

## 1.1 Introduction

Un ordinateur peut être vu comme une machine à calculer et à retenir des informations.

### Définition 1

**Programmer**, c'est donner un ensemble de tâches à réaliser à un ordinateur.

### Définition 2

Un **langage de programmation** est un langage spécial qui permet d'écrire des commandes compréhensibles par un ordinateur.

En SNT, vous allez utiliser le langage **Python**, qui est très utile pour la majorité des usages. **Python** est dit "de haut niveau", non pas parce qu'il est difficile, mais parce qu'il propose de nombreux raccourcis et outils qui facilitent la programmation.

Une des premières notions à apprendre pour programmer est la mémoire. Il est bien utile de se rappeler de choses quand on programme, votre nombre de points de vie dans un jeu vidéo, le pseudo de votre personnage, etc, pour cela on utilise des variables.

### Définition 3

Une **variable** est l'association de trois éléments, un **espace mémoire**, un **nom**, et une **valeur**. On peut voir cela comme une étiquette collée sur une boîte contenant une donnée.

### Exemple 1

```
1 x = 5
```

signifie que la variable `x` contient la valeur 5.

```
1 prenom = "Alice"
```

signifie que la variable `prenom` contient le texte "Alice".

**Remarque 1.** Le signe `=` ne veut pas dire « égal » comme en mathématiques, mais « prend la valeur ». En informatique `x = x + 1` a un sens. La variable "x" voit sa valeur augmenter de 1.

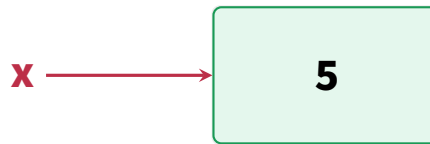


FIGURE 1 – Une variable nommée x contenant la valeur 5

**Définition 4**

Associer une valeur à une variable s'appelle une **affectation** et utilise le signe =

## 1.2 Conventions de nommage

**Propriété 1.** Un **nom de variable** doit respecter certaines règles :

- Il ne peut contenir que des lettres (sans accent), des chiffres et le caractère \_.
- Il doit toujours commencer par une lettre ou par \_.
- Il ne doit pas être un **mot réservé** de Python (on en verra tout un tas par la suite).

**Remarque 2.** Par convention, en Python, on utilise souvent le style dit **snake\_case** : les mots sont en minuscules et séparés par des tirets bas (\_). Exemple : `age_eleve`, `note_math`, `prix_total`.

**Exemple 2**

**Correct :** `compteur`, `nombre_eleves`, `a2`.

**Incorrect :** `2x`, `âge`, `nom élève`, `print`.

**Exercice 1**

Parmi les noms de variables suivants, entourez ceux qui sont incorrects et expliquez pourquoi :

- 2notes
- note\_finale
- p  tate
- nom eleve
- porte-monnaie
- moyenne\_generale
- \$hello\$

**Correction :** les noms de variables incorrects sont :

- 2notes (commence par un chiffre)
- nom eleve (espace interdit)
- porte-monnaie (tiret)
- \$hello\$ (caract  re \$ non autoris  )

note\_finale et moyenne\_generale sont corrects.

p  tate est techniquement correct mais peu recommand  !

## 1.3 Utiliser les variables

**D  finition 5**

Une commande donn  e    l'ordinateur gr  ce    un langage informatique est appel  e une **instruction**

**Exemple 3**

Vous avez d  j   vu un exemple d'instruction pr  c  demment.

```
ville = "Montivilliers"
```

donne l'ordre    l'ordinateur de cr  er une variable, de lui donner le nom "ville" et la valeur "Montivilliers"

**Définition 6**

Un ensemble d'instructions est appelé un **programme**

**Définition 7**

**Exécuter un programme**, c'est demander à l'ordinateur de lire et de réaliser chaque instruction du programme, l'une après l'autre, dans l'ordre.

Comme en Mathématiques, on peut combiner les variables grâce à des opérations indiquées dans le tableau suivant :

Opérateur	Opérations
+	addition
-	soustraction
*	multiplication
/	division (résultat décimal)
//	quotient de la division entière (ou euclidienne)
%	reste de la division euclidienne

**Exercice 2**

Considérons le programme suivant :

```

1  x = 5
2  y = 3
3  x = x + y + 2

```

1. Représentez l'évolution des valeurs contenues dans les variables  $x$  et  $y$  au fur et à mesure de l'exécution du programme.

**Correction :**

- après  $x = 5$  :  $x = 5$ ,  $y$  n'existe pas encore;
- après  $y = 3$  :  $x = 5$ ,  $y = 3$ ;
- après  $x = x + y + 2$  :  $x = 10$ ,  $y = 3$ .

