Chapitre 1
Python en mode console

Leçon 01 : installer Python

Il existe aujourd'hui des milliers de logiciels différents : systèmes d'exploitation, traitements de textes, logiciels grapiques, tableurs, logiciels de jeux, etc. Aucun ordinateur, aucune tablette ne peut fonctionner sans eux ! Tous, sans exception, sont réalisés à l'aide des langages de programmation.

1. Le langage machine

Les logiciels modernes sont conçus et réalisés par des équipes de plusieurs dizaines de personnes. Pour écrire les milliers d'*instructions* qui composent un logiciel, ces équipes utilisent des *langages de programmation*.

Au début de l'histoire des ordinateurs, vers 1950, on commandait le fonctionnement d'un ordinateur à l'aide de *codes binaires*[1] transmis directement à la machine. L'ensemble de tous ces codes constituait un langage de programmation appelé *langage machine*. Cette façon de faire était longue et pénible car la moindre opération nécessitait des milliers de signes 0 et 1. Par exemple, le code « 10110000 01100001 » commandait à la machine d'écrire le nombre entier 97. Le même code binaire permettait également de coder les lettres et les mots.

■ Exemple : le mot « *Wikipédia* » s'écrit « 101 0111 110 1001 110 1011 110 1001 111 0000 110 0101 110 0100 110 1001 110 0001 » en langage machine[2].

2. Les langages d'assemblage

Après le langage machine, on a inventé des *langages d'assemblage* appelés aussi *assembleurs*. Leurs codes, c'est à dire les « mots » qu'ils emploient sont beaucoup plus lisibles que les codes binaires du langage machine. Par exemple, les instructions « movb $0x61,%al » signifient elles aussi « écrire le nombre 97 ». Bien qu'ils soient plus faciles à mémoriser et à manipuler que des suites de 1 ou de 0, ces codes restent tout de même assez difficiles à utiliser. Pour être opérationnel, un programme écrit en assembleur doit être traduit en langage machine grâce à un programme spécial appelé *compilateur*.

---

[1] Ce sont des suites de 0 et de 1. Schématiquement, le « 1 » correspond au passage d'un courant électrique, le « 0 » à l'absence de courant.

[2] Source : article *Langage machine,* encyclopédie *Wikipédia*.

3. Les langages évolués

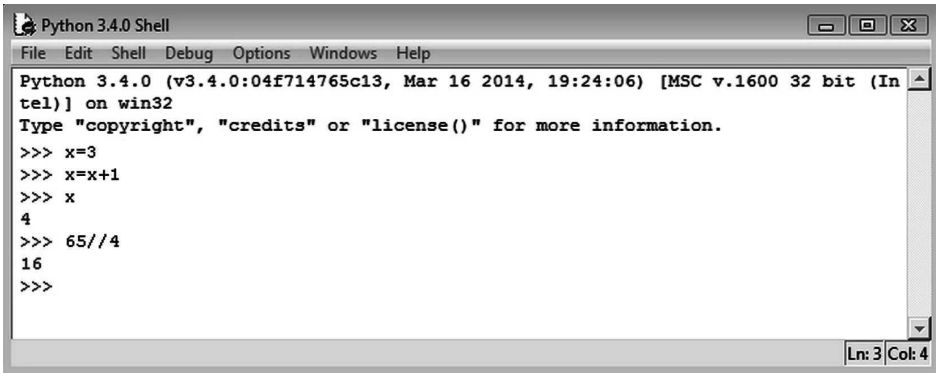
La troisième génération des langages informatiques comprend quelques centaines de langages *évolués*. Leurs *instructions* sont des mots qui appartiennent aux langues courantes, à l'anglais surtout. Un langage évolué comprend deux logiciels différents. Il y a un *éditeur de textes* qui permet de rédiger les programmes et un logiciel spécialisé qui traduit ces programmes en langage machine.
Certains langages évolués (C, C++ ou Java par exemple) sont *compilés*, ce qui signifie qu'ils sont intégralement traduits en langage machine avant d'être exécutés.
D'autres sont seulement *interprétés*. Cela signifie que les lignes d'un programme sont traduites en langage machine au fur et à mesure de leur lecture. Le langage Python[3], qui a été créé en 1990 par le néerlandais Guido van Rossom et qui fait l'objet de ce livre, est un langage interprété.

4. Comment installer Python sur l'ordinateur ?

Pour installer Python sur un ordinateur, il faut se connecter au site **https://www.python.org** et suivre la procédure indiquée pour télécharger le logiciel. Choisissez la version 3.4.0 ou la version 3.4.1. Sauf demande contraire, l'installation se fera par défaut sur le disque C: dans le répertoire **C://Python 34.**

5. Deux façons de travailler avec Python

On peut utiliser Python en *mode console* ou bien en *mode programmation*.
En mode console, c'est à dire avec **Python Shell**, on écrit les instructions à exécuter ligne par ligne derrière l'invite **>>>** et on appuie sur la touche **Entrée** à la fin de chaque ligne pour les faire exécuter.

