7 TP N° 7 – Variables et affectation

1 Les bases de l'affectation

Exercice 7.1

La variable a prend la valeur 5,
la variable b prend la valeur 8,
la variable b prend la valeur de la variable a.

Autrement dit :
$a \leftarrow 5$
$b \leftarrow 8$
$b \leftarrow a$

Quelles sont les valeurs de a et de b ?

Vérifions à la console de Python :

```
>>> a=5
>>> b=8
>>> b=a
>>> a
5
>>> b
5
```

Définition 7.1 (Affectation)

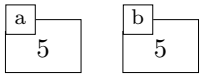
Dire que **la variable *b* prend la valeur de la variable *a*** signifie que :
- la valeur qui était dans la variable b est effacée ;
- la valeur qui était dans la variable a est recopiée dans la variable b ;
- la valeur qui était dans la variable a est inchangée.

On dit aussi que **la valeur de *a* est affectée à *b***.
On écrit : $b \leftarrow a$

Exemple 7.1

$a \leftarrow 5$ 

$b \leftarrow 8$ 

$b \leftarrow a$ 

Exercice 7.2

Quelqu'un souhaite affecter la valeur 3 à la variable c et saisit à la console : `>>> 3=c`

Quand il valide, cela déclenche un message d'erreur. Pourquoi ?

```
>>> 3=c
  Last command
SyntaxError: can't assign to expression
```

Traduction : *can't assign to expression* signifie : *on ne peut pas affecter quelque chose à une valeur.*
En effet, 3 ne peut pas prendre une valeur, parce que 3 est une valeur.

Propriété 7.1

On ne peut pas affecter quelque chose à une valeur, autrement dit une valeur ne peut pas prendre une valeur.

Vocabulaire

En anglais, en programmation, le mot *assign* signifie *affecter* et le mot *expression* signifie *valeur*

Exercice 7.3

1. Saisir à la console `>>> 4*a`, puis valider.
2. Saisir à la console `>>> 4*d`, puis valider.

```
>>> 4*a
20
>>> 4*d
  Last command
NameError: name 'd' isn't defined
```

Traduction :

- *NameError* signifie *erreur de variable*
- *name 'd' is not defined* signifie *la variable d n'est pas définie*

Propriété 7.2

En Python, tant qu'une variable n'a pas reçu d'affectation, cette variable n'est pas définie.

2 Fonction et affectation

Un élève crée la fonction Python nommée `pr` ci-dessous pour calculer le périmètre d'un rectangle.

Les variables `lo` et `la` sont respectivement la longueur et la largeur d'un rectangle, et ce sont des nombres réels.

```
def pr(lo,la):
  return(2*(lo+la))
```

Pour calculer le périmètre d'un rectangle de longueur 9 et de largeur 6, il saisit `>>> pr(9,6)` à la console et il valide.

```
>>> pr(9,6)
30
```

L'équivalent de cette définition de fonction et de son exécution à la console est donnée par la suite d'instructions à la console détaillée ci-dessous.

```
>>> lo=9
>>> la=6
>>> p=2*(lo+la)
>>> p
30
```

3 Type de variables

1. Saisir à la console : `>>> r=7` puis `>>> type(r)`

2. Saisir à la console : `>>> s=1.4` puis `>>> type(s)`

```
>>> r=7
>>> type(r)
<class 'int'>
>>> s=1.4
>>> type(s)
<class 'float'>
```

Explication

- la variable r a pris la valeur 7, qui est un nombre entier (*integer* en anglais), donc la variable r est du type "entier".
- la variable s a pris la valeur 1,4 qui est un nombre décimal, et *float* signifie *floating point number* qui désigne un nombre à virgule.

Définition 7.2 (types de variable `int` et `float` en Python)

- Une variable qui a pris la valeur d'un nombre entier, est du type `int` (*integer* en anglais).
- Une variable qui a pris la valeur d'un nombre décimal, est du type `float` (*floating point number* en anglais).

Remarque 7.1

Il y a d'autres types de variables en Python, qui seront étudiés ultérieurement.