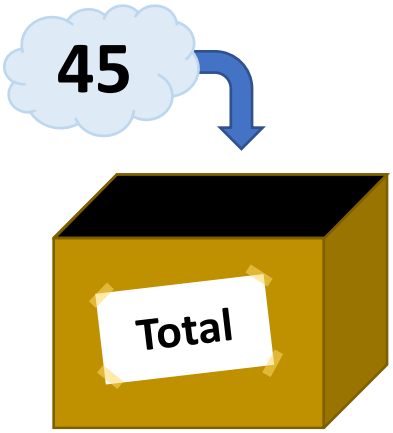
PYTHON 02 : Les variables, les types de données, et les entrées / sorties

I./ Notion de variable :

On peut représenter une variable par une boîte qui peut contenir une donnée. Sur cette boîte, on colle une étiquette où on va écrire le **nom** (ou l'**identifiant**) de la variable. La donnée que l'on place à l'intérieur de la boîte s'appelle la **valeur** (ou le **contenu**) de la variable.



Par exemple, si on tape : **`total = 45`** dans la console Python, on crée une variable de nom « **`total`** » et on y place la valeur 45, qui est un nombre entier.

On peut alors utiliser la variable « **`total`** » dans des calculs : si on tape : **`total + 10`** dans la console Python, on obtient : **`55`**.

De la même façon qu'on peut vider une boîte de son contenu pour y stocker autre chose, on peut modifier la valeur d'une variable : dans la console Python, taper : **`total = 4`**. Pour vérifier que le contenu de la variable a changé, on peut taper « **`total`** » suivi de la touche « Entrée » dans la console Python.

On peut aussi stocker le résultat d'un calcul dans une variable : taper « **`x = 5 + 6`** », puis « **`x`** » pour afficher le résultat. Taper ensuite « **`x = x * 2`** » puis « **`x`** ». Que remarque-t-on ?

On peut donner n'importe quel nom à une variable à condition de respecter les règles suivantes :

- Le nom de la variable ne peut contenir que des lettres majuscules, des lettres minuscules, des chiffres ou le caractère « **`_`** » (appelé « underscore » ou « tiret du 8 »), à l'exception de tout autre symbole.
- Le premier caractère du nom de la variable ne peut pas être un chiffre.
- Le nom de la variable ne peut pas être un **mot–clef réservé** du langage Python : ce sont des mots du langage Python ayant une signification spéciale, comme « **`and`** », « **`or`** », « **`not`** », « **`break`** », « **`continue`** », « **`def`** », etc… Ces mots–clefs ne sont pas très nombreux. Une liste plus complète peut être trouvée à l'adresse suivante : https://fr.wikibooks.org/wiki/Programmation_Python/Tableau_des_mots_r%C3%A9serv%C3%A9s

Remarque : le fait de créer une variable et de placer une valeur dedans, en faisant par exemple **`variable = 12`**, s'appelle l'opération d'affectation.

II./ Les types de données :

Dans les exemples précédents, nous avons utilisé des **nombres entiers**. Python peut aussi travailler sur d'autres **types de données**, comme par exemple les nombres à virgule (appelés **nombres flottants** ou nombres à virgule flottante), ou les **chaînes de caractères**. Il existe aussi d'autres types de données manipulables par Python mais dont nous ne parlerons pas cette année.

1./ Les nombres entiers (ou int) :

Les nombres entiers correspondent à l'ensemble $\mathbb{Z}$ des entiers relatifs vus en mathématiques. Ce sont donc des nombres entiers pouvant être négatifs, positifs, ou nuls. En Python, les nombres entiers peuvent être aussi grands qu'on veut, il n'y a pas de limite (à part la mémoire disponible).

Les opérations usuelles pouvant s'appliquer sur les nombres entiers sont : **`+`**, **`-`**, **`*`**, **`/`**, **`//`** et **`%`**. **`//`** donne le quotient de la division euclidienne, et **`%`** donne le reste de la division euclidienne.

