

COURS

Initiation au langage Python

Table des matières

1	Objectif du document	2
2	Présentation langage Python	2
3	Comment utiliser ce document ?	2
4	Types Variables et Operateurs	3
4.1	Type int (integer : nombres entiers)	3
4.2	Type float (flottant ou à virgule flottante)	3
4.3	Type bool (booléen)	4
4.4	Type str (string ou chaîne de caractères)	4
4.5	Type list (liste)	6
4.6	Type dict (dictionnaire)	7
4.7	Autres types	7
4.8	Variables	7
5	Entrées / Sorties	8
5.1	Fonction INPUT()	8
5.2	fonction PRINT()	9
6	Structures alternatives	9
6.1	Instruction IF (SI)	9
6.2	Instruction ELSE (SINON)	10
6.3	Instruction ELIF (SINON SI)	10
7	Structures répétitives	11
7.1	Boucle WHILE (TANT QUE)	11
7.2	Boucle FOR...IN RANGE (Boucle POUR)	12
7.3	Instruction BREAK pour sortir d'une boucle infinie	14
8	Fonctions	14
8.1	Utilité des fonctions	14
8.2	Syntaxe	14
8.3	Passage de paramètres à la fonction	15
8.4	Sortir des résultats d'une fonctions	16
8.5	Portées des variables : variable LOCALES et variable GLOBALES	16
9	Modules et packages	17

9.1 Modules et packages	17
9.2 Bibliothèque standard	17

1 Objectif du document

Ce document a pour vocation de faire une introduction aux éléments de base de la programmation en Python. Pour que cela soit efficace, je vous invite à faire un copié-collé des scripts Python et d'observer le comportement. Pour ce faire vous pouvez utiliser les logiciels ou lien suivants :

- ▶ Thonny (installé sur les ordinateurs de la salle)
- ▶ Via Toutatice (CAPYTALE → Console Python)

2 Présentation langage Python

Le langage Python est un langage de programmation objet **interprété**. Python offre un environnement complet de développement comprenant un interpréteur performant et de **nombreux modules**. Un atout indéniable est sa disponibilité sur la grande majorité des plates-formes informatiques courantes : Mac OS X, Unix, Windows ; Linux, Android, IOS. . . Python est un langage **open source**. **Libre et gratuit**.

Avec le langage Python il est possible de faire :

- ▶ du calcul scientifique (bibliothèque **NumPy**) ;
- ▶ des graphiques (bibliothèque **matplotlib**) ;
- ▶ du traitement du son ;
- ▶ du traitement d'image (bibliothèque PIL) ;
- ▶ des applications avec interface graphique GUI (bibliothèques **Tkinter**, **PyQt**, wxPython, PyGTK . . .) ;
- ▶ des jeux vidéo en temps réel (bibliothèque Pygame) ;
- ▶ des applications Web (serveur Web Zope ; framework Web Django, Karrigell ; framework JavaScript Pyjamas) ;
- ▶ interfacier des systèmes de gestion de base de données (bibliothèque MySQLdb . . .) ;
- ▶ des applications réseau (framework Twisted) ;
- ▶ communiquer avec des ports série RS232, Bluetooth. . . (bibliothèque **PySerial**) ;
- ▶ . . .

En deux mots, il est possible de tout faire en Python !

3 Comment utiliser ce document ?

Ce document va vous présenter les rudiments de la programmation en Python, les **bases fondamentales** pour une bonne fluidité à l'apprentissage. Ce document est loin d'être exhaustif, vous trouverez sur le site w3schools dans l'onglet dédié à Python, la référence du langage.

4 Types Variables et Operateurs

4.1 Type int (integer : nombres entiers)

Un entier peut être exprimé en décimal, en binaire ou hexadécimal.

Script PYTHON : Désignation du script

```
>>> 2013 # ecriture en decimal
2013
>>> 0b11111011101 # ecriture en binaire
2013
>>> 0x7DD # ecriture en hexadecimal
2013
```

4.2 Type float (flottant ou à virgule flottante)

Une donnée de type **float** ou **réelle** est notée avec un point décimal ou en notation exponentielle :

Script PYTHON : Flottants

```
>>> 4.215
4.215

>>> .0087
0.0087

>>> 8e9
8000000000.0

>>> 1.025e38
1.025e+38
```

Les flottants supportent les mêmes opérations que les entiers. L'importation du module **math** permet l'utilisation de fonctions mathématiques usuelles.

