programme?
----------------------------------
je suis perdu
----------------------------------
```

5. La boucle bornée « for »

Dans ce programme, on utilise la « concaténation, c'est-à-dire la somme de caractères !

```
Etoile="*"
for i in range (10):
 print(Etoile)
 Etoile=Etoile+"*"
```

```
Etoile="*"
for i in range (10):
  print(Etoile)
  Etoile=Etoile+"*"
```

Console →

```
*
**
***
****
*****
******
*******
********
*********
**********
```

6. La commande « range » en détails

```
L = range(1,4)
for nombre in L :
  print (nombre)
print("------")

L = range(4)
for nombre in L :
  print (nombre)
```

```
print("------")

L = range(1,10,2)
for nombre in L :
  print (nombre)
```

```
1   L = range(1,4)
2   for nombre in L : print (nombre)
3   print("------")
4   L = range(4)
5   for nombre in L : print (nombre)
6   print("------")
7   L = range(1,10,2)
8   for nombre in L : print (nombre)
```

```
1
2
3
------
0
1
2
3
------
1
3
5
7
9
```

7. La boucle non bornée « while »

En général, cette boucle est utilisée pour la recherche d'un rang à partir duquel une condition est réalisée. Pour cela, il faut **utiliser la négation de cette condition dans le test**.

```
# Invitation d'entrée de la précision, ici 0.00001
precision = float(input("entrer la précision"))
print("précision entrée = ",precision)

#Initialisation
u=1
n=0

# Boucle while
while u>precision:
   u=0.5*u
   n+=1
# On notera dans la ligne précédente la syntaxe utilisée à la place de n=n+1
print ("la plus petite valeur telle que (0,5)^n<=precision est : ",n)
```

```python
# Invitation d'entrée de la précision, ici 0.00001
precision = float(input("entrer la précision"))
print("précision entrée = ",precision)

#Initialisation
u=1
n=0

# Boucle while
while u>precision:
    u=0.5*u
    n+=1
# On notera dans la ligne précédente la syntaxe utilisée à la place de n=n+1
print ("la plus petite valeur telle que (0,5)^n<=precision est : ",n)

```

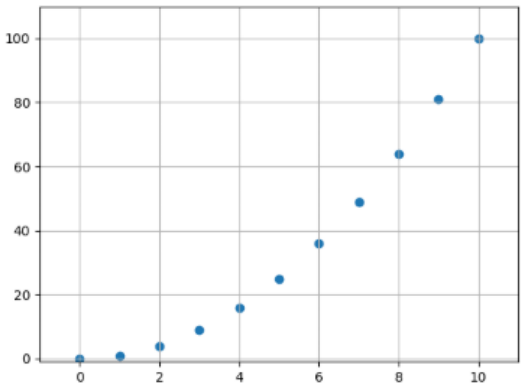
Console →

```
précision entrée = 1e-05
la plus petite valeur telle que (0,5)^n<=precision est : 17
```

8. **<u>Construire un nuage de points (d'une suite numérique par exemple)</u>**

```
import matplotlib.pyplot as plt
# Définition des axes [xmin,xmax,ymin,ymax]
plt.axis([-1,11,-1,110])
# Définition des coordonnées des points du nuage à abscisses entières
x=[n for n in range(11)]
y=[n**2 for n in range(11)]
# Création des points
plt.scatter(x,y)
# Création de la grille
plt.grid()
# Affichage du graphique
plt.show()
```

Affichage graphique →



```python
import matplotlib.pyplot as plt
# Définition des axes [xmin,xmax,ymin,ymax]
plt.axis([-1,11,-1,110])
# Définition des coordonnées des points du nuage à abscisses entières
x=[n for n in range(11)]
y=[n**2 for n in range(11)]
# Création des points
plt.scatter(x,y)
# Création de la grille
plt.grid()
# Affichage du graphique
plt.show()


```

9. **Créer une liste, ajouter et retirer un élément d'une liste, utiliser une liste**

```
from random import randint

# Création de la liste "liste"
liste=[10,20,30,40,50]

# Affichage (attention le premier rang d'une liste est 0 !)
print(liste)
print ("L'élément de rang 2 est ", liste[2])

# Ajout et retrait d'éléments (attention le premier rang d'une liste est 0 !)
liste.append("caramel au beurre salé")
print (liste)
liste.pop(5)
print (liste)

