Avignon Université - Culture numérique et code

CM - Premiers pas

07/02/2020

Calculer avec Python

Python présente la particularité de pouvoir être utilisé de plusieurs manières différentes. Vous allez d'abord l'utiliser en mode interactif, c'est-à-dire d'une manière telle que vous pourrez dialoguer avec lui directement depuis le clavier.

```
carlos@wifi:~$ python3
Python 3.6.9 (default, Nov  7 2019, 10:44:02)
[GCC 8.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> 10 + 4
14
>>> exit()
carlos@wifi:~$
```

Opérateurs arithméthiques

```
# Ceci est un commentaire
10.5 / 2 # oui :)
10,5 / 2 # NON ! :(
```

In [1]:

```
10 + 8
```

Out[1]:

18

In [2]:

```
10 - 4
```

Out[2]:

6

In [3]:

```
10 * 4
```

Out[3]:

40

In [4]:

```
10 ** 4
```

Out[4]:

```
10000
```

In [5]:

```
10 / 4
```

Out[5]:

```
2.5
```

In [6]:

```
10 // 4
```

Out[6]:

```
2
```

In [7]:

```
10 % 4
```

Out[7]:

```
2
```

Attention à la hiérarchie des opérations !!!

In [8]:

```
10 * 4 + 11 / 2
```

Out[8]:

```
45.5
```

In [9]:

```
10 * (4 + 11) / 2
```

Out[9]:

```
75.0
```

In [10]:

```
10 * (4 + 11 / 2)
```

Out[10]:

```
95.0
```

Données et variables

L'essentiel du travail effectué par un programme d'ordinateur consiste à manipuler des **données**. Ces données peuvent être très diverses, mais dans la mémoire de l'ordinateur elles se ramènent finalement à une suite finie de nombres binaires.

Pour pouvoir accéder aux données, le programme d'ordinateur (quel que soit le langage dans lequel il est écrit) utilise des variables de différents types.

Une **variable** apparaît dans un langage de programmation sous un *nom de variable*. Pour l'ordinateur, il s'agit d'une référence désignant une adresse mémoire, c'est-à-dire un emplacement précis dans la mémoire vive.

Nom de variables

- Un nom de variable est une séquence de lettres [a-z,A-Z] et de chiffres [0-9], qui doit toujours commencer par une lettre.
- Seules les lettres ordinaires sont autorisées. Les lettres accentuées, les cédilles, les espaces, les caractères spéciaux tels que $, #, @, etc. sont interdits, à exception du caractère _ (souligné).
- La casse est significative (les caractères majuscules et minuscules sont différents).
- Vous ne pouvez pas utiliser comme nom de variables les 33 « mots réservés » ci-dessous (ils correspondent à la grammaire du langage et sont utilisés par l'interpreteur python) :

> and, as, assert, break, class, continue, def, del, elif, else, except, False, finally, for, from , global, if, import, in, is, lambda, None, nonlocal, not, or, pass, raise, return, True, try, while, with, yield

Assignation de valeurs

```
a = 10 # Python assigne la valeur 10 à la variable a
b = 3.1416 # Python assigne la valeur 3.1416 à la variable b
c = "Un petit message." # Python assigne la valeur "Un petit message." à la
 variable c
```

Attention !!! L'assignation de valeurs s'effectue toujours de droit à gauche

```
a = 10 # oui :)
10 = a # NON ! :(
```

Affichage de valeurs

In [11]:

```
a = 10
b = 3.1416
c = "Un petit message."
```

In [12]:

```
a
```

Out[12]:

```
10
```

In [13]:

```
b
```

Out[13]:

```
3.1416
```

In [14]:

```
c
```

Out[14]:

```
'Un petit message.'
```

In [15]:

```
print(a)
print(b, c)
```

```
10
3.1416 Un petit message.
```

Affectations multiples

In [16]:

```
x = y = 10.5
jour, mois, annee = 27, 1, 2020

print("La valeur de 'x' :", x)
print("La valeur de 'y' :", y)

print("Le jour :", jour)
print("Le mois :", mois)
print("L'année :", annee)
```

```
La valeur de 'x' : 10.5
La valeur de 'y' : 10.5
Le jour : 27
Le mois : 1
L'année : 2020
```

Typage des variables

Sous Python, il n'est pas nécessaire d'indiquer spécifiquement le type des variables avant de pouvoir les utiliser. En effet, il suffit d'assigner une valeur à un nom de variable pour que celle-ci soit automatiquement créée avec le type qui correspond au mieux à la valeur fournie.