Script PYTHON : Fonctions mathématiques usuelles

```
>>> import math # importation du module math
>>> dir(math) # la commande dir liste le contenu du repertoire math ou on y trouve
les fonctions mathematiques elementaires ['__doc__', '__name__', '__package__',
acos', 'acosh', 'asin', 'asinh', 'atan', 'atan2', 'atanh', 'ceil', 'copysign', 'cos',
'cosh', 'degrees', 'e', 'erf', 'erfc', 'exp', 'expm1', 'fabs', 'factorial', '
floor', 'fmod', 'frexp', 'fsum', 'gamma', 'hypot', 'isfinite', 'isinf', 'isnan',
'ldexp', 'lgamma', 'log', 'log10', 'loglp', 'modf', 'pi', 'pow', 'radians', '
sin', 'sinh', 'sqrt', 'tan', 'tanh', 'trunc']

>>> math.sin(math.pi/4) # sin(pi/4) On verra plus loin (chapitre Modules et
Packages) qu'il est possible de faire plus simple.
0.7071067811865475

>>> math.degrees(math.pi) # pi en degres 180.0

>>> math.sqrt(2) # racine carree de 2
1.4142125623730951
```

4.3 Type bool (booléen)

Les données du type bool ne présentent que deux valeurs : False et True. Les opérations logiques et comparaison sont évaluées et le résultat est un booléen.

Script PYTHON : Opérateurs de comparaison

```
>>> 2 < 8 # strictement inferieur
True
>>> 2 <= 8 # inferieur ou egal
True
>>> 2 == 8 # egal
False
>>> 2 > 8 # strictement superieur
False
>>> 2 >= 8 # superieur ou gal
False >>> 2 != 8 # different
True
```

Script PYTHON : Opérateurs logiques

```
>>> (3 == 3) or (9 > 24) # OU logique
True
>>> (9 > 24) and (3 == 3) # ET logique
False
>>> not(3 == 3) # NON logique
False
```

4.4 Type str (string ou chaîne de caractères)

Une donnée de type **str** représente une séquence constituée de caractères.

Script PYTHON : Représentation d'une chaîne de caractères

```
>>> "Dupont" # utilisation des guillemets pour les chaines de caracteres
'Dupont'

>>> 'Pierre' # utilisation des apostrophes est egalement possible
'Pierre'
```

Pour une chaîne de caractères avec apostrophes, il faut utiliser la séquence d'échappement \.

Script PYTHON : Séquence d'échappement \

```
>>> "Aujourd'hui" # apostrophe entre d et hui mal interpretee
File "<interactive input>", line 1
"Aujourd'hui"

SyntaxError: invalid syntax
>>> "Aujourd\'hui" # utilisation de la sequence d echappement
"Aujourd'hui"
```

Pour un saut à la ligne il faut utiliser la séquence d'échappement ou la forme multi-lignes avec triples guillemets.

Script PYTHON : Saut à la ligne

```
>>> chaine = 'Dupont\nPierre' # sequence d echappement \n
>>> print(chaine)
Dupont
Pierre
>>> chaine = """Dupont
... Pierre""" # Forme multi-lignes
>>> print(chaine)
Dupont Pierre
```

Opérations sur les chaînes de caractères :

Script PYTHON : Opérations sur les chaînes de caractères

```
>>> 'Dupont'+ ' '+ 'Pierre' # concatenation de i de caracteres
'Dupont Pierre'
>>> chaine = 'Dupont Pierre'
>>> len(chaine) # longueur d une chaine de caracteres
13
>>> chaine = 'Ha ! '
>>> chaine * 3 # repetition
>>> print(chaine)
'Ha ! Ha ! Ha !'
```

Script PYTHON : Indexage

```
>>> chaine = 'Dupont Pierre'
>>> print(chaine[0]) # premier caractere D a l index 0
>>> print(chaine[-1]) # dernier caractere e
>>> print(chaine[2:6]) # du 3ieme au 6ieme caractere. Le 7ieme (index 6) est exclus
pont
```

Il n'est pas possible de réaliser des opérations arithmétiques sur des chaînes de caractères. La fonction `float()` permet de convertir un type `str` en type `float` et la fonction `int()` permet de convertir un type `str` en type `int`.