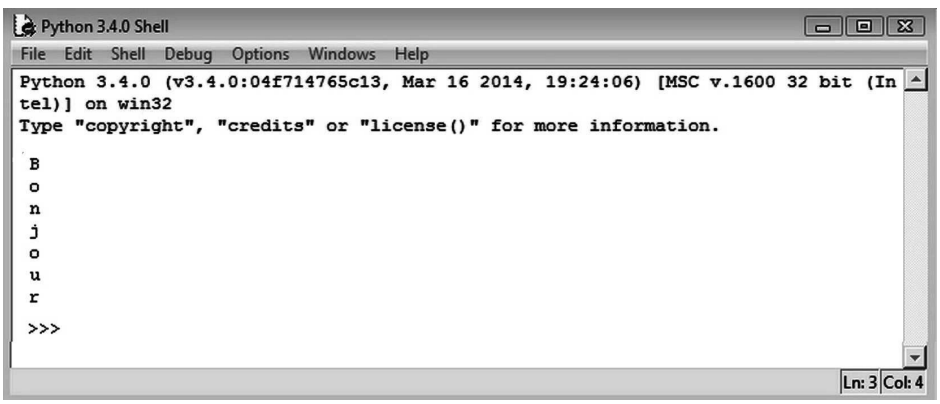


■ Remarque : la première fois que vous utiliserez Python, vous devrez taper le mot *license* derrière l'invite **>>>** et appuyer ensuite sur la touche **Entrée**.

[3] Ce nom a été choisi en hommage au groupe anglais des *Monty Python*.

En mode programmation, on peut écrire des programmes, les sauvegarder et les faire fonctionner. Pour utiliser ce mode, il faut ouvrir d'abord la fenêtre **Python Shell** puis ouvrir ensuite l'*éditeur de textes* avec la commande **New file** du menu **File**. La copie d'écran qui suit montre cet éditeur de textes avec un programme prêt à être exécuté. Ce petit programme (très court !) fait afficher les lettres du mot « bonjour *»*.

Observez la présence du menu *Run* dans la barre des menus. Ce menu sert à lancer l'exécution du programme dont le script figure dans l'éditeur de textes. Il n'existe pas dans la fenêtre **Python Shell.** La copie d'écran suivante montre l'exécution du programme précédent.



■ Remarque : observez que l'exécution d'un programme ne se fait pas dans l'éditeur de textes mais dans la fenêtre **Python Shell**.

6. Comment créer un raccourci pour démarrer Python 3.4.0 ?

Pour lancer Python plus rapidement, il est souhaitable de placer un raccourci sur le bureau. Pour cela, il faut :

- ouvrir le dossier **C:**//**Python34** ;
- ouvrir ensuite le dossier **Lib** puis, dans celui-ci, le dossier **Idlelib** ;
- sélectionner le dernier des trois fichiers nommés **Idle** et créer le raccourci avec le bouton droit de la souris (commande **créer un raccourci**). Le raccourci sera placé sur le *bureau*.

Il suffira par la suite de cliquer sur ce raccourci pour ouvrir la fenêtre **Python Shell** et la rendre disponible.

Leçon 02 : les variables numériques

En mode console, Python accepte les variables numériques. Ecrire a=3 dans un programme revient en quelque sorte à placer le nombre 3 dans un « tiroir » de la mémoire de l'ordinateur. Ce « tiroir » portera l'étiquette « a ».

1. Comment utiliser Python en mode console ?

Pour accéder à ce mode, il faut ouvrir la fenêtre de travail de **Python Shell** en cliquant sur le raccourci qui a été placé sur le bureau. Lorsqu'il fonctionne dans ce mode, Python exécute les lignes d'instructions une par une, au fur et à mesure de leur écriture. Pour chaque ligne, il faut écrire les instructions derrière l'invite **>>>**. On valide les instructions de la ligne en cours en appuyant ensuite sur la touche **Entrée**. Si aucune *erreur de syntaxe*[1] n'a été commise, les instructions sont exécutées et les résultats sont affichés sur la ligne suivante.