2. Que contiennent les variables  $x$  et  $y$  après l'exécution de ce programme ?

**Correction :**  $x = 10$  et  $y = 3$ .

**Exercice 3**

Considérons le programme suivant :

```
1 a = 10
2 b = 2
3 a = a - b
4 b = a * b
```

1. Quelle est la valeur de  $a$  après la première ligne ?

**Correction :** après  $a = 10$ , on a  $a = 10$ .

2. Quelle est la valeur de  $b$  après la deuxième ligne ?

**Correction :** après  $b = 2$ , on a  $b = 2$ .

3. Donnez les valeurs finales de  $a$  et  $b$  une fois le programme entièrement exécuté.

Après  $a = a - b$ ,  $a = 10 - 2 = 8$  et  $b = 2$ .

Après  $b = a * b$ ,  $b = 8 \times 2 = 16$ .

**Correction :** valeurs finales  $a = 8$  et  $b = 16$ .

## 1.4 Les types de variables

Vous avez pu constater précemment que dans certaines variables, nous avons rentré des nombres et dans certaines autres des mots. Nous ne pouvons pas faire exactement comme on le souhaite et devons suivre des règles.

**Définition 8**

Une **variable** en informatique est toujours associée à un **type**, qui indique de quelle sorte de donnée il s'agit (nombre, texte, etc.).

**Propriété 2.** Le **type** d'une variable détermine ce que l'on peut faire avec la valeur qu'elle contient.

**Exemple 4**

En Python, les types les plus simples sont :

- `int` : les nombres entiers (ex. 7, -3, 2025)
- `float` : les nombres décimaux (ex. 3.14, -0.5) (⚠ On utilise le point pas la virgule!)
- `str` : les chaînes de caractères (les mots), entourées de guillemets (ex. "bonjour")
- `bool` : les valeurs de vérité Vrai ou Faux (cf. Chap 0.2)

**Remarque 3.** Une même opération peut avoir un sens différent selon le type :

```
1 >>> 3 + 4
2 7
3 >>> "bonjour" + "vous"
4 bonjour vous
```

#### Exercice 4

```
1 a = 7
2 b = 2.5
3 c = "bonjour"
4 d = "7"
```

1. Quel est le type de a, de b, de c, et de d?

**Correction :** a est de type `int`, b est de type `float`, c est de type `str`, d est aussi de type `str` (même si elle contient le caractère "7").

2. Quelle va-t-il se passer si on exécute `7 + "7"` selon vous?

**Correction :** Python affiche une erreur de type (`TypeError`) : on ne peut pas additionner un entier (`int`) et une chaîne de caractères (`str`).

#### Exercice 5

```
1 x = 10
2 y = 3
3 rep_1 = x // y
4 rep_2 = x % y
5 rep_3 = x / y
```

1. Que vaut `rep_1`?

**Correction :** `rep_1` vaut  $10 // 3 = 3$  (quotient entier).

2. Que vaut `rep_2`?

**Correction :** `rep_2` vaut  $10 \% 3 = 1$  (reste de la division par 3).

3. Que vaut `rep_3` et comment va-t-il être affiché selon vous?

**Correction :** `rep_3` vaut  $10 / 3 \approx 3.3333333333333335$ , il est affiché comme un nombre décimal (type `float`) avec un point.

## 1.5 Une première fonction, l’affichage !

### Définition 9

Une **fonction** est un outil qui réalise une action précise lorsqu’on l’appelle par son nom. On peut lui donner des informations (les **arguments**) et elle peut renvoyer un résultat.

### Définition 10

`print` est la fonction qui permet d’**afficher une valeur** pour l’utilisateur.ice.

### Exemple 5

```
1 >>> print("Bonjour")
2 Bonjour
```

Elle affiche à l’écran le texte Bonjour.

**Remarque 4.** La notation `>>>` désigne l’utilisation de Python en **ligne de commande**, qui sera détaillée pendant votre premier TP.

### Exercice 6

Que va afficher ce programme ?

```
1 a = 3
2 b = 4
3 print(a + b)
```

**Correction :** le programme affiche 7.

Afficher un élément, c’est bien, mais parfois on veut en afficher plusieurs, on peut alors les séparer par une **virgule**

### Exemple 6

```
1 >>> age = 18
2 >>> print("la personne a", age, "ans")
3 la personne a 18 ans
```

**Remarque 5 (Bonus).** L’entier 4 est différent de la chaîne de caractère "4", pour passer de l’un à l’autre, on utilise les fonction **int** et **str**