
- Langages et Programmation -

LES PREMIERS PAS AVEC PYTHON



Plan du chapitre

I. PRÉSENTATION DE PYTHON

Bref historique

Python : un langage de programmation interprété

Installation de Python

L'interface de l' IDE Spyder

L'édition et l'exécution d'un premier programme

II. LA DÉCLARATION ET L'AFFECTATION DU CONTENU DE VARIABLES

La déclaration d'une variable

Quel intérêt d'utiliser une variable ?

Typage des variables

Affectations (assignments) multiples

III. EFFECTUER DES CALCULS AVEC PYTHON

Les opérateurs arithmétiques

Quelques fonctions mathématiques

IV. CONCATÉNATION DE CHAINES DE CARACTÈRES

V. EXERCICES

I. PRÉSENTATION DE PYTHON

Bref historique

Python est un langage de programmation, dont la première version est sortie en 1991. Créé par Guido van Rossum, il a voyagé du Macintosh de son créateur, qui travaillait à cette époque au Centrum voor Wiskunde en Informatica aux Pays-Bas, jusqu'à se voir associer une organisation à but non lucratif particulièrement dévouée, la Python Software Foundation, créée en 2001. Ce langage a été baptisé ainsi en hommage à la troupe de comiques les « Monty Python ».

Python est un langage puissant, à la fois facile à apprendre et riche en possibilités. Dès l'instant où vous l'installez sur votre ordinateur, vous disposez de nombreuses fonctionnalités intégrées au langage.

Il est, en outre, très facile d'étendre les fonctionnalités existantes. Ainsi, il existe ce qu'on appelle des bibliothèques qui aident le développeur à travailler sur des projets particuliers. Plusieurs bibliothèques peuvent ainsi être installées pour, par exemple, développer des interfaces graphiques en Python.



[Retour au plan](#)

Python : un langage de programmation interprété

Python est un langage de programmation interprété, c'est-à-dire que les instructions que vous lui envoyez sont « transcrites » en langage machine au fur et à mesure de leur lecture. D'autres langages (comme le C / C++) sont appelés "langages compilés" car, avant de pouvoir les exécuter, un logiciel spécialisé se charge de transformer le code du programme en langage machine. On appelle cette étape la "compilation". À chaque modification du code, il faut rappeler une étape de compilation. Les avantages d'un langage interprété sont la simplicité (on ne passe pas par une étape de compilation avant d'exécuter son programme). En contrepartie, un langage compilé se révélera bien plus rapide qu'un langage interprété (la traduction à la volée de votre programme ralentit l'exécution), bien que cette différence tende à se faire de moins en moins sentir au fil des améliorations.

[Retour au plan](#)

Installation de Python

Il est possible d'installer directement python, mais il vous faudra aussi installer un IDE : environnement de développement intégré (*Integrated Development Environment*) et également les bibliothèques qui vous seront utiles, en prenant bien soin d'installer des versions compatibles. Cela peut s'avérer fastidieux !

Il existe des distributions Python clé en main, c'est-à-dire un package comprenant Python, un ou plusieurs IDE et un certain nombre de bibliothèques les plus courantes...

Nous utiliserons WinPython 3.7.1, téléchargeable à l'adresse : <https://winpython.github.io/>.