Script PYTHON : fonctions int() et float()

```
>>> '17.5' # il s agit d une chaine de caracteres
'17.5'
>>> float('17.5') # conversion de la chaine de caracteres en flottant
17.5
>>> int('17.5') #operation impossible
Traceback (most recent call last):
  File "<pysshell>", line 1, in <module>
ValueError: invalid literal for int() with base 10: '17.5'
>>> int(float('17.5')) # utilisation de la fonction float() encapsulee dans int()
17
>>> int(3.7) # instruction int() tronque la partie decimale
3
>>> '17.45' + 2 # operation impossible car '17.5' est une chaine de caractere et 2
un nombre
Traceback (most recent call last):
File "<interactive input>", line 1, in <module> TypeError: Can not convert 'int'
object to str implicitly

>>>float('17.5')+2 # utilisation de float()
19.5
```

4.5 Type list (liste)

Une liste est une **structure de données**. Le premier élément d'une liste possède l'**indice 0**(ou l'**index 0**). Une liste peut être constituée d'éléments types différents.

Script PYTHON : type list

```
>>> Donnees = ['Dupont','Pierre',17,1.75,72.5] # liste constituee de str, int et
float
>>> print(Donnees) ['Dupont','Pierre',17,1.75,72.5]
>>> print('Nom : ',Donnees[0]) # premier element indice 0
Nom : Dupont
>>> print('Age : ',Donnees[2]) # troisieme element indice 2
Age : 17
>>> print('Taille : ',Donnees[3]) # quatrieme element indice 3
Taille : 1.75
```

Il est possible de créer des listes à 2 dimensions (équivalentes à des tableaux).

Script PYTHON : liste à 2 dimensions

```
>>> liste = [[0,1,2],[3,4,5],[6,7,8]] # liste a 2 dimensions
>>> print(liste)
[[0,1,2],[3,4,5],[6,7,8]]
>>> print(liste[0]) # elements de la 1ere ligne
[0,1,2]
>>> print(liste[1][2]) # elements de la 2nde ligne et 3ieme colonne
5
```

4.6 Type dict (dictionnaire)

Un dictionnaire permet de stocker des données sous la forme `(float('17.5')+2; float('17.5')+2)`. Une clé est unique et n'est pas nécessairement un entier.

Script PYTHON : type dict

```
>>> moyenne = {'Math':14,'Anglais':12.5,'Français':13}
>>> print(moyenne) # tout le dictionnaire
{'Anglais':12.5,'Math':14,'Français':13}
>>> print(moyenne['Math']) # la valeur qui a pour cle  Math
14
>>> moyenne['Anglais'] = 16 # nouvelle affectation
>>> print(moyenne) # tout le dictionnaire
{'Math': 14, 'Anglais': 16, 'Français': 13}
```

4.7 Autres types

Il en existe bien d'autres types :

- ▶ long : nombres entiers de longueur quelconque (4284961775562012536954159102);
- ▶ complex : nombres complexes (1 + 2.5 j);
- ▶ tuple : structure de données;
- ▶ file : fichiers;
- ▶ ...

4.8 Variables

Une variable est un espace mémoire dans lequel il est possible de stocker une valeur (une donnée). Il s'agit donc d'un identifiant associé à une valeur. La notion de variable n'existe pas dans le langage Python.

On affecte une variable par une valeur en utilisant le signe `=`. Dans une affectation, le membre de gauche reçoit le membre de droite.

Script PYTHON : affectations simples de variables

```
>>> a = 2 # la variable a recoit la valeur 2
>>> b = math.sqrt(2) # la variable b recoit la valeur racine carree 2
>>> c = a*b # la variable c recoit la valeur de a fois la valeur de b
>>> print(c)
2.8284271247461903
```

Outre l'affectation simple, on peut aussi utiliser les formes suivantes :

Script PYTHON : autres formes d'affectations de variables

```
>>> a = 3 # affectation simple. On stocke 3 dans la variable a.
>>> print(a)
3
>>> a =a+ 3 # affectation augmentee (notation equivalente a += 3)
>>> print(a)
6
>>> a = b = 7 # affectations multiples
>>> print(a)
7
>>> print(b)
7
>>> a,b = 2.7,5.1 # affectation parallele de sequences
>>> print(a)
2.7
>>> print(b)
5.1
>>> a,b,c = ['A','B','C'] # affectation parallele de sequences : liste
>>> print(a)
A
>>> print(b)
B
>>> print(c)
C
```