■ Exemple :

```
>>> 3+7
10
>>>
```

En mode console, les capacités d'édition sont très limitées. On peut corriger une ligne lorsqu'elle est en cours d'édition mais on ne peut ni modifier ni supprimer une ligne déjà éditée. En cas d'erreur, il faut retaper les instructions sur une nouvelle ligne en corrigeant les erreurs qui avaient été commises. On peut écrire plusieurs instructions sur une même ligne à condition de les séparer par des points-virgules**.**

2. Les différents types de nombres

Python 3.4.0 accepte les nombres entiers et les nombres décimaux. Pour écrire un nombre décimal, il faut utiliser le *point décimal* à la manière anglo-saxonne, jamais la virgule.

■ Exemple : tapez chacune des lignes suivantes et appuyez sur la touche **Entrée** à la fin de chaque ligne.

```
>>> 7.5/4
>>> 7,5/4
```

Python reconnaît que 7.5 est un nombre décimal et effectue la division demandée. Par contre, l'écriture 7,5 ne représente pas un nombre mais une *chaîne de caractères*, notion qui sera développée dans les leçons 04 et 12.

[1] Les erreurs de syntaxe comprennent les fautes de frappe, les oublis de caractère, les mots réservés mal orthographiés, etc.

Python accepte aussi les nombres écrits en *notation scientifique*. Ecrivez par exemple la ligne suivante et observez ce qu'affiche Python.

```
>>> 2.4e7
```

3. Les opérations

Le tableau suivant montre les symboles des opérations élémentaires acceptées par Python.

Noms des opérations	*Symboles des opérateurs*	*Exemples*
addition	+	5.7+7 donne 12.7
soustraction	-	
multiplication	*	6*7 donne 42
élévation à une puissance	**	4**2 donne 16
division décimale	/	57/5 donne 11.4
division euclidienne	//	57//5 donne le quotient euclidien 11

■ Exemple : Tapez la ligne suivante et vérifiez que le résultat est un entier de 10 chiffres.

```
>>> 2**30
```

4. Qu'est-ce qu'une variable numérique ?

Lancez Python Shell puis tapez les trois lignes suivantes :

```
>>> x=3
>>> x=x+2
>>> print("Maintenant x vaut :  ", x)
```

La première ligne définit une *variable numérique* en lui donnant un *nom* et un *contenu*. L'instruction ***x*=3** est interprétée de la façon suivante par Python :
- le nom *x* est stocké dans une première zone de la mémoire de l'ordinateur ;
- le nombre entier 3 est stocké dans une autre zone de la mémoire de l'ordinateur ;
- les deux zones sont associées grâce au symbole =, ce qui attribue la valeur 3 au nom *x*.

La ligne ***x*=*x*+2** signifie qu'il faut augmenter de 2 le contenu de la mémoire dont le nom est *x*. L'instruction **print** qui figure sur la troisième ligne fait afficher à la fois le texte placé entre des guillemets et la valeur du nombre *x*.
Notez bien la syntaxe de cette instruction **print** : ce qu'on veut faire afficher doit toujours être placé entre des parenthèses. Quand il s'agit d'un texte, celui-ci doit toujours être placé entre des guillemets simples ou doubles.

■ Remarque : le signe = n'a pas du tout le même sens qu'en mathématique. Utilisé avec Python, il ne marque pas l'égalité mais sert à relier un nom et un contenu. Ecrire ***x*=3**, c'est *assigner* la valeur 3 à la variable *x*.

5. Comment choisir un nom de variable ?

Pour donner un nom à une variable numérique, on peut utiliser des lettres de l'alphabet et des chiffres ou encore des mots complets. On peut choisir par exemple *a*, *b*, *x*, *y, aa, a1,b2* ou *x1* mais on peut également employer des mots comme *longueur* ou *largeur*.

■ Exemple :

```
>>> longueur=20
>>> largeur=15
>>> 2*(longueur+largeur)
70
>>>
```

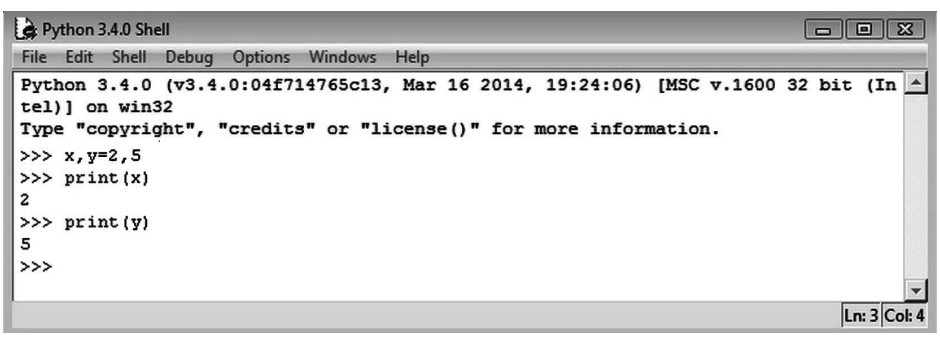
Bien que le choix des noms soit assez large, il faut tout de même respecter quelques règles :

- un nom doit toujours commencer par une lettre ;
- les lettres accentuées, les cédilles, les espaces et les caractères spéciaux tels que $, %, # ou encore @ sont interdits ;
- les lettres majuscules et les lettres minuscules sont considérées comme des caractères différents ; par exemple, les trois noms Aire, aire et AIRE désignent trois variables différentes.
- un nom de variable ne peut pas être l'un des mots réservés du langage Python. Par exemple, on ne peut pas nommer **print** une variable.