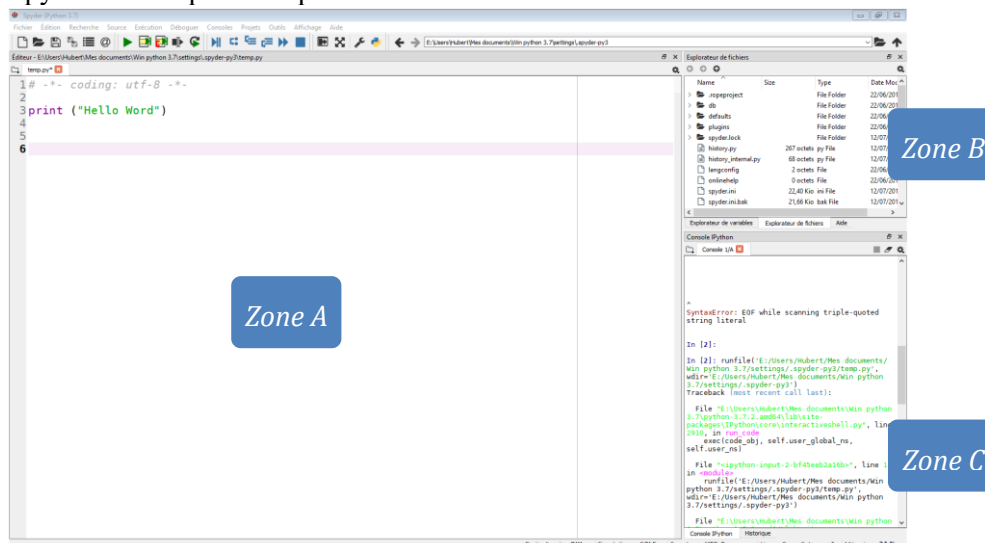
WinPython peut être installé en version portable, c'est-à-dire qu'il peut fonctionner sur une simple clé USB (l'ensemble des fichiers est dans un répertoire facilement accessible). WinPython comprend notamment la version 3.4 de Python, les IDE Spyder et Pyzo, et l'application Web Jupyter Lab.



[Retour au plan](#)

L'interface de l' IDE Spyder

La fenêtre de l'éditeur Spyder est composée de plusieurs zones :



A : Zone de l'éditeur, zone de saisie du programme

B : Zone de l'explorateur de fichiers, de variables ou zone d'aide

C : Zone où le programme s'exécute, cette zone s'appelle aussi la console Python

[Retour au plan](#)

L'édition et l'exécution d'un premier programme

Comme la coutume l'exige, notre premier programme sera l'affichage du texte "Hello World". Pour cela :

- Dans la barre d'outils de l'IDE, cliquer sur "nouveau Fichier..." ;
- Dans la fenêtre de l'éditeur, taper l'instruction :

```
1 print ( " Hello_World " )      # affiche le texte entre guillemets
```

- Cliquer dans la barre d'outils sur le triangle vert qui signifie *Exécuter* (ou touche F5) ;



- Avant d'exécuter le programme, Spyder demande de l'enregistrer. Créer un dossier "PgmsPython" dans le répertoire de votre choix et enregistrez le programme sous le nom hello.py (vous remarquerez que l'extension .py est celle réservée à python).

Si tout va bien, votre programme s'exécute dans la console.

```
Console IPython
Console 1/A
In [11]: runfile('E:/Users/Hubert/Mes documents/LCDG/NSI/Hello.py',
wdir='E:/Users/Hubert/Mes documents/LCDG/NSI')
Hello_World
```

Rmq : Il est possible (et fortement conseillé) d'inclure des commentaires dans les lignes de code, ils doivent être précédés de # afin qu'ils ne soient pas interprétés, ce qui provoquerait un message d'erreur.

[Retour au plan](#)

II. LA DÉCLARATION ET L'AFFECTATION DU CONTENU DE VARIABLES

La déclaration d'une variable

Tapez maintenant l'instruction :

```
1 print (texte)      # sans utiliser les guillemets
```

Alors que le premier programme fonctionnait parfaitement, l'interpréteur affiche maintenant le message d'erreur suivant :

```
File "E:/Users/Hubert/Mes documents/LCDG/NSI/Hello.py", line 1, in
<module>
    print (texte)      # sans utiliser les guillemets
NameError: name 'texte' is not defined
```

Le message d'erreur vient du fait que l'interpréteur cherche à afficher à l'écran non plus le texte "texte" mais **le contenu de la variable texte**. On peut considérer une variable comme une boîte (en fait une case mémoire de la RAM : Random Access Memory – mémoire vive) qui porte un nom (ici texte) et dans laquelle on va stocker un contenu. L'erreur vient du fait que cette variable n'a pas été définie.

