

Chapitre 1 : Séquentialité et variables

- Les *instructions* sont *évaluées* les unes après les autres en suivant l'ordre d'écriture dans le programme.
- Les *instructions* peuvent être:
 - une *opération*,
 - un *appel de fonction*,
 - une *opération d'affectation* à une *variable*,
 - une *composition* des opérations précédentes
- Seule une *valeur* est *affectée* (stockée) dans une *variable*. Cette *valeur* est le résultat de l'*évaluation* des *instructions* à droite du signe =.
- La *valeur affectée* à une *variable* peut être modifiée par une *affectation* ultérieure lors du déroulement séquentiel du *programme*.
- Pour utiliser la *valeur* d'une *variable* dans une *instruction*, on donne simplement son nom. Le nom est substitué par la *valeur* de la *variable* avant l'*évaluation* de l'*instruction*.

```
In [ ]: x = 2          # x vaut désormais 2
        y = 3          # y vaut désormais 3
        print(x)
        x = x + y + 5   # avant l'opération x et y sont remplacés respectivement par 2 et 3
        print(x)       # x vaut 10, le résultat de l'opération 2 + 3 + 5 et de l'affectation
```

- Les *variables* et les *littéraux* sont de différents *types*:
 - int: **entier**
 - float: **nombre à virgule**
 - str: **chaîne de caractères**
 - bool: **booléen**
- Le *type* de la *valeur/variable* conditionne les *opérations* que l'on peut faire avec.
- Il est, dans certains cas, possible de *convertir (caster)* une *valeur* d'un *type* à un autre *type*.
- l'appel à la fonction `print(...)` permet d'afficher

```
In [ ]: print("Hello")
```

- l'appel à la fonction `input()` dans l'instruction `x = input()` permet d'affecter à la *variable* `x` ce qu'un utilisateur tape au clavier (**Attention** sans cast explicite, la valeur dans `x` est une chaîne de caractères).

```
In [ ]: x = input()
        print(int(x)+1)
```

Vocabulaire

- valeur, littéral, variable, type, instruction, fonction, opération, évaluer, affecter, convertir - caster, appel,...