Exemple : taper dans la console Python : **`ab = 18`**, puis : **`cd = 5`**. Taper ensuite les opérations suivantes et noter les résultats :

`ab + cd` Résultat :

`ab - cd` Résultat :

ab * cd Résultat :

ab / cd Résultat :

ab // cd Résultat :

ab % cd Résultat :

2./ Les nombres flottants (ou float) :

Ce sont les nombres à virgule, positifs ou négatifs. Ils ne peuvent pas être aussi grands qu'on veut et sont forcément compris entre $-1{,}8 \times 10^{308}$ et $+1{,}8 \times 10^{308}$ environ, ce qui est très largement suffisant pour la majorité des calculs. Ces nombres ne correspondent pas exactement à l'ensemble $\mathbb{R}$ des nombres réels en mathématiques, pour des raisons que vous verrez éventuellement en spécialité NSI de première.

Les opérations usuelles pouvant s'appliquer sur les nombres entiers sont : **+**, **−**, *****, **/**.

Exemple : taper dans la console Python : **ab = 8.2** puis : **cd = 5.1**. Taper ensuite les opérations suivantes et noter les résultats :

ab + cd Résultat :

ab − cd Résultat :

ab * cd Résultat :

ab / cd Résultat :

3./ Les chaînes de caractères (ou str) :

Python peut manipuler des chaînes de caractères. Une chaîne de caractères est un ensemble de lettres, de chiffres ou de symboles divers, pouvant représenter un mot, une phrase ou un texte. Une chaîne de caractères doit être délimitée par des apostrophes « **'** » ou par des guillemets « **"** ».

Exemple : dans la console Python, taper : **nom = 'ALEXANDER'**, puis **nom** suivi de la touche Entrée.

Les opérations « **+** » et « ***** » peuvent s'appliquer sur les chaînes de caractères, mais elles ne correspondent bien évidemment pas aux opérations mathématiques vues précédemment.

Dans la console Python, taper : **first = 'BONJOUR, '**, puis **second = 'LES AMIS !'**.

Puis taper les commandes suivantes et notez le résultat :

first + second Réponse : (Cette opération porte le nom de **concaténation**).

first * 4 Réponse :

3 * second Réponse :

4./ La fonction type :

On a vu que l'on pouvait créer des variables de trois types différents : **int**, **float** et **str**. Mais comment savoir quel est le type d'une variable en particulier ?

Dans la console Python, taper : **a = 42**, puis **x = 4.56** et enfin **message = 'SALUT !'**.

Puis taper les commandes suivantes et noter les résultats :

type(a) Résultat :

`type(x)` Résultat :

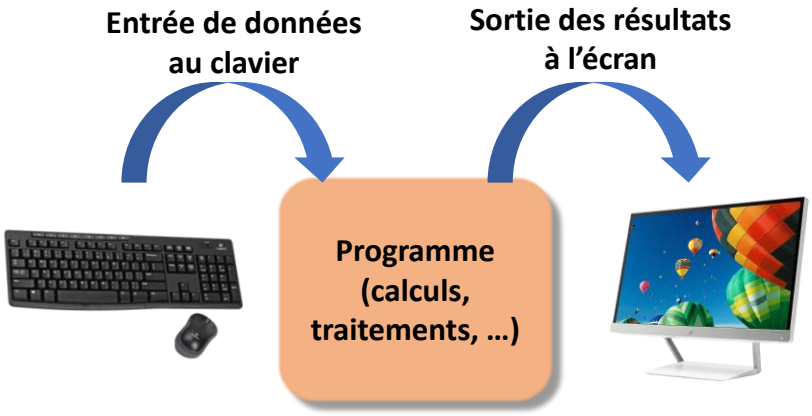
`type(message)` Résultat :

Remarques :

- Si on veut créer une variable `x` de type `float` mais qui contient un nombre entier, il faut penser à rajouter un point décimal. Taper par exemple : `x = 42`, puis `type(x)`. Taper ensuite : `x = 42.0`, puis `type(x)`.
- Certaines opérations peuvent changer le type d'une variable. Taper par exemple : `a = 7`, puis `type(a)`. Taper ensuite : `a = a / 2`, puis `type(a)`. Le résultat était-il prévisible ?
- Les opérations mathématiques peuvent s'appliquer à la fois à des variables de type `int` et de type `float`. Taper par exemple : `x = 5`, puis `type(x)`. Taper ensuite : `y = 5.6`, puis `type(y)`. Faire enfin : `x + y`.