Pour définir cette variable, on rajoute la ligne suivante en début de programme :

```
1 texte = 'Hello World'           # ou encore texte = "Hello Word"
2 print (texte)                   # affiche le contenu de la variable texte
3
```

Nous venons de **déclarer** une variable nommée `texte` et de lui **affecter** (ou **assigner**) le texte "Hello World". En python comme dans bien d'autres langages, l'opération d'affectation est représentée par le signe égal.

[Retour au plan](#)

Quel intérêt d'utiliser une variable ?

Prenons la définition du mot ordinateur d'après le dictionnaire *Le Petit Larousse* : « *Machine automatique de traitement de l'information, obéissant à des programmes formés par des suites d'opérations arithmétiques et logiques* ». Qui dit traitement de l'information, dit donc données à manipuler. Or les données sont amenées à changer !

Imaginons un programme qui vous demande votre nom et l'affiche :

```
1 input ("Bonjour, quel est ton nom ?")           # input() est une fonction qui demande de saisir du texte au clavier
2 input("Et quel est ton prénom ?")
3 print ("J'ai bien compris que tu t'appelles Jérémy Guillaume ?")
```

Ce programme fonctionne-t-il comme vous le souhaitez ? Qu'est-ce qui, selon vous, pose un problème ?

Essayez maintenant le programme suivant :

```
1 nom = input ("Bonjour, quel est ton nom ?")
2 prenom = input("Et quel est ton prénom ?")
3 print ("J'ai bien compris que tu t'appelles ", prenom, " ", nom)
```

Rmq : Avec Python, lorsque l'on veut écrire une chaîne de caractères, on la met entre des apostrophes (' ') ou entre des guillemets (" "). Dans la 3^{ème} ligne de code, il est impératif d'utiliser des guillemets. Pourquoi ? Maintenant, votre programme fonctionne parfaitement !

[Retour au plan](#)

Typage des variables

Dans certains langages de programmation (java par exemple), avant de déclarer une variable, il faut impérativement déclarer le type de variable que l'on va utiliser. Les différents types de variables sont les suivants :

- **int** qui désigne les nombres entiers (int est l'abréviation de "integer number", nombre entier) ;
- **float** qui désigne les nombres réels, ou nombres à virgule flottante (**f**loating point number) ;
- **bool** qui désigne un booléen, c'est-à-dire une variable logique, son contenu ne peut donc être que True (vrai) et False (faux)
- **str** qui désigne les chaînes de caractères ;

Python est un langage à **typage dynamique**, cela signifie qu'en fonction du contenu affecté à la variable, il détermine automatiquement le type de la variable.

L'*explorateur de variables* permet d'afficher le nom de la variable, son type, sa taille et sa valeur.

Nom	type	taille	Valeur
nom	str	1	PEREAU
prenom	str	1	Vincent


L'instruction *type* permet également d'afficher le type de la variable dans la console :

```
1 A = "Ceci est du texte"
2 print(type(A))
3
4 B = 4
5 print(type(B))
6
7 C="4"
8 print(type(C))
9
10 D = 8.2          # attention, pour les nombres décimaux on utilise le point et non pas la virgule !
11 print(type(D))
12
13 E = 4.5648e-12   # e pour exposant
14 print(type(E))
```

Vous remarquerez dans l'explorateur de variable que les variables *nom* et *prenom* utilisées précédemment sont toujours présentes (et donc les cases mémoires inutilement occupées) après l'exécution du programme.

Une bonne pratique consiste à effacer les variables en fin de programme si elles ne sont pas utilisées par la suite afin de libérer la mémoire. Pour cela il suffit de rajouter l'instruction :

```
16 del A,B,C,D,E
```

Remarque : Spyder propose un outil *gomme* dans la console Python afin d'effacer la totalité des variables. 

Le type d'une variable peut être modifié de manière dynamique en cours de programme. Le vérifier en testant le programme suivant :

```
1 A = 8
2 print(type(A))
3 A = A + 1.5
4 print(type(A))
```

L'explorateur de variable affiche uniquement les dernières valeurs et types pris par les variables en fin de programme. Pour observer le changement de typage de la variable en cours d'exécution, se positionner à la ligne de code n°1 et appuyer sur la touche F9 (ou menu Exécution/Exécuter la sélection ou la ligne courante). Le programme s'exécute alors ligne par ligne à chaque appui de la touche F9.

