Langage de programmation Python

1 Prise en main

1.2. Bibliothèques et modules

- Des morceaux de code prédéfinis, appelés **fonctions**, permettent d'exécuter un ensemble d'instructions. Les **modules** sont des fichiers regroupant plusieurs fonc… …**èque** est un répertoire de modules. Les modules et bib… …s à la suite Anaconda et utilisés dans les fichiers numér… …nt :
  - le module **ma**… …ns mathématiques ;
  - le module **py**… …que **Matplotlib** pour les graphiques ;
  - la bibliothèque **NumPy** pour les tableaux de valeurs.
- Pour accéder à un module ou à une bibliothèque dans un programme, il faut l'« importer » à l'aide d'une ligne de code insérée en début de programme comportant la commande **import**. Il est d'usage de les renommer par une ou plusieurs lettres lors de l'importation, c'est le sens de la commande **as**.
- Le nom donné, appelé **préfixe**, sert à appeler un **élément** du module ou de la bibliothèque pour l'utiliser dans le corps du programme grâce à la syntaxe **préfixe.élément()**. En l'absence de préfixe, l'appel d'un élément du module ou de la bibliothèque se fait avec le nom complet du module (DOC. 4).

```
# importe toutes les fonctions du module math
import math
b = math.cos(math.pi/4) # Affecte cos(pi/4) à b

# importe uniquement la fonction cos et
# la constante pi depuis le module math
from math import cos, pi
b = cos(pi/4)      # Affecte cos(pi/4) à b

# importe le module math en le nommant m
import math as m
b = m.cos(m.pi/4) # Affecte cos(pi/4) à b
```

```
# importe NumPy en la renommant np
import numpy as np

# importe le module pyplot de matplotlib
# en le renommant plt
import matplotlib.pyplot as plt
from matplotlib import pyplot as plt
```

4. Importation et appel des modules et bibliothèques.
Le dièse # permet d'insérer des commentaires dans le programme, tout ce qui est écrit en italique dans une ligne de code après un # n'est pas exécuté.

Les principales fonctions et instructions ainsi que les objets de base utilisés dans le langage de programmation Python sont décrits dans les pages suivantes

2 Présentation des objets de base

Le langage de programmation Python utilise des objets, regroupés par type, qui ont chacun un comportement et un mode d'interaction propres.

2.1. Types de variables et conversions d'un type à un autre

```
# Affectation : signe =
x = ... # signifie « x prend la valeur ... »
```

Exemple

```
x = 2  # Affectation simple

a,b,c = 2.5,2e3,-53 # Affectation multiple

# Affectation combinée à une opération
x += 1 # Affecte x+1 à la variable x
x -= 1 # Affecte x-1 à la variable x
```

```
Type integer <int> # nombre entier relatif
int()   # convertit si possible un réel ou un texte en entier

Type float <float> # nombre réel.
float() # convertit si possible un entier ou un texte en réel

Type string <str>  # chaîne de caractères (texte)
                   # suite de caractères d'imprimerie
                   # délimitée par des guillemets
str()   # convertit un nombre en une chaine de caractères

Type booléen <bool> # logique : ne prend que deux valeurs
                    # True et False (Vrai et Faux)
```

```
x = 2                # entier
y = '13'             # chaîne de caractères
z = 9.23             # réel
t = float('2e-3') # renvoie 0.002
z1 = str(x)+y     # renvoie '213' (chaîne)
z2 = x+int(y)     # renvoie 15 (entier)
```

2.2. Opérations numériques

```
+, -, *, / # opérations mathématiques usuelles
**         # puissance
aeN        # a×10^N
//         # quotient de la division euclidienne
%          # reste de la division euclidienne
round(x,n) # arrondi le nombre x à n décimales
           # sans garder tous les 0 décimaux
```

```
m = 23/4  # renvoie 5.75
p = 3**2  # renvoie 9
q = 23//4 # renvoie 5 car 23=4*'5'+3
r = 23%4  # renvoie 3 car 23=4*5+'3'

round(0.489,2) # renvoie 0.49
round(0.499,2) # renvoie 0.5
round(1.000,2) # renvoie 1.0
```

2.3. Entrées et sorties

- **Entrées :**

```
input('texte')  # affiche texte et prend pour valeur la chaîne
                # de caractères saisie au clavier
int(input('texte'))   # convertit si possible la saisie en entier
float(input('texte')) # convertit si possible la saisie en réel
eval(input('texte'))  # évalue la saisie comme une expression
```

```
# Affiche le texte, convertit en entier la
# valeur saisie et l'affecte à la variable N
N = int(input('Nombre de points ='))
```

Nombre de points = 3

```
# Affiche texte, évalue l'expression saisie
# et affecte à la variable C le résultat
C = eval(input('Calcul ='))
```

Calcul = 2*N

- **Sorties :**

```
print(objet1,objet2,...) # affiche les objets séparés par une tabulation
```

```
print('Le résultat du calcul vaut',C)
```

```
Le résultat du calcul vaut 6
```

3 Présentation des blocs d'instructions

Dans la syntaxe Python, un bloc d'instructions est défini par deux points « : » suivis, à la ligne, d'une indentation fixe (1 tabulation ou 4 espaces). La fin de l'indentation indique la fin du bloc d'instructions.

