TP1 - Les types de base et `if`

M. Tellene

1 Création de l'environnement de travail

Avant de commencer à coder, vous allez créer votre environnement de travail. Pour ce faire vous allez dans votre dossier personnel, puis « apprentissage_python ». Une fois arrivé, créer un dossier « TP2 ». C'est dans ce dossier qu'il faudra mettre tous les exercices du TP.

2 Les types de base

Une variable peut prendre différents types en fonction de la valeur qu'elle contient. Parmi les types les plus communs en Python, on trouve :

- les nombres entiers (de type `integer`, noté `int`) positifs ou négatifs.
- les nombres flottants (de type `float`) comparables aux décimaux. Attention, le séparateur décimal est le point « . », pas la virgule ! Ainsi, le nombre 2,3 s'écrira 2.3.
- les chaînes de caractères (de type string, noté `str`). Pour distinguer les noms de variables des chaînes de caractères, il faut écrire ces dernières entre apostrophes « ' » ou guillemets « " ».

3 Un peu de manipulation

Ouvrir Thonny Python et taper dans la console (la partie du bas) le code suivant :

```
>>> n = 7
>>> n

>>> type(n)
```

Quel est le type de `n` ? .....

A quoi sert l'instruction `type(...)` ? .....

Maintenant taper le deux instructions suivantes :

```
>>> float(n)

>>> str(n)
```

L'affichage de la variable a-t-il changé ? Si oui, comment ? .....

.....

.....

Dans la console (la partie du bas de l'IDE), taper le code suivant :

```
>>> message = "Bonjour à tous"
>>> message

>>> type(message)

>>> int(message)
```

Quel est le type de `message` ? .....

Que se passe-t-il quand on a exécuté la ligne `int(message)` ? .....

Que peut-on conclure ? .....

Maintenant que vous savez qu'il existe des types en Python et que vous savez faire des opérations, nous allons nous intéresser aux types des résultats suivant l'opération faite. Par exemple pour connaître le type du résultat d'une addition entre deux entiers (`int`), on peut faire :

```
>>> type(1+1)
```

Opération	Type du résultat avec en entrée	
	2 entiers	2 flottants
Addition	`int`	
Soustraction		
Multiplication		
Division		
Division entière		
Puissance		

4 Un type bien utile : `bool`

Les variables de type `bool`, ou **variables booléennes** ne peuvent prendre que **deux valeurs** : `True` ou `False`. Ce type de variable est souvent utilisé comme résultat d'une condition. On peut vérifier que les valeurs `True` et `False` sont des valeurs booléennes en tapant les deux commandes suivantes dans la console.

```
>>> type(True)

>>> type(False)
```

Ces deux commandes donnent le même résultat : `<class 'bool'>`.

Afin de tester si une condition est vraie ou fausse, nous allons utiliser ce que l'on appelle des **opérateurs de comparaison**. Vous en avez déjà vu comme le « > » ou « < » mais il en existe d'autres. Soit le tableau suivant, à vous de le remplir avec les bons opérateurs. Si vous avez du mal à remplir le tableau, vous pouvez vous aider de ce site : https://realpython.com/python-operators-expressions/.

Type de comparaison	Opérateur	Exemple	Signification de la comparaison
Égal		a ..... b	Vrai si la valeur de a est égale à la valeur de b. Faux sinon
Non-égal (ou différent)		a ..... b	Vrai si la valeur de a n'est pas égale à la valeur de b. Faux sinon
Inférieur		a ..... b	Vrai si la valeur de a est **strictement inférieure** à la valeur de b. Faux sinon
Inférieur ou égal		a ..... b	Vrai si la valeur de a est inférieure ou égale à la valeur de b. Faux sinon
Supérieur		a ..... b	Vrai si la valeur de a est **strictement supérieure** à la valeur de b. Faux sinon
Supérieur ou égal		a ..... b	Vrai si la valeur de a est supérieure ou égale à la valeur de b. Faux sinon

Entrer les commandes suivantes dans la console et noter les résultats obtenus :