[Retour au plan](#)

Affectations (assignments) multiples

Sous Python, on peut assigner une valeur a plusieurs variables simultanément.

```
1 x=y=7
2 print ('x a pour valeur ', x)
3 print ('y a pour valeur ', y)
```

On peut aussi effectuer des *affectations parallèles* a l'aide d'un seul operateur = :

```
1 x,y=7,9.4
2 print ('x a pour valeur ', x)
3 print ('y a pour valeur ', y)
```

Les variables x et y prennent simultanément les nouvelles valeurs 7 et 9,4

[Retour au plan](#)

III. EFFECTUER DES CALCULS AVEC PYTHON

Les opérateurs arithmétiques

Un ordinateur est bien évidemment capable d'effectuer des opérations arithmétiques et mathématiques. Dans le tableau ci-après, sont présentés les symboles utilisés pour les opérations de base.

Opération	Symbole	Exemples à tester
Addition	+	3+5 donne 8
Soustraction	-	7-2 donne 5
Multiplication	*	7*8 donne 56
Division	/	7/2 donne 3.5
Reste de la division	%	7%3 donne 1
Quotient de la division	//	7//2 donne 3
Puissance	**	10**3 donne 1000

Les tests peuvent s'effectuer de différentes manières :

- Depuis l'éditeur de texte en utilisant par exemple les commandes suivantes

```
1 a=2
2 b=8
3 c = a**b # calcul a à la puissance b
4 print (c)
5
6 # ou plus simplement sans utiliser de variables
7 print (2**8)
```

- Depuis la console, qui peut s'utiliser comme une calculatrice, en utilisant des variables ou pas (dans la console le print n'est pas obligatoire)

```
In [1]: a=2
In [2]: b=8
In [3]: c=a**b
In [4]: c
Out[4]: 256
In [5]: # ou plus simplement
In [6]: 2**8
Out[6]: 256
In [7]: |
```

Prévoir le résultat du programme suivant et le tester pour vérifier vos prévisions

```
1 a = 90
2 a= a+10
3 a=a**3
4 print(a)
```

Détails du programme :

Ligne	Commentaires	Valeur de la variable a
1	La variable a est créée et la valeur 90 lui est affectée	90
2	Il faut lire cette ligne de droite à gauche : la valeur de a, c'est-à-dire 90 est additionnée à la valeur 10. Le résultat : 100 est affecté à la variable a	100
3	Le cube de a est calculé : $100^3 = 1000000$. Le résultat est affecté à la variable a	1000000
4	Affichage de a	

[Retour au plan](#)

Quelques fonctions mathématiques

Pour utiliser certaines fonctions mathématiques plus avancées (x^a , $\exp(x)$, $\sin(x)$, $\log(x)$...) il est nécessaire d'importer des bibliothèques.

Si l'on utilise la bibliothèque *math*, la première ligne de code devra commencer par :

```
1 from math import *
```

Cette bibliothèque inclut entre autres les fonctions suivantes :

Fonction	Symbole	Exemples à tester
constante π	pi	pi donne 3.141592653589793
sinus (angle en radians)	sin(x)	sin(pi/2) donne 1.0
arc sinus	asin(x)	asin(1) donne 1.5707963267948966 radians
conversion degrés \rightarrow radians	degrees(x)	degrees(pi) donne 180.0 degrés
conversion radians \rightarrow degrés	radians(x)	radians(90) donne 1.5707963267948966 radians
logarithme décimal	log10(x)	log10(100) donne 2
logarithme népérien	log(x)	log(100) donne 4.605170185988092
exponentielle	exp(x)	exp(100) donne 2.6881171418161356e+43
puissance x^y	pow(x,y)	pow(2,8) donne 256.0
racine carrée	sqrt(x)	sqrt(100) retourne 10
arrondi au plus proche entier	round(x)	
arrondi au plus proche réel avec y décimales après la virgule	round(x,y)	round(100.235789,2) retourne 100.24
arrondi au plus petit entier $\geq x$	ceil(x)	ceil(100.235789,2) retourne 101
arrondi au plus grand entier $\leq x$	floor(x)	floor(100.235789,2) retourne 100