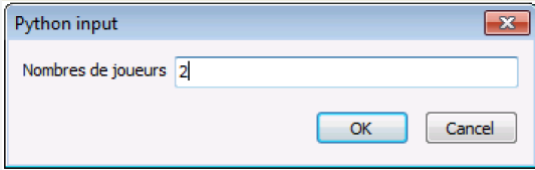
5 Entrées / Sorties

5.1 Fonction INPUT()

La fonction standard `input()` interrompt le programme et attend que l'utilisateur entre une donnée et la valide.

Script PYTHON : fonction input()

```
>>> nb_joueurs = input("Nombres de joueurs") # nb_joueurs est une chaine de
caracteres
```



```
>>> print(nb_joueurs)
2
>>> nb_joueurs = float(input("Nombres de joueurs")) # nb_joueurs est transtypé en
flottant
>>> print(nb_joueurs)
2.0
```

5.2 fonction PRINT()

La fonction print() est indispensable pour l'affichage des résultats.

Script PYTHON : fonction print()

```
>>> a,b = 2,5
>>> print(a,b)
2 5

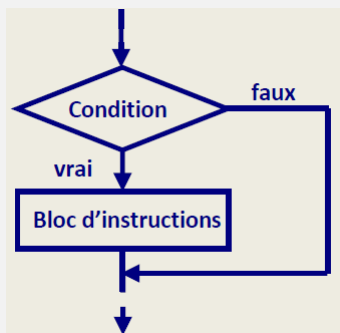
>>> print("Somme = ", a + b)
Somme = 7

>>> print("Le produit de ",a," par ",b," vaut : ",a * b)
Le produit de 2 par 7 vaut : 10
>>> print() # cree un saut de ligne (meme effet que print("\n"))
```

6 Structures alternatives

6.1 Instruction IF (SI)

Algorithme et algorithme : IF (SI)



```
if Condition :
    Instructions
Suite du programme
```

Si la condition est vraie (True) alors le bloc d'instructions est exécuté. Si la condition est fausse (False) on passe directement à la suite du programme.

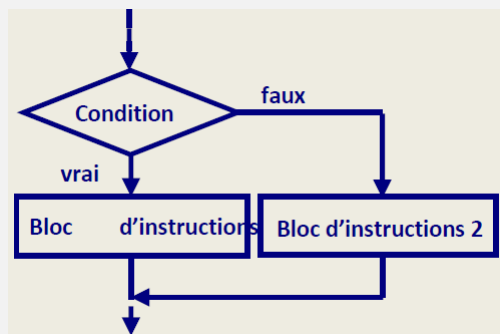
Script PYTHON : INSTRUCTION IF

```
nb = input("Entrer un nombre plus petit que 100 : ")
nb = float(nb) #float() ou int() est necessaire car la commande "input" renvoie une
               chaine de caracteres
if nb < 100 :
    print("Le nombre",nb,"convient")
>>> Le nombre 50.0 convient
```

6.2 Instruction ELSE (SINON)

Une instruction **else** est toujours associée à une instruction **if**.

Algorithme et algorithme : ELSE (SINON)



```
if Expression :  
    Bloc Instructions 1  
else :  
    Bloc Instructions 2  
Suite du programme
```

Script PYTHON : INSTRUCTION ELSE

```
nb = input("Entrer un nombre plus petit que 100")  
nb = float(nb)  
if nb < 100 :  
    print("Le nombre", nb, "convient")  
else :  
    print("Le nombre", nb, "est trop grand")  
>>>  
Le nombre 20.0 convient  
>>>  
Le nombre 120.0 est trop grand
```

6.3 Instruction ELIF (SINON SI)

Dans le cas de structures alternatives imbriquées, il est possible d'utiliser une instruction **elif** (contraction de **else if**).