6. Les affectations multiples

Il est possible de définir plusieurs variables numériques en une seule fois.

■ Exemple :



Au lieu d'écrire ***x*=2** sur une première ligne et **y=5** sur une seconde, on a écrit ***x*,y=2,5** sur une seule ligne.

7. Les expressions numériques

On peut former des expressions numériques avec des variables numériques, des symboles d'opérations et, s'il y a lieu, des parenthèses. Lorsqu'une expression

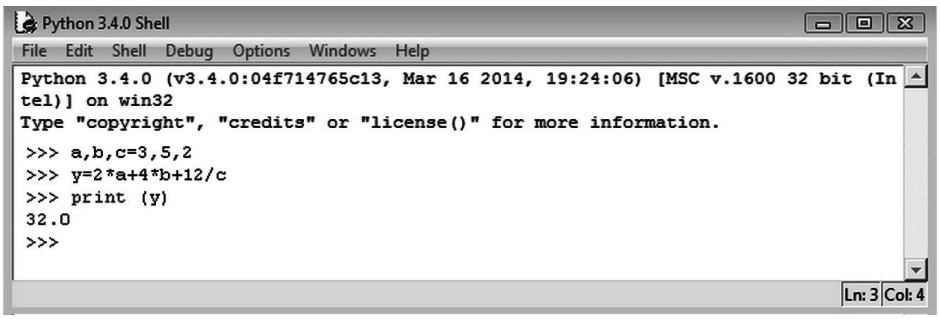
numérique est écrite sans parenthèses, Python respecte les priorités habituelles du calcul.

■ Exemple : entrez les trois lignes suivantes et comparez les résultats obtenus.

```
>>> 4,25 +3,08×2,5
>>> (4,25 +3,08)×2,5
>>> 4,25 +(3,08×2,5)
```

Pour évaluer des expressions qui contiennent des variables numériques, il faut que celles-ci aient d'abord été définies.

■ Exemple :



■ Remarque : au lieu de nommer y l'expression $2a+4b+\frac{12}{c}$, on aurait pu utiliser l'instruction **print** et faire afficher directement le résultat en écrivant cette simple ligne :

```
>>> print (2*a+4*b+12/c)
```

8. Exercices

E01. Exécutez ces deux calculs, 5678×987 ; 324567×678999, à raison de un par ligne.

E02. Exécutez ces trois calculs à raison de un par ligne : 5,45-(3,32×0,8) ; 999**2 puis (5,37+6,24×3)-(3,45×2-56,876).

E03. Comparez les résultats des opérations 56//9 et 56/9.

E04. Effectuez $3,45+\frac{4,6}{5}$ puis $\frac{4,6-3}{5-2,4}$.

E05. Calculez 2**100. Combien le résultat a-t-il de chiffres ?

E06. Quel est le calcul effectué ci-dessous ?

```
>>> q=157//8
>>> r=157-8*q
```

```
>>> print (q)
>>> print (r)
```

E07. Quel est le résultat de ce calcul ?

```
>>> print(15 + 3*2).
```

E08. Que se passe-t-il si vous demandez 5%3 ? Quel est le rôle de l'opérateur **%** ?

E09. Exécutez les calculs suivants et dites ce que vous observez.

```
>>>  x=3
 >>> y=2
>>>  c=y
>>>  y=x
>>>  x=c
>>>  print( "x=", x, "y=", y)
```

E10. Entrez les lignes suivantes et dites ce que vous observez.

```
>>> x, y = 6, 4
>>> x, y = y, x
>>> print( "x=", x, "y=", y)
```

E11. Entrez les lignes suivantes et dites ce que vous observez.

```
>>> a, b, c, d =1, 2, 3, 4
>>> a, b, c, d = b, c, d, a
>>> print ("a=", a, "b=", b, "c=", c, "d=", d )
```

E12. Entrez les lignes suivantes. Python accepte-t-il les calculs avec des nombres relatifs ?

```
>>> 5 - 12
>>> 5 + (-3)
>>> (-4) - (-9
```