Informatique - Notes de cours 1

Initiation à Python

M. Marmorat

17 septembre 2024

1 Introduction

1.1 Python, qu'est-ce que c'est ?

Python est un langage informatique interprété : l'utilisateur (vous, l'enseignant, n'importe qui) écrit un programme ou un script Python, constitué de différentes instructions. Lorsqu'on exécute le programme, les différentes instructions le constituant sont interprétées par le processeur de l'ordinateur et produisent un ou des résultats. Par exemple, le script suivant calcule le double de 42 et affiche le résultat.

```
x=42
y=2*42
print(y)
```

Python est un langage informatique placé sous licence libre, ce qui signifie sommairement que tout un chacun peut contribuer à son évolution et à son amélioration, et que son utilisation est gratuite, à titre privé, public, institutionnel ou commercial.

C'est le langage utilisé pour les cours d'informatique de la classe de BCPST.

1.2 L'environnement de développement

Comment écrire un programme Python (on parle parfois de script Python) sur un ordinateur et comment l'exécuter ? On utilise pour cela ce que l'on appelle un environnement de développement.

Il existe plusieurs environnements de développement différents pour Python. Celui que je vous demande d'utiliser est **Pyzo**, c'est l'environnement de développement que vous utiliserez pendant toute votre formation en BCPST, et c'est aussi celui que vous retrouverez le jour des concours.

A faire pour la semaine prochaine : installer l'environnement de développement **Pyzo** sur son ordinateur et reprendre chaque exemple de ce polycopié.

Vous pouvez vous référer à ce lien pour un tutoriel sur l'installation de Pyzo et de Python : https://www.s2i-chateaubriand-joliotcurie.net/laboratoire-de-sii/logiciels-installation/pyzo

N'hésitez pas à me contacter en cas de difficulté. Il est indispensable que vous puissiez utiliser Pyzo sur votre ordinateur personnel.

1.3 Les commentaires

On peut utiliser des commentaires en Python : ce sont des lignes de code qui ne sont pas interprétées (comme si elles étaient ignorées par la machine), et qui permette d'expliquer le fonctionnement d'un programme.

En Python, un commentaire s'écrit soit sur une ligne, en commençant par le symbole #, soit sur plusieurs lignes, entre les symboles """.

Exemple 1.

```
x = 2 # je suis un commentaire sur une seule ligne, et x vaut 2
"""
Je suis un commentaire
sur
plusieurs lignes
"""
```

2 Les variables

Un des concepts fondamentaux du développement informatique consiste à stocker des valeurs en mémoire. Les variables permettent d'organiser l'exécution du programme en sauvegardant la valeur de certains paramètres importants dans l'exécution de celui-ci.

On parle de l'adresse mémoire de la variable et de la valeur de la variable.

Exercice 1. Qu'affichent les scripts suivants ?

```
y="Bonjour, classe de BCPST 1A"
print(y)
```

```
x=12
z=3*x
print(z)
```

```
x=12
x=x+1
print(x)
```

Exercice 2. Que valent les variables `a` et `b` après exécution du script suivant ?

```
a = 9 * 9
b = a + 1
a = 100
b = b + 2
```

2.1 Une erreur courante...

Pour être utilisée en Python, une variable doit absolument avoir été **déclarée** avant son utilisation.

Exercice 3. Que se passe-t-il lors de l'exécution du script suivant ?

```
y=23
x=2*y
print(z)
```

A retenir : lorsqu'on utilise une variable en Python, on veillera toujours à ce que cette variable ait été **déclarée** ou instanciée au préalable.

2.2 Le type d'une variable

En Python, une variable possède un **type**, qui représente le type de données stocké dans la variable. On accède au type de la variable `x` en tapant `type(x)`. Les différents types de variable sont

1. **int** (comme integer en anglais) pour stocker des nombres entiers ;

```
x = 2
type(x)
```

2. **float** (pour nombre flottant) pour stocker des nombres réels ;

```
y = 2.3
type(y)
```

3. **bool** (pour booléen) pour stocker des valeurs de vérité, vraies ou fausses ; ces deux valeurs sont représentées par les constantes `True` et `False`. Nous reviendrons sur ce type de variable lors du prochain cours.
4. **str** (comme string en anglais) pour stocker des chaînes de caractère.

```
phrase = "Bienvenue au Lycée Marcelin Berthelot"
type(phrase)
```

2.3 Opérations sur les nombres en Python

On dispose de plusieurs opérations standards sur les nombres en Python.

1. +, −, ∗ et / sont l'addition, la soustraction, la multiplication et la division.
2. ∗∗ est l'opération de puissance.
3. // permet d'obtenir le quotient de la division euclidienne d'un nombre entier par un autre.
4. % permet d'obtenir le reste de la division euclidienne d'un nombre entier par un autre.