III./ Instructions d'entrée–sortie :

Nous avons vu jusqu'ici comment nous servir de Python pour faire des calculs et pour mémoriser des nombres dans des variables. En gros nous l'avons utilisé comme une calculatrice avec mémoire. Mais on peut faire beaucoup plus, en réalisant des programmes. Un **programme Python** (ou **script Python**) est un ensemble d'**instructions** qui va réaliser une tâche plus ou moins complexe. Pour qu'un programme soit utile, il faut qu'il puisse recevoir des informations en provenance de l'utilisateur, et qu'il puisse donner les résultats à l'utilisateur. C'est ce que l'on appelle les **instructions d'entrée–sortie**.



1./ La fonction print :

Dans l'éditeur Python, taper le programme suivant :

```
a = 2
b = 3
resultat = a + b
print("Le résultat de", a, "plus", b, "est", resultat)
```

Lancer le script à l'aide de la flèche verte. Qu'affiche l'ordinateur ?

La fonction `print()` permet d'afficher les différents éléments placés entre les parenthèses. Les différents éléments doivent être séparés par des virgules.

2./ La fonction input :

Dans l'éditeur Python, taper le programme suivant :

```
nom = input("Entrez votre nom : ")
print("Bonjour", nom)
```

La fonction `input()` demande à l'utilisateur de taper une **chaîne de caractères** qui peut être ensuite stockée dans une variable.

Taper maintenant le programme suivant :

```
a = input("Tapez un premier nombre : ")
b = input("Tapez un deuxième nombre : ")
resultat = a + b
print("Le résultat de", a, "plus", b, "est", resultat)
```

Lancer ce programme à l'aide de la flèche verte, et taper les deux nombres de votre choix. Est-ce que le programme fonctionne correctement ? N'y a-t-il pas un problème ? Le soucis vient du fait que la fonction **`input()`** fournit une chaîne de caractères. Les variables **`a`** et **`b`** contiennent donc chacune d'elle une chaîne de caractères, et l'opérateur « **`+`** » va leur faire subir une opération de concaténation. Nous n'avons donc pas le résultat voulu ! Comment faire pour que l'utilisateur puisse communiquer un nombre au programme ?

Modifier le programme précédent afin d'obtenir :

```
a = int(input("Tapez un premier nombre : "))
b = int(input("Tapez un deuxième nombre : "))
resultat = a + b
print("Le résultat de", a, "plus", b, "est", resultat)
```

Lancer ce programme à l'aide de la flèche verte, et taper **`4`** puis **`5`**. Est-ce que le résultat vous convient ? Relancer le programme et taper **`5`** et **`4.5`**. Que se passe-t-il ? À votre avis, comment faudrait-il modifier le programme pour qu'il puisse fonctionner avec des nombres flottants ?

Mettons maintenant en pratique ce que nous venons de voir. Pour cela vous allez devoir réaliser les programmes suivants. Pensez à les enregistrer, soit sur votre lecteur réseau, soit sur votre clef USB.

1./ Faire un programme qui vous demande votre prénom et votre nom de famille, puis qui vous salue de la manière de l'exemple suivant :

```
Quel est votre prénom : Jacques
Quel est votre nom : Cepte
Bonjour Jacques Cepte
```

2./ Faire un programme qui demande l'âge de l'utilisateur, puis qui indique quel âge l'utilisateur aura dans vingt ans. Exemple :

```
Quel est votre âge ? 25
Dans vingt ans, vous aurez 45 ans.
```

3./ Modifier le programme précédent de façon à ce qu'il demande l'âge de l'utilisateur ainsi qu'un nombre d'années, puis qui calcule le décalage. Exemple :

```
Quel est votre âge ? 52
Entrez un nombre en années : 15
Dans 15 ans, vous aurez 67 ans.
```

4./ Faire un programme qui demande deux nombres flottants, un dividende et un diviseur, puis qui affiche le résultat de la division. Exemple :

```
Entrez le dividende : 98765432
Entrez le diviseur : 8
Le résultat de 98765432.0 divisé par 8.0 est 12345679.0
```

Que se passe-t-il si on entre **`0`** pour la valeur du diviseur ? Dans le prochain chapitre, nous verrons comment corriger ce défaut.