# Création d'une liste avec la somme cumulée des termes
S=0
liste2=[]
for i in range (len(liste)):
  S=S+liste[i]
  liste2.append(S)
print(liste2)

# Création d'une liste de nombres aléatoires 0 et 1 et comptage du nombre de 1
Echantillon=[]
for i in range (10):
  alea=randint(0,1)
  Echantillon.append(alea)
print("Echantillon = ",Echantillon)

Nombre_de_1=0
for j in range(len(Echantillon)):
  Nombre_de_1=Nombre_de_1+Echantillon[j]
print("Nombre de 1 dans l'échantillon = ",Nombre_de_1)
```

```python
from random import randint

# Création de la liste "liste"
liste=[10,20,30,40,50]

# Affichage (attention le prmeir rang d'une liste est 0 !)
print(liste)
print ("L'élément de rang 2 est ", liste[2])

# Ajout et retrait d'éléments (attention le prmeir rang d'une liste est 0 !)
liste.append("caramel au beurre salé")
print (liste)
liste.pop(5)
print (liste)

# Création d'une liste avec la somme cumulée des termes
S=0
liste2=[]
for i in range (len(liste)):
  S=S+liste[i]
  liste2.append(S)
print(liste2)

# Création d'une liste de nombres aléatoires 0 et 1 et comptage du nombre de 1
Echantillon=[]
for i in range (10):
  alea=randint(0,1)
  Echantillon.append(alea)
print("Echantillon = ",Echantillon)

Nombre_de_1=0
for j in range(len(Echantillon)):
  Nombre_de_1=Nombre_de_1+Echantillon[j]
print("Nombre de 1 dans l'échantillon = ",Nombre_de_1)

```

Console →

```
[10, 20, 30, 40, 50]
L'élément de rang 2 est 30
[10, 20, 30, 40, 50, 'caramel au
beurre salé']
[10, 20, 30, 40, 50]
[10, 30, 60, 100, 150]
Echantillon = [0, 0, 1, 1, 1, 1, 0,
1, 1, 0]
Nombre de 1 dans l'échantillon = 6
```

10. Quelques commandes complémentaires (utiles… ou pas !)

Instruction Python	Effet
If a !=2	Si « a » est différent de 2

a%b	Reste de la division euclidienne de « a » par »b »
Liste est une liste	
Liste.index(n)	Indice de la première apparition de l'élément de valeur n
Liste.copy()	Duplication de la liste Liste
List.sort()	Ordonne une liste dans l'ordre croissant ou alphabétique
Random.shuffle(Liste)	Mélange une liste
Liste.count(n)	Compte le nombre d'apparitions de l'élément de valeur n dans la liste .
Liste.remove(n)	Supprime l'élément de valeur n

```
1   liste=[2,7,24,-1,0,7]
2   x=liste.index(7)
3   print("Le rang de la premmière apparition du 7 est ",x)
4   # Attention, le premier rang d'une liste est 0 !
5   y=liste.count(7)
6   print("Le nombre d'apparitions du 7 est  ",y)
7   y=liste.sort()
8   print("La liste ordonnée : ",liste)
9   print("Le reste de la division euclidienne de 7 par 2 est ",liste[5]%liste[0])
10
```

Console →

```
Le rang de la premmière apparition du 7 est 1
Le nombre d'apparitions du 7 est 2
La liste ordonnée : [-1, 0, 2, 7, 7, 24]
Le reste de la division euclidienne de 7 par 2 est 0
```

11. **Quelques programmes :**

Calcul des fréquences d'apparition des numéros lors de lancers d'un dé à 6 faces :

```
import random
N=int(input('Nombre de lancers de dés'))
print( "Le dé a été lancé ", N," fois")
L=[];
for i in range(0,6):
   L.append(0)
print('I------------------I')
for i in range(0,N):
```

```
    resde=random.randint(0,5)
    T=int(1+L[resde])
    L[resde]=T
for i in range(0,6):
    print( '    f(',i+1,')=',float(L[i])/N)
    print('I-------------------I')
```

