

Python

Les différents types de variables

Type	Nom	Commande	Exemple
Entier	integer	int	a=2
Décimal	float	float	b=2.0
Chaîne de caractères	string	str	c="2"

Exemples

a=2

b=a*2

print(b)

Le programme permet de dire que a et b sont des entiers. Le programme donne le résultat de b (4).

a="2" b=a*2

print(b)

Le programme permet de dire que a et b sont des chaînes de caractères. Le programme donne le résultat de b (22).

a=7.5 b=int(a) print(b)

Le programme permet de dire que a est un décimal (float) et b est un entier (int). Le programme donne le résultat de b (7).

a="fromage" b=float(a)

Le programme permet de dire que a est une chaîne de caractères (str) et b est un entier décimal (float). Ce programme est source d'erreur car on ne peut pas transformer des lettres en nombre (sauf si l'on crée un programme spécial).

Remarque

- Le séparateur des nombres décimaux est le point et non la virgule (comme dans les pays anglo-saxons).
 Ainsi, 3,7 s'écrit 3.7 en Python.
- Les symboles opératoires sont classiques (+ pour l'addition, pour la soustraction, / pour la division et * pour la multiplication. Par contre, une puissance se fait à l'aide de deux astérisques **. Ainsi 3^2 se note 3^{**2} en Python (dans d'autres langages, c'est souvent l'accent circonflexe qui est utilisé : 3^2). Pour calculer une racine carrée, on a besoin d'importer la bibliothèque math à l'aide de l'instruction import math en début de programme puis d'écrire math.sqrt(3) pour calculer $\sqrt{3}$. Sqrt est l'acronyme de square root (racine carrée en anglais).

Instruction conditionnelle

En python, l'expression d'une condition se rédige avec les marqueurs **if** et **else** (si et sinon). Au niveau de la syntaxe les lignes contenant if et else se terminent par deux points (:). On indente chaque ligne qui dépend de if et else à l'aide de la touche tabulation



L'indentation est constituée par un espace en début de ligne. En cas de programme complexe, on peut avoir plusieurs indentations successives. La touche espace ne permet pas de créer des indentations.

Remarque

Si on n'utilise pas else, il ne se passe rien si la condition if n'est pas réalisée.

Exemple

L'instruction **input** permet à l'ordinateur de demander à l'utilisateur du programme de noter quelque chose. Ici, la demande étant de type string (chaîne de caractères), l'ordinateur considère que la réponse est une chaîne de caractères. L'instruction **float** permet de transformer la réponse de l'utilisateur en décimal.

Le programme permet donc à l'utilisateur de choisir un nombre et de lui dire s'il est positif ou négatif.

Remarque

On peut générer plusieurs conditions dans le programme à l'aide de l'instruction **elif**. On peut complexifier une condition avec **and** (et) et **or** (ou).

Instructions utiles

```
    == égal à (test d'égalité)
    != différent de
    > strictement supérieur à
    >= supérieur ou égal à
    <= inférieur ou égal à</li>
```

Exemple

```
def jeu(n):
    if 1<=n<=4:
        print("Vous avez perdu 3 euros")
    elif n==5:
        print("Vous avez gagné 1 euro")
    elif n==6:
        print("Vous avez gagné 5 euros")</pre>
```

Cette fonction permet dans un jeu où l'on tire un dé à 6 faces, de déterminer le gain en fonction du résultat : on perd 3 euros si l'on obtient entre 1 et 4, on gagne 1€ si l'on fait 5 et 5€ si l'on fait 6.

Les boucles

Lorsque l'on veut répéter plusieurs fois une instruction, on utilise des boucles. Il en existe de 2 types : les boucles **bornées** (**for ... in range**) et les boucles **non bornées** (**while**).

Les boucles bornées

Elles permettent de répéter une instruction (éventuellement dépendante de l'indice de la boucle) un nombre de fois fixé à l'avance.

Exemple

A la première exécution de la boucle, l'indice i prend la valeur 1 puis 2 puis 3 puis 4 puis 5.

Il y a un décalage d'une unité entre les valeurs prises par i et la borne supérieure de la boucle (ici 6).

Remarques

- Si la boucle commence à 0, on utilise l'instruction for i in range(6) au lieu de range(0,6).
- On peut utiliser un autre nom que i pour l'indice du moment que ce n'est pas une variable utilisée dans le programme.

Les boucles non bornées

Elles permettent de répéter une instruction tant que la condition n'est pas remplie.

Exemple

```
Import random
n=random.randint(1,20)
k=0
while k!=n:
    k=int(input("Choisis un nombre entier entre 1 et 20 : "))
print("Tu as deviné le nombre que j'avais choisi")
```

Tant que l'utilisateur du programme n'a pas trouvé le nombre, l'instruction dans la boucle continue à tourner.

Remarque

Import random permet d'importer la bibliothèque random qui contient des instructions concernant les probabilités telles que **random.random()** (choisit un nombre décimal au hasard entre 0.0 et 1.0) ou **random.randint(a,b)** (qui choisit un entier entre a et b).

Les fonctions

Les fonctions en informatique diffèrent un peu des fonctions mathématiques : elle permettent de donner un résultat de tout type en fonction d'aucun, d'un ou de plusieurs paramètres de tous types (entier, décimal, chaîne de caractères,...). On utilise ainsi la commande def pour définir une fonction.

Exemples

```
def carré(n) :
return n*n
```

Cette fonction calcule le carré d'un nombre.

```
def concat(a,b) :
    print("a","b")
```

Cette fonction écrit l'assemblage de deux mots (concaténation).

Remarques

• L'instruction def ne lance aucune commande : c'est comme une recette de cuisine que l'on lit dans un livre sans préparer la recette. Si l'on veut exécuter la fonction, il faudra rajouter une ligne de commande de la fonction en indiquant les paramètres choisis. Par exemple, si l'on écrit

```
concat(car,ré)
```

à la suite du programme précédent, le programme écrira carré lors de son exécution.

- Si la fonction ne possède aucun paramètre, on écrira () après le nom de la fonction dans sa définition.
- Les variables utilisées à l'intérieur de la fonction sont locales et ne peuvent donc être appelées en-dehors de la définition de la fonction. Une variable globale peut quant elle être rappelée et donc utilisée n'importe où dans le programme.