1.1 *Variable et affectation*

Une variable est un espace mémoire désigné par une étiquette (un nom). Lorsqu'on appelle une variable nommée `A` dans une instruction (par exemple dans l'addition `A+3` ou dans la comparaison `A > 0`), l'instruction s'effectue en fait sur la valeur affectée à la variable `A`, c'est-à-dire à la valeur stockée dans l'espace mémoire étiqueté par `A`.

Affecter une valeur `v` à une variable `A`, c'est stocker la valeur v dans l'espace mémoire étiqueté par `A`. L'affectation d'une valeur `v` à une variable `A` se fait de la manière suivante :

en langage naturel	en Python
A ← v	`A = v`

Remarque 1 Lorsqu'on affecte la variable `A`, soit elle existait déjà et alors son ancienne valeur est supprimée et remplacée par la nouvelle valeur affectée soit la variable `A` est créée avec la valeur affectée.

Remarque 2 ATTENTION ! En Python, l'ordre est essentiel lors de l'affectation. Pour affecter la valeur `v` à la variable `A`, on ne peut pas écrire `v = A`.

Remarque 3 Il est possible de faire plusieurs affectations simultanées sous la forme `A, B, C = v, w, x`.

Remarque 4 Il est également possible d'échanger les valeurs de deux variables `A` et `B` (ou plus) en utilisant la syntaxe `A, B = B, A`.

1.2 *Opérations sur les nombres et types `int` et `float`*

Python permet d'effectuer des opérations sur les nombres à l'aide des opérateurs suivants : `+` (addition), `-` (soustraction), `*` (multiplication), `/` (division) et `**` (puissance).

En Python, les données sont typées, c'est-à-dire que leurs valeurs sont associées à un certain type. Il existe deux types de nombres : les entiers (type `int`) et les nombres flottants, c'est-à-dire les nombres « à virgule » (type `float`). Rappelons qu'en Python le séparateur numérique n'est pas la virgule mais le point. Ainsi, `A = 3` affecte l'entier 3 à la variable `A` alors que `A = 3.0` ou `A = 3.` affecte le nombre flottant 3 à la variable `A`. Dans ces deux cas, la valeur de `A` est la même (3) mais le type est différent.

L'addition, la soustraction, la multiplication et la puissance (entière) d'entiers renvoient un entier. Tout autre opération renvoie un flottant. Ainsi,

`3+4` renvoie l'entier `7`,
`3*4` renvoie l'entier `12`,
`3/4` renvoie le flottant `0.75`,
`4.6+1.25` renvoie le flottant `5.85`,
`4.3*2.5` renvoie le flottant `10.75`,
`4.5/1.5` renvoie le flottant `3.0`,
`3+2.6` renvoie le flottant `5.6`,
`3*2.5` renvoie le flottant `7.5`,
`2**3` renvoie l'entier `8`,
`3.5+1.5` renvoie le flottant `5.0`,
`12/3` renvoie le flottant `4.0`,
`2.5**2` renvoie le flottant `6.25`.

Il est possible de convertir des entiers en flottants et inversement à l'aide des fonctions `float` et `int`. Par exemple, `float(145)` renvoie le flottant `145.0`. Attention, en revanche, `int(35.67)` renvoie l'entier `35` et `int(-217.345)` renvoie l'entier `-217`.

Le typage a une importance cruciale lorsqu'on fait des opérations sur les variables car certaines opérations ne sont possibles que sur des variables du même type ou car une même opération donnera des résultats différents selon le type des variables.

Il existe des opérateurs plus spécifiques aux entiers : `//` qui renvoie le quotient dans le division euclidienne (c'est-à-dire la division exacte avec reste) et `%` qui renvoie le reste dans la division euclidienne. Ainsi, `17//3` et `17%3` renvoient respectivement `5` et `2` car `17 = 5 × 3 + 2` donc, dans cette division, le quotient est `5` et le reste est `2`.

Remarque 5 Pour connaître le type d'une donnée ou d'une variable, on peut utiliser la fonction `type` qui renvoie `<class 'int'>` pour entier et `<class 'float'>` pour un flottant.

1.3 *Comparaisons et type* `bool`

En Python, on peut effectuer des comparaisons à l'aide des opérateurs suivants.

Tester si	$a = b$	$a \neq b$	$a < b$	$a > b$	$a \leqslant b$	$a \geqslant b$
Syntaxe Python	`a == b`	`a != b`	`a < b`	`a > b`	`a <= b`	`a >= b`

Ce type de test renvoie `True` (vrai) ou `False` (faux) suivant que la comparaison est vraie ou fausse. Ce résultat fait partie d'un type spécifique : le type booléen (`bool`).

On peut effectuer des comparaisons plus complexes en enchaînant plusieurs comparaisons grâce aux opérateurs `or` (ou), `and` (et) et `not` (contraire de).

Ainsi, si on effectue les affectations `A = 3` et `B = -2` alors `A != 0` renvoie `True`, `A == B` renvoie `False`, `A >= 3` renvoie `True`, `A > 0 or B > 0` renvoie `True`, `A > 0 and B > 0` renvoie `False` et `not A == B+5` renvoie `False`.

1.4 *Exercices*

♦ **PYT.1** Pour chacune des instructions suivantes, déterminer le résultat renvoyé ainsi que le type de celui-ci.

1) `4+3`
2) `4+3.`
3) `4*3.`
4) `3/4`
5) `20//3`
6) `20%3`
7) `3>1`
8) `3==2+1`
9) `3**3`
10) `3.**2`
11) `not 4>5`
12) `2>1 and 3>4`
13) `2>1 or 3>4`
14) `17%2 ==1`
15) `not (2**3 > 3**2)`
16) `(3 !=1+1) and (not 3>4)`
17) `not (2==1 or 2>1)`
18) `1.2+3.8 !=5`

♦ **PYT.2** Pour chacun des algorithmes suivants, déterminer la valeur des différentes variables à la fin de l'exécution. Traduire ensuite l'algorithme en un programme Python.

1)
$a \leftarrow 2$
$b \leftarrow 3$
$a \leftarrow a + 1$
$b \leftarrow a + b$

2)
$a \leftarrow 2$
$b \leftarrow 3$
$a \leftarrow a \times b$
$b \leftarrow a^2$

3)
$a \leftarrow 0$
$b \leftarrow 1$
$a \leftarrow b$
$b \leftarrow a$

4)
$a \leftarrow 1$
$b \leftarrow 2$
$a \leftarrow a + b$
$a \leftarrow a \times b$
$a \leftarrow a^b$

♦ **PYT.3** Pour chacun des programmes suivants, déterminer la valeur des différentes variables à la fin de l'exécution.

1)
```
a = 2
b = 3
a = a+b
b = a+b
```

2)
```
a = 2
b = 3
a,b = b,a
b = 2*a+b**3
```

3)
```
a = 2
b = 3
c = 4
a,b,c = 3*c,2*a,b
```

♦ **PYT.4** Sans utiliser l'échange de variable (c'est-à-dire, l'affectation `a, b = b, a`), modifier le programme de la question 3 de l'exercice 2 de telle façon qu'à la fin de l'exécution, les valeurs de `a` et `b` soient échangées.