Algorithme : ELIF (SINON SI)

```
if Condition 1 :  
    Bloc Instructions 1  
elif Condition 2 :  
    Bloc Instructions 2  
else :  
    Bloc Instructions 3  
Suite du programme
```

Script PYTHON : INSTRUCTION ELIF

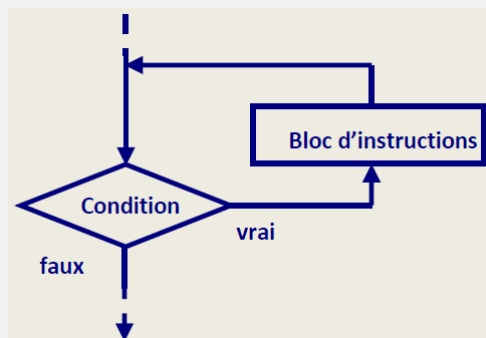
```
nb = input("Entrer un nombre plus petit que 100")
nb = float(nb)
if nb == 100 :
    print("Ce nombre vaut 100")
elif nb == 0 :
    print("Ce nombre est nul")
elif nb > 0 and nb < 100 :
    print("Le nombre",nb,"convient")
else :
    print("Le nombre",nb,"est trop grand")
>>>
Ce nombre vaut 100
>>>
Ce nombre est nul
>>>
Le nombre 20.0 convient
>>>
Le nombre 200.0 est trop grand
```

7 Structures répétitives

Une structure répétitive ou boucle permet de répéter une portion de code.

7.1 Boucle WHILE (TANT QUE)

Algorithme et algorithme : ELSE (SINON)



```
while Condition :
    Bloc Instructions
Suite du programme
```

Tant que la condition est vraie (True) le bloc d'instructions est exécuté. Le cycle continue jusqu'à ce que la condition soit fautive (False) : on passe alors à la suite du programme.

Script PYTHON : table de multiplication par 8 avec la boucle WHILE

```
print("Ta table de multiplication par 8.. jusque 5*8")
compteur = 1 # initialisation de la variable de comptage
while compteur <= 5 : # ce bloc est execute tant que la condition (compteur<=5) est
    vraie
    print (compteur, "* 8 =", compteur*8)
    compteur += 1 # incrementation du compteur : compteur = compteur + 1
# on sort de la boucle
print("Eh voila")
>>>
Table de multiplication par 8
1 * 8 = 8
2 * 8 = 16
3 * 8 = 24
4 * 8 = 32
5 * 8 = 40
Eh voila
```

Script PYTHON : affichage de l'heure courante avec la boucle WHILE

```
import time # importation du module time
quitter = 'n' # initialisation de la reponse
while quitter != 'o' : # ce bloc est execute tant que la condition (quitter != 'o')
    est vraie
    print ("Heure courante",time.strftime('%H:%M:%S'))
    quitter = input("Voulez-vous quitter le programme (o/n) ?") # on sort de la
    boucle
print("A bientôt")
>>>
Heure courante 17:36:02
Voulez-vous quitter le programme (o/n) ?n
Heure courante 17:36:08
Voulez-vous quitter le programme (o/n) ?o
A bientôt
```

7.2 Boucle FOR...IN RANGE (Boucle POUR)

Algorithme : FOR ... IN RANGE (POUR)

```
for element in sequence :
    Bloc Instructions
Suite du programme
```

La séquence est parcourue élément par élément. L'élément peut être de tout type : entier, caractère, élément d'une liste...

L'utilisation de la boucle for est intéressante si le nombre de boucles à effectuer est connu à l'avance.

Script PYTHON : table de multiplication par 9 avec boucle FOR...IN RANGE

```
print("Table de multiplication par 9 jusque 5*9")
for compteur in range(1,6) :
    print(compteur, "* 9 =", compteur*9) # on sort de la boucle
print("Et voila")
```

La valeur initiale de l'élément compteur est égale à 1. On exécute la boucle tant que l'élément compteur est inférieur à 6.

```
>>>
Table de multiplication par 9
1 * 9 = 9
2 * 9 = 18
3 * 9 = 27
4 * 9 = 36
5 * 9 = 45
Et voila
```

Script PYTHON : parcourt d'une chaîne de caractères avec boucle FOR...IN RANGE

```
chaine = "Python"
for lettre in chaine : # lettre est la variable d iteration
    print(lettre) # on sort de la boucle
print("Fin de la boucle")
```

