

De l'algorithmique à Python

1. Notion d'affectation

Affecter une variable, c'est lui attribuer une valeur. Le contenu précédent, s'il y en avait un, est **effacé**.

Exercice 1. On considère le script ci-dessous :

ligne 1	$a = 3$
ligne 2	$b = 5$
ligne 3	$c = a + b$
ligne 4	$a = 2$
ligne 5	c

Un élève prétend que la variable c vaut 7 à la fin du script ? Qu'en pensez-vous ?

2. Codage des nombres en machine


En informatique, les nombres réels n'existent pas ! Il faut en effet coder tous les nombres dans un langage compréhensible par les ordinateurs. Chaque nombre est ainsi codé par une liste finie des deux chiffres 0 et 1 (on parle de l'écriture des nombres en binaire). Aucun problème à priori pour les entiers qui ont une écriture en binaire finie. Python code même de façon exacte de très grands entiers. Pour les non entiers, c'est plus compliqué ! Certains décimaux comme 0,5 sont codés de façon exacte en Python. Mais d'autres, qui paraissent « simples » dans notre écriture comme 0,1 ne sont pas codés de façon exacte. C'est donc une valeur approchée du nombre qui est codée en machine. Ceci explique le résultat ci-après. Ce n'est ni un bug de Python ni un bug de l'ordinateur !


```
1 >>> 0.5+0.5
2 >>> 1.0
```

```
1 >>> 0.1+0.7
2 >>> 0.7999999999999999
```

Bilan :

- Python travaille avec deux types de nombres :
 - les entiers de type **int**(integer) qui sont codés de façon exacte.
 - Les autres nombres de type **float**(flottant) qui sont pour beaucoup codés en machine de façon approchée.
- Les calculs sur les entiers uniquement sont des calculs exacts alors que des calculs avec des flottants sont souvent approchés.

 5 et 5,0 sont deux écritures du même nombre entier en mathématiques. En revanche, 5 et 5.0 ne sont pas le même objet en Python.

 Certaines opérations produisent des flottants, même si, d'un point de vue mathématique, le résultat est un entier. C'est le cas de la **division** et de la **racine carrée**.

```
1 >>> (2+4)/2
2 >>> 3.0
```

```
1 >>> sqrt(25)
2 >>> 5.0
```

3. Conséquence sur les booléens

- Lorsqu'on veut effectuer un test, on crée une expression, par exemple $a > b$ qui peut prendre deux valeurs : **Vraie**(True) ou **Faux**(False) suivant les valeurs de a et b . On a ainsi créé un booléen.
- Pour tester une égalité, le signe d'égalité étant utilisé pour l'affectation, on utilise un \gg double égal \gg soit $==$.

Exercice 2. Voici deux scripts :

Script 1

```
1 >>> 5>13/3
2 . . . . .
```

Script 2

```
1 >>> 4<11/3
2 . . . . .
```

Compléter les pointillés relatifs à chacun des deux scripts.

Exercice 3.

1. Soit ABC un triangle tel que $AB = 5, 15$, $AC = 4, 12$ et $BC = 3, 09$.
Démontrer que le triangle ABC est rectangle.
2. Voici un script associé à la situation précédente :

```
1 >>> 3.09**2+4.12**2==5.15**2
2 False
```

Comment peut-on expliquer le False en fin de script ?



On ne peut pas tester de façon certaine une égalité entre deux nombres de type **float**. Il est donc important de déterminer les types de variable en jeu.

Exercice 4. On affecte à n un entier naturel. Dans chacun des cas suivants, exprimer à l'aide d'une phrase en français :

1. $n * 2$

3. $n \% 10$

5. $n / 10$

2. $n + 1$

4. $n * 2$

6. $n - 1$

Exercice 5. Déterminer, le résultat de chaque calcul en Python et son type (**int** ou **float**) :

1. $13 // 5$

3. $(4 + 14) / 2$

2. $13 \% 5$

4. $\text{sqrt}(12 * 2 + 5 * 2)$