```
import random
N=int(input('Nombre de lancers de dés'))
print( "Le dé a été lancé ", N," fois")
L=[];
for i in range(0,6):
    L.append(0)
print('I-------------------I')
for i in range(0,N):
    resde=random.randint(0,5)
    T=int(1+L[resde])
    L[resde]=T
for i in range(0,6):
    print( '    f(',i+1,')=',float(L[i])/N)
    print('I-------------------I')
```

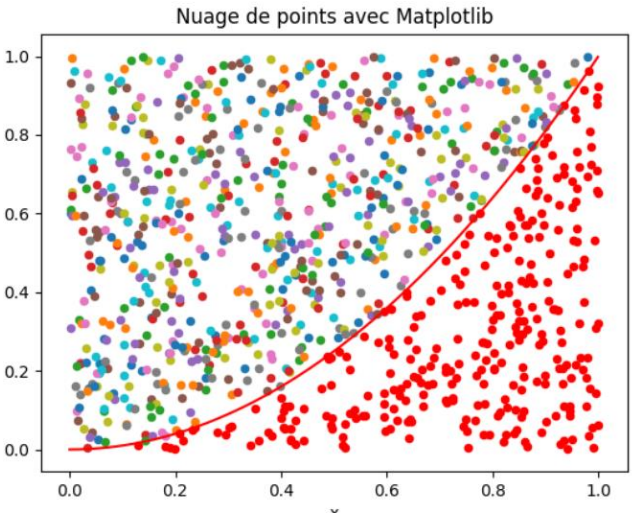
Console

```
Le dé a été lancé 100000 fois
I-------------------I
f( 1 )= 0.16696
I-------------------I
f( 2 )= 0.16607
I-------------------I
f( 3 )= 0.16671
I-------------------I
f( 4 )= 0.16623
I-------------------I
f( 5 )= 0.16848
I-------------------I
f( 6 )= 0.16555
I-------------------I
```

Approximation de l'aire sous la courbe c_f: $y = x^2$ sur [0 ;1] par la méthode de Monte-Carlo

```
import random
import math
import numpy
import matplotlib.pyplot as plt

N=1000
D=0
plt.xlabel('x')
plt.ylabel('y')
for i in range (1,N) :
    x=random.random()
    y=random.random()
    if y<=x*x:
        D=D+1
        plt.scatter(x,y,s=20,c='red')
    else:
        plt.scatter(x,y,s=20)
I=float(D)/N
print ('Aire=',I)
plt.title('Nuage de points avec Matplotlib')
a = numpy.arange(0,1, .001)
b = a*a
plt.plot(a,b,'red')

plt.show()
```

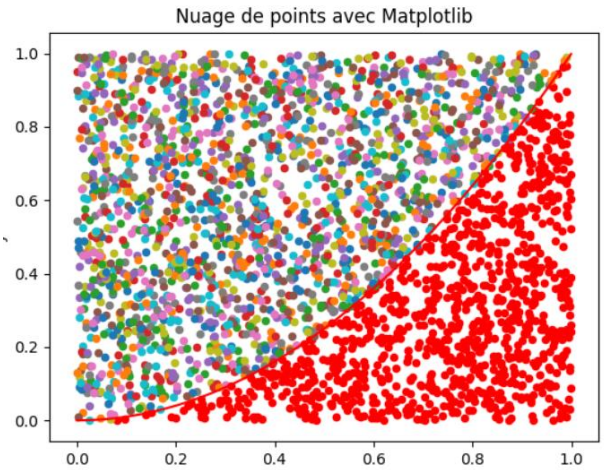
On obtient dans ce cas, pour n=1000, une valeur approchée de 0,294 (pour une valeur exacte de $\frac{1}{3}$).




```python
import random
import math
import numpy
import matplotlib.pyplot as plt


N=1000
D=0

plt.xlabel('x')
plt.ylabel('y')

for i in range (1,N) :
    x=random.random()
    y=random.random()
    if y<=x*x:
        D=D+1
        plt.scatter(x,y,s=20,c='red')
    else:
        plt.scatter(x,y,s=20)
I=float(D)/N
print ('Aire=',I)

plt.title('Nuage de points avec Matplotlib')
a = numpy.arange(0,1, .001)
b = a*a
plt.plot(a,b,'red')
plt.show()
```

Pour n= 3000

On obtient dans ce cas une valeur approchée de 0.3426666666666667



Pour n=4000

On obtient dans ce cas une valeur approchée de 0.33275