La variable lettre est initialisée avec le premier élément de la séquence ('P'). Le bloc d'instructions est alors exécuté. Puis la variable lettre est mise à jour avec le second élément de la séquence ('y') et le bloc d'instructions à nouveau exécuté... La boucle est exécutée jusqu'à ce on arrive au dernier élément de la séquence ('n').

```
>>>
P
y
t
h
o
n
Fin de la boucle
```

Script PYTHON : parcourt d'une liste avec boucle FOR...IN RANGE

```
liste = ["Pierre", "Dupont", 67.5, 17]
for element in liste : # element est la variable d iteration
    print(element)
# on sort de la boucle
print("Fin de la boucle")
```

La variable liste est initialisée avec le premier élément de la séquence ('Pierre'). La boucle est exécutée jusqu'à ce on arrive au dernier élément de la séquence (17).

```
>>>
Pierre Dupont
67.5
17
Fin de la boucle
```

7.3 Instruction BREAK pour sortir d'une boucle infinie

L'instruction `break` provoque une sortie immédiate d'une boucle `while` ou d'une boucle `for`.

Script PYTHON : instruction BREAK

```
import time # importation du module time
while True : # l'expression est toujours vraie
    print("Heure courante",time.strftime('%H:%M:%S'))
    quitter = input("Voulez-vous quitter le programme (o/n) ?")
    if quitter = 'o' :
        break # on sort de la boucle
print("A bientôt")
```

L'expression `True` est toujours vraie : il s'agit d'une boucle sans fin. L'instruction `break` est donc le seul moyen de sortir de la boucle.

```
>>>
Heure courante 09:04:02
A bientôt
```

8 Fonctions

8.1 Utilité des fonctions

Une fonction est une portion de code (sorte de sous-programme) que l'on peut appeler au besoin.

L'utilisation des fonctions permet :

- ▶ d'éviter la répétition ;
- ▶ de mettre en relief les données et les résultats : entrées et sorties de la fonction ;
- ▶ la réutilisation dans d'autres scripts par l'intermédiaire du mécanisme de l'import ;
- ▶ de décomposer une tâche complexe en tâches plus simples.

On obtient ainsi des programmes plus courts et plus lisibles.

8.2 Syntaxe

Script PYTHON : Instruction DEF

```
def nomFonction(parametres1,parametre2,parametre3):
    # Documentation de la fonction.
    bloc_instructions
    return resultat
```

Script PYTHON : fonction « Conversion degrés Celsius en degrés Kelvin »

```
def conv_celsius_kelvin(degrees_celsius) :
    # cette fonction permet de convertir des
    # degrés Celsius en Kelvin
    kelvin = degrees_celsius + 273
    return kelvin
>>> conv_celsius_kelvin(0)
273
>>> conv_celsius_kelvin(-273)
0
>>> conv_celsius_kelvin(30)
303
```

8.3 Passage de paramètres à la fonction

Le passage de paramètres permet de fournir les données utiles à la fonction. Ce passage s'effectue lors de l'appel de la fonction. Il est possible de fournir plusieurs paramètres à la fonction. Dans l'exemple précédant, il faut fournir le paramètre « `degrees_celsius` » à la fonction pour son exécution.

Script PYTHON : fonction « Portion de table de multiplication quelconque »

```
def Table_Mul(table,debut,fin) :
    # -----
    # cette fonction permet d'afficher une portion
    # d'une table de multiplication quelconque
    # table : table de multiplication attendue
    # debut : a partir de quelle valeur
    # fin : jusque quelle valeur
    # -----
    n = debut
    while n <= fin :
        print(n,"*",table,"=",n*table)
        n = n + 1
    # programme principal
    num_table = int(input("Quelle table voulez-vous : "))
    num_debut = int(input("A partir de quelle valeur : "))
    num_fin = int(input("Jusque quelle valeur : "))
    print("Table de multiplication par",num_table,"de",num_debut,"a",num_fin)
    Table_Mul(num_table,num_debut,num_fin) # appel de la fonction avec passage des
    parametres

>>>
Table de multiplication par 8 de 5 a 7
5 * 8 = 40
6 * 8 = 48
7 * 8 = 56
```

Dans l'exemple ci-dessus, il faut fournir les paramètres « `table` », « `debut` » et « `fin` » à la fonction « `Table_Mul` ». Par contre le corps d'instruction de la fonction « `Table_Mul` » ne contient pas de `return`, c'est-à-dire qu'elle ne retourne pas de résultat. Il s'agit d'une procédure. Les paramètres passés en arguments peuvent être de types simples (`int`, `float`, `str`...) mais également de types plus complexes (`tuple`, `list`, `dict`...). Il est également possible de passer en argument d'autres fonctions.