Si vous avez besoin d'autres fonctions mathématiques : <https://docs.python.org/3/library/math.html>

La bibliothèque *random* permet de générer des nombres aléatoires, la première ligne de code devra commencer par :

```
1 from random import *
```

Cette bibliothèque inclut entre autres les fonctions suivantes :

Fonction	Symbole	Exemples à tester
Génération nombre aléatoire entier	randint(x,y)	Retourne un entier aléatoire compris entre x et y inclus
Génération nombre aléatoire flottant	random()	Retourne un flottant aléatoire compris entre 0 et 1 exclus

[Retour au plan](#)

IV. CONCATÉNATION DE CHAINES DE CARACTÈRES

Testez le programme suivant :

```
1 mot1 = "Hello "
2 mot2 = "the "
3 phrase = mot1 + mot2 + "world"
4 print (phrase)
```

L'opérateur + ne se limite pas aux additions. Appliqué à 2 chaînes de caractères, il effectue une **concaténation**, c'est-à-dire qu'il met bout à bout les chaînes pour n'en former plus qu'une.

Testez maintenant le programme suivant :

```
1 mot1 = "j'ai "
2 mot2 = 18
3 mot3 = " ans"
4 phrase = mot1 + mot2 + mot3
5 print (phrase)
```

Vous constatez un message d'erreur :

```
File "E:/Users/Hubert/Mes documents/LCDG/NSI/essais.py", line 4, in
<module>
    phrase = mot1 + mot2 + mot3
TypeError: can only concatenate str (not "int") to str
```

Le problème vient du fait qu'une concaténation ne peut s'effectuer qu'entre 2 (ou plusieurs) chaînes de caractères. Pour remédier à ce problème, on utilise la fonction `str()` qui convertit un entier ou un flottant en chaîne de caractères :

```
1 mot1 = "j'ai "
2 mot2 = 18
3 mot3 = " ans"
4 phrase = mot1 + str(mot2) + mot3
5 print (phrase)
```

Remarque : la fonction `int()` ou `float()` effectue l'opération inverse, c'est-à-dire qu'elle convertit, si cela est possible, une chaîne de caractères en entier ou flottant.

```
1 mot1 = 17.6
2 mot2 = "2.4"
3 total = mot1 + float(mot2)
4 print (total)
```

[Retour au plan](#)



V. EXERCICES

Exercice n°1 :

Écrire un programme qui vous demande votre nom, votre prénom, votre classe, une discipline et les 5 notes obtenues dans cette discipline.

Il calculera la moyenne dans cette discipline, arrondie à l'entier supérieur et affichera le résultat sous la forme :

PEREAU Vincent – classe de 2GT7 : moyenne en MATHÉMATIQUES = 13

Exercice n°2 :

Écrire un programme qui calcul l'angle de réfraction i_2 (exprimé en degrés et avec 1 chiffre après la virgule) d'un faisceau lumineux, après avoir demandé l'angle d'incidence i_1 (en degrés) ainsi que les indices de réfractions n_1 et n_2 des milieux 1 et 2.

Donnée : Loi de Descartes relative à la réfraction : $n_1 \times \sin(i_1) = n_2 \times \sin(i_2)$

Exercice n°3 :

Écrire un programme qui calcule combien de bonbons recevra chaque élève de la classe ainsi que le nombre de bonbons restants, après avoir demandé le nombre total de bonbons et le nombre d'élèves de la classe.

Le résultat sera affiché sous la forme : *chaque élève recevra 5 bonbons et il en restera 3*

Exercice n°4 :

Le pH d'une solution aqueuse d'acide fort est donné par $\text{pH} = -\log(c)$ et est généralement exprimé avec un seul chiffre après la virgule. Écrire un programme qui demande la concentration c de la solution et calcul le pH.

Le résultat sera affiché sous la forme : *Le pH de la solution est de 2.1*

[Retour au plan](#)