In [17]:

```
variable_entier = 19
variable_reelle = 9.80665
autre_variable_reelle = 6.626e-34
variable_chaine_de_caracteres = "Ceci est un string"
```

La fonction `type()` permets de savoir le typage d'un variable.

In [18]:

```
print(type(variable_entier))
```

```
<class 'int'>
```

> **Exercice 1** À partir du code suivant expliquer la sortie.

In [19]:

```
v1 = 6
v2 = "6"

print(v1*3)
print(v2*3)
```

```
18
666
```

Conversion de types

In [20]:

```
v2 = "6"
print(v2)
print(type(v2))

v2 = int(v2)
print(v2)
print(type(v2))

v3 = 3
print(v3)
print(type(v3))

v3 = float(v3)
print(v3)
print(type(v3))
```

```
6
<class 'str'>
6
<class 'int'>
3
<class 'int'>
3.0
<class 'float'>
```

Interaction avec l'utilisateur : la fonction input()

La plupart des scripts élaborés nécessitent à un moment donné une intervention de la part de l'utilisateur (entrée d'un paramètre, clic de la souris sur un bouton, etc.). Dans un script, la méthode la plus simple d'interaction consiste à utiliser la fonction intégrée `input()`. Cette fonction provoque une interruption dans le programme courant.

L'utilisateur est invité à entrer des caractères au clavier et à terminer avec `<Entrée>`. Lorsque cette touche est enfoncée, l'exécution du programme se poursuit, et la fonction fournit en retour une **chaîne de caractères** correspondant à ce que l'utilisateur a effectivement saisi. Cette chaîne peut alors être assignée à une variable quelconque, peut être convertie, etc.

In [21]:

```
prenom = input("Entre ton prénom : ")
print("Bonjour", prenom, "!")
```

```
Entre ton prénom : Carlos
Bonjour Carlos !
```

> **Exercice 2** Modifier le code pour demander la longueur du carré à l'utilisateur

```
cote_carre = 5
aire_carre = cote_carre ** 2

print("L'aire du carré est :", aire_carre)
```

Premiers scripts. Comment sauvegarder les programmes

Jusqu'à présent, nous avons toujours utilisé Python en mode interactif: vous avez à chaque fois entré les commandes directement dans l'interpréteur, sans les sauvegardr au préable dans un fichier.

Avant de poursuivre, il est temps d'apprendre à sauvegarder les programmes (synonyme de script) dans des fichiers, de manière à pouvoir les retravailler par étapes successives, ainsi comme les tranférer sur d'autres machines.

Écrire le code suivant dans un **éditeur de texte** quelconque (Notepad, Notepad++, Gedit, Pluma, Geany, SublimeText). SVP PAS DE WORD, POWER POINT, EXCEL, ETC.

```
# airecarre.py
# Auteur : MON_PRENOM MON_NOM
# Date : 07/02/2020
# Mon première script Python,
# le programme calcule l'aire d'un carré de longueur égale à 5.

cote_carre = 5 # Assignation de longueur.
aire_carre = cote_carre ** 2 # Calcul de l'aire.

print("L'aire du carré est :", aire_carre) # Affichage du resultat
```

Enregistrer le programme avec le nom suivant : `airecarre.py`. *Atention au dossier d'enregistrement.*

Par la suite, lorsque nous voudrons tester l'exécution de notre programme, il suffira de lancer l'interpréteur Python en lui fournissant le nom du fichier qui contient le script.

```
carlos@wifi:~$ python3 airecarre.py
L'aire du carré est : 25
```

Exercice 3 Créer le script `airecarre2.py` à partir de l'**Exercice 2** et l'exécuter.