8.4 Sortir des résultats d'une fonction

L'instruction **return** stoppe l'exécution de la fonction et retourne une ou plusieurs données.

Script PYTHON : fonction « Calcul de la surface et du volume d'une sphère »

```
import math
def surface_volume_sphere(R) :
    # -----
    # cette fonction calcule et retourne a
    # partir du rayon R, la surface S et le
    # volume V d une sphere
    # -----
    S = 4.0 * math.pi * R**2 # calcul de la surface de la sphere
    V = S * R / 3 # calcul du volume de la sphere
    return S,V # retourne les resultats surface et volume

# programme principal
rayon = float (input("Rayon (en cm):"))
s,v = surface_volume_sphere(rayon)
print("Sphere de rayon",rayon,"cm")
print("Sphere de surface",s,"cm2")
print("Sphere de volume",v,"cm3")

>>>
Sphere de rayon 25.0 cm
Sphere de surface 7853.981633974483 cm2
Sphere de volume 65449.84694978735 cm3
```

8.5 Portées des variables : variable LOCALES et variable GLOBALES

La portée d'une variable dépend de l'endroit du programme où on peut accéder à la variable. Une variable **globale** est visible et utilisable dans tout le programme. Une variable **locale** est créée par une fonction et n'est visible que par cette fonction. **Lors de la sortie de la fonction, la variable est détruite.**

Script PYTHON : variable globale et variable locale

```
x = 10 # variable globale
def ma_fonction() :
    x = 20 # variable locale
    print("La variable locale est",x)

# programme principal
print("La variable globale est",x)
ma_fonction()
>>>
La variable globale est 10 La variable locale est 20
```

Bien que possédant, le même identifiant, les deux variables x sont distinctes.

9 Modules et packages

9.1 Modules et packages

Un programme Python est généralement composé de plusieurs fichiers sources, appelés modules. Ces fichiers ont également pour extension `.py`.

Ces modules doivent être indépendants les uns des autres pour être réutilisés à la demande dans d'autres programmes.

Il est possible d'importer tout un module :

Script PYTHON : Importation du module MATH

```
import math
```

Il est également possible d'importer quelques fonctions d'un module :

Script PYTHON : Importation des fonctions PI - SIN - SQRT et DEGREES du module MATH

```
>>> from math import pi, sin, sqrt, degrees #importation de pi, sin, sqrt et
      degrees evite de saisir math.pi, math.sin(...) math.degrees(...) math.sqrt(...)

>>> sin(pi/4)
0.7071067811865475
>>> degrees(pi/4)
45.0
>>> sqrt(16)
4.0
```

Lors de l'importation de modules, il est conseillé de respecter l'ordre d'importation suivant :

- ▶ modules de la bibliothèque standard ;
- ▶ modules des bibliothèques tierces ;
- ▶ modules personnels.

Un package permet de grouper plusieurs modules. Les modules d'un package peuvent être des sous-packages, ce qui donne une structure arborescente. En résumé, un package est simplement un répertoire qui contient des modules et un fichier `__init__.py` décrivant l'arborescence du package.

9.2 Bibliothèque standard

7.2 – BIBLIOTHEQUE STANDARD La bibliothèque standard contient de plus de 200 packages et modules répondant aux problèmes courants les plus variés.

Parmi les différents modules, on peut citer les fonctionnalités suivantes :

- ▶ le module `textwrap` est utilisé pour formater un texte : longueur de chaque ligne, contrôle de l'indentation ;
- ▶ le module `struct` permet de convertir des nombres, booléens et des chaînes en leur représentation binaire ;
- ▶ le module `io.StringIO` permet la gestion des fichiers ;
- ▶ les modules mathématiques : `math`, `fraction`, `decimal`, `random` ;
- ▶ les modules de gestion du temps : `calendar`, `time` et `datetime` ;
- ▶ ...

Ne pas hésiter à vous rendre sur le site [w3schools](http://w3schools.com) pour des précisions sur le langage ainsi que demander à une intelligence artificielle de vous aider.