3.1. Fonction : def : ... return

```
def fonction(arg1,arg2,...): # arg='argument'
    <Instructions>           # exécute les instructions
    return résultat          # renvoie le résultat

fonction(valeur1,valeur2,...)# appel de la fonction
```

```
# fonction d'argument la vitesse v en km/h
# retournant la vitesse en m/s
def v_m_s(v):
    v1 = v*10**3/3600
    return v1

v_m_s(90)
```

```
25.0
```

3.2. Test ou instruction conditionnelle : if : ... elif : ... else :

Un test renvoie une valeur booléenne : True ou False.

```
if (test 1):          # si test 1 vérifié
    <Instructions 1>  # alors exécute les instructions 1
elif (test 2):        # sinon, si test 2 vérifié
    <Instructions 2>  # alors exécute les instructions 2
else:                 # sinon
    < Instructions>   # exécute les dernières instructions
```

```
pH = float(input('pH ='))
```

```
pH = 3.5
```

```
if pH<7:
    print('Acide')
elif pH==7:
    print('Neutre')
else:
    print('Basique')
```

```
Acide
```

```
# Comparateurs
# pour les tests
== # égal
!= # différent
>= # sup ou égal
>  # supérieur
<= # inf ou égal
<  # inférieur
```

3.3. Boucle « tant que » : while :

```
while test:          # Tant que test vérifié
    <Instructions>   # exécute les instructions
```

```
# Recherche du plus petit entier dont
# le carré est supérieur ou égal à 1000
n = 0          # Initialise le compteur
while n**2<1000:
    n += 1  # Incrémente le compteur
print(n)
```

```
32
```

3.4. Boucle « pour » : for ... in... :

```
for élément in ensemble: # Pour chaque 'élément' dans 'ensemble'
    <Instructions>       # repète les instructions
```

```
liste_de_masses = [2.5,1.8]
for m in liste_de_masses:
    poids = m*9.8
    print(poids)
```

```
24.5
17.64
```

Les ensembles parcourus par **in** de la boucle **for** sont des objets dont on peut parcourir les éléments un à un, comme les **chaînes de caractères**, les **listes**, les **tableaux à une dimension** et les **listes d'entiers** de la fonction **range()**.

La boucle **for** et la fonction **range()** sont utilisées dans la création de listes.

```
# Création de liste avec L.append()
L = []       # Initialise une liste vide
for i in range (1,4):#pour i entier de 1 à 3
    L.append(i*10) # insère i*10
                   # en dernière position
    print ('L =',L)
```

```
L = [10]
L = [10, 20]
L = [10, 20, 30]
```

```
# Création de liste en compréhension
L = [i*10 for i in range(1,4)]
print(L)
```

```
[10, 20, 30]
```

4 Représentation graphique

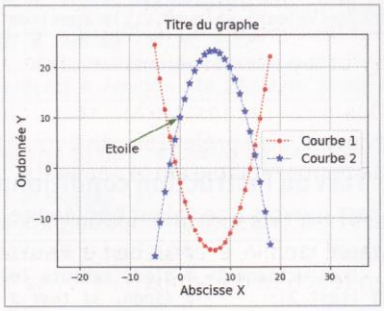
Une représentation graphique en langage Python se fait grâce au module **pyplot** de la bibliothèque **Matplotlib**.

4.1. Représentation d'un nuage de points

La fonction **plt.plot(X,Y)** représente le nuage de points d'abscisses dans X et d'ordonnées dans Y. Pour cela, X et Y, de type liste ou tableau à une ligne, doivent avoir le même nombre d'éléments.