Exercice 4. Quelle est la valeur des variables `a`,`b` et `c` après exécution du script suivant ?

```
a = 2+3*4**2
b = 12//5+3**2
c = (5\%2)*3
```

2.4 Opérations sur les chaînes de caractères

On peut concaténer (réunir) plusieurs chaînes de caractères en Python à l'aide du symbole +.

```
bienvenue = "Bienvenue dans le cours "
matiere = "d'informatique"
message = bienvenue + matiere
print(message)
```

L'instruction `print(message)` affiche la chaîne de caractères stockée dans la variable `message` à l'écran.

Remarque 1. Pour demander à l'utilisateur d'interragir avec le programme, on peut utiliser la fonction `input`.

```
nom = input("Quel est votre nom ?")
print("Bonjour " + nom + ", nous sommes le 8 septembre 2023")
```

2.5 Une astuce : échanger la valeur de deux variables en Python

On peut échanger la valeur des variables `x` et `y` grâce à l'instruction

```
x, y = y, x
```

Exercice 5. Que valent les variables `c` et `d` après exécution du script suivant ?

```
c = 42
d = "alea jacta est"
c, d = d, c
```

3 Les fonctions

Un concept fondamental en programmation informatique est celui de **fonction**. On utilise une fonction lorsque l'on souhaite répéter plusieurs fois la même opération.

Lorsque l'on souhaite déclarer une nouvelle fonction, on utilise la syntaxe suivante :

```
def nom_de_la_fonction(argument1, argument2, ..., argumentN):
    # le corps de la fonction
    ...
    return valeur_de_sortie
```

On notera l'utilisation du mot-clé `def`, du mot-clé `return` et de *l'indentation* (l'espacement) qui précède le corps de la fonction.

L'indentation sert à préciser à Python que toute la zone indentée constitue le corps de la fonction.

3.1 Valeur de retour d'une fonction

Une fonction est une machine à calculer, comme en mathématiques. Elle prend un ou plusieurs arguments (à qui l'on donne un nom arbitraire), elle calcule une ou plusieurs quantités grâce à ces arguments dans le corps de la fonction, et les renvoie à l'extérieur grâce au mot clé `return`.

Exemple 2. Que fait la fonction suivante ?

```
def mystere(x):
    y = 2*x
    z = y + 1
    return z
```

Une fois une fonction créée (grâce à la construction `def`/`return`), il faut l'appeler pour l'utiliser.

Exemple 3. Que vaut la variable `t` après exécution du script suivant ?

```
t = mystere(5)
```

Toutes les variables qui sont créées à l'intérieur d'une fonction et qui ne sont pas "communiquées au monde extérieur" à l'aide du mot-clé `return` sont détruites après exécution de la fonction.

Exemple 4. Que se passe-t-il après exécution du script suivant ?

```
from math import exp #on importe la fonction exponentielle depuis le
   module math

def f(x):
    message = "Je vais calculer l'exponentielle de 2x+1"
    y = exp(2*x+1)
    return y

print(f(1))
print(message)
```

Attention : prenez garde à bien distinguer les deux instructions `return` et `print` ! Le corps d'une fonction terminera presque systématiquement par l'instruction `return`.
Considérez qu'il est interdit d'utiliser la fonction `print` à l'intérieur d'une fonction !

3.2 Portée d'une variable en Python

On distingue les notions de variables globales et locales.

- **Variable globale :** Une variable déclarée hors du corps d'une fonction est dite globale. Toute modification d'une variable globale dans le corps d'une fonction n'a d'effet que dans le corps de la fonction.

```
mon_nombre = 10 #mon_nombre est une variable globale
def test():
    a = 5
    print("la variable a vaut " + a)

test()
print("la variable a vaut " + a)
```

- **Variable locale :** Une variable déclarée dans le corps d'une fonction est dite locale. Elles sont créées lors de l'appel de la fonction et détruites à la fin de son exécution.

```
def test():
    # nouvelle_variable est locale
    nouvelle_variable = 10
    print("Ma variable vaut " + nouvelle_variable)
test()
print("Ma variable vaut " + nouvelle_variable)
```

4 Quelques exercices

Exercice 6.
1. Quelle expression en Python permet de calculer $\frac{12+2\times(4+102)}{12+78}$?
2. Quelle expression en Python permet de calculer 2^{100} ?
3. Quel calcul peut-on faire en Python pour savoir si 2023 est divisible par 7 ?

Exercice 7. On exécute le code suivant.

```
x = 9
y = x**2
x = 10
y = y + 1
```

Une fois executé, que valent les variables `x` et `y` ?

Exercice 8.

1. Écrire en Python la fonction mathématique qui à x associe $f(x) = x^2 + x - 2$. Que vaut $f(2)$?
2. Écrire une fonction qui prend en argument un entier n et qui renvoie le reste de la division euclidienne de n par 7. Comment calculer le reste de la division euclidienne de 624 par 7 ?