EXEMPLE Les principales fonctions de représentation d'un graphe figurent dans le programme suivant :

```
import matplotlib.pyplot as plt #Importe le module pyplot et le renomme plt
X=list(range(-5,19,1))  # Domaine des abscisses liste des entiers x de -5 à 18
Y1=[0.3*x**2-4*x-3 for x in X]  # Domaine des ordonnées de la courbe 1
                                # liste des y=0.3x2-4x-3
Y2=[-0.3*x**2+4*x+10 for x in X]# Domaine des ordonnées de la courbe 2
                                # liste des y=-0.3x2+4x+10
plt.figure('Titre de la figure')# Initialise et nomme la figure
                                # Option : figsize=(largeur,hauteur)
plt.title('Titre du graphe')    # Nomme le graphe
plt.plot(X,Y1,'or:',ms=4,label='Courbe 1')# Nuage de points de coordonnées
                                # dans X et dans Y1, 'o' points 'r' rouges
                                # de taille (ms=markersize) 4, ':'reliés par des
                                # petits points, label=nom de la courbe
plt.plot(X,Y2,'*b-.',lw=0.5, ms=8,label='Courbe 2')# Nuage de points de
                                # coordonnées dans X et Y2, '*' étoiles 'b' bleues
                                # '-.' reliées par des pointillés d'épaisseur
                                # (lw=linewidth) 0.5, label=nom de la courbe
plt.xlabel('Abscisse X')        # Etiquette de l'axe des abscisses
plt.ylabel('Ordonnée Y')        # Etiquette de l'axe des ordonnées
plt.text(-15,3,'Etoile', fontsize =14) # Affiche le texte 'Etoile' en commençant
                                #  au point de coordonnées -15,3, taille (fontsize)
plt.arrow(-10,5,8,4,color='g',head_width=1) # Trace une flèche verte (g) depuis
                                # le point de coordonnées (x=-10,y=5) jusqu'au point
                                # de coordonnées (x+dx=-10+8,y+dy=5+4)
plt.axis('equal')               # Repère orthonormé
plt.grid()                      # Affiche une grille
plt.legend(loc=7, fontsize=14)# Affiche les noms en légende
                                # position loc 7 (droite, centrée), taille (fontsize)
plt.show()                      # Affiche la figure
```



REMARQUE La liste de toutes les fonctions du module **pyplot** est accessible à partir de la commande **dir(plt)** et l'aide sur la syntaxe d'un élément **nom** du module s'obtient grâce à la commande **help(plt.nom)**.

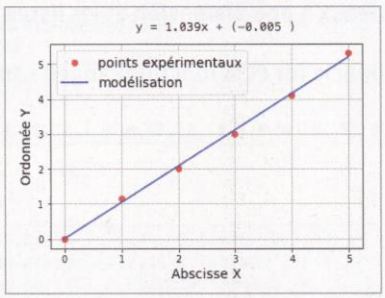
4.2. Modélisation d'un nuage de points expérimentaux

Les coordonnées d'une série de points expérimentaux sont rangées dans deux listes ou deux tableaux à 1D nommés X pour les abscisses et Y pour les ordonnées. Modéliser le nuage de points consiste à déterminer l'équation mathématique de la courbe qui se rapproche le plus de celle qu'ils tracent.

EXEMPLE Modélisation par une droite d'équation $y = ax + b$

La fonction **np.polyfit(X,Y,1)** modélise le nuage de points d'abscisses dans X et d'ordonnées dans Y par un **droite** d'équation $y = ax + b$ et renvoie le tableau : [a b].

```
import numpy as np          # Importe la bibliothèque numpy en np
import matplotlib.pyplot as plt # Importe le module pyplot en plt
X = np.array([0,1,2,3,4,5]) # Tableau des abscisses des points expérimentaux
Y = np.array([0.,1.15,2,3,4.1,5.3])#Tableau des ordonnées des points expérimentaux
plt.plot(X,Y,'ro',label='points expérimentaux') # Nuage des points expérimentaux
                            # d'abscisses dans X et d'ordonnées dans Y
                            # sous forme de points rouges non reliés
Modele = np.polyfit(X,Y,1)  # Calcule les paramètres de la droite modélisant le
                            # nuage de points et les range dans le tableau Modele
a,b = [coef for coef in Modele]# Affecte dans cet ordre les paramètres du modèle
                            # aux variables a et b
plt.plot(X,a*X+b,'b-',label='modélisation') #Nuage de points d'abscisses dans X et
                            # d'ordonnées dans a*X+b, en bleu et sous forme reliée
print('y=',round(a,3),'x+(',round(b,3),')')# Affiche l'équation de la droite modèle
plt.grid()                  # Affiche une grille
plt.legend()                # Affiche la légende
plt.show()                  # Affiche la figure
```



REMARQUE La fonction **np.polyfit(X,Y,2)** modélise le nuage de points d'abscisses dans X et d'ordonnées dans Y par la courbe d'équation $y = ax^2 + bx + c$ et renvoie le tableau [a b c].

Les instructions concernant l'animation d'une courbe pour la simulation de la propagation d'une onde sont développées dans le fichier numérique disponible sur le site sirius.nathan.fr.