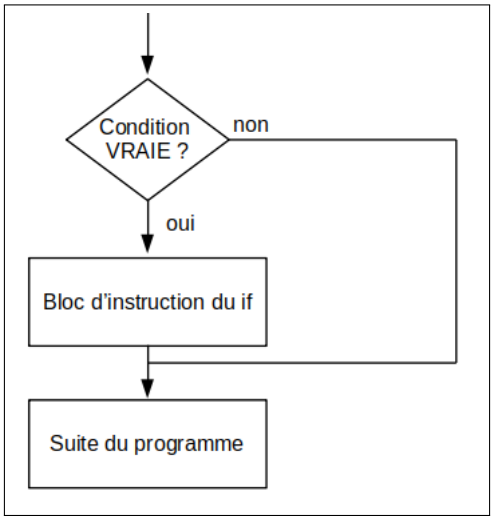
```
>>> a = 2

>>> b = 3.5

>>> a == b

>>> a < b

>>> type(a >= b)

>>> 3 < b < 5

>>> c = 'a'

>>> c == a

>>> c == 'a'

>>> 'a' == c
```

Ligne	Résultat	Ligne	Résultat
1		11	
3		13	
5		15	
7		17	
9		19	

5 L'instruction `if`

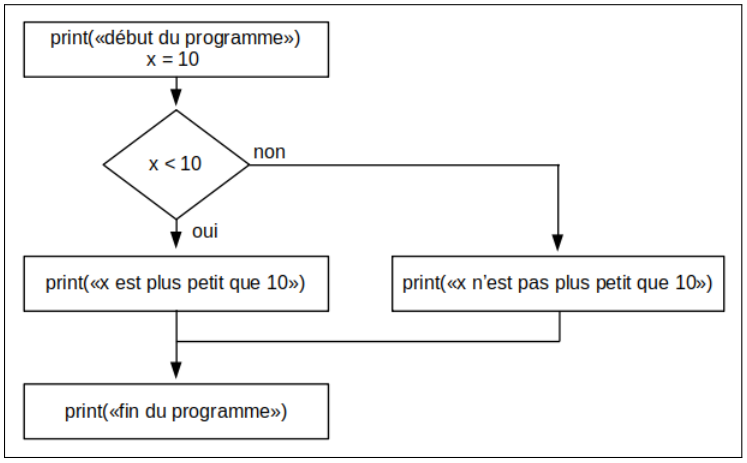
Il a été dit précédemment que les valeurs booléennes étaient utilisées comme résultat d'une condition. Une condition peut se retrouver dans un bloc d'instruction `if`.

L'instruction `if` est utilisée pour exécuter des instructions en fonction d'une condition. On peut illustrer le fonctionnement d'une telle instruction par le schéma suivant :



Nous pouvons donc imaginer un parallèle entre le code et le schéma comme ceci :

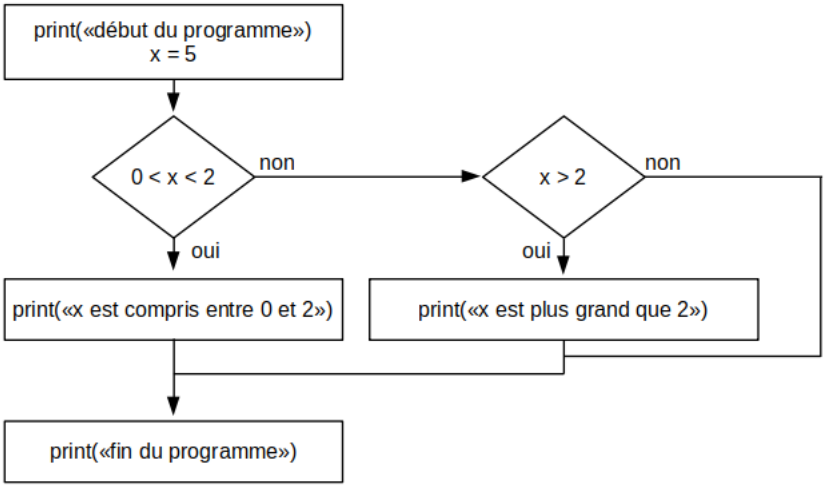
```
print("début du programme")
x = 10
if x < 10 :
    print("x est plus petit que 10")
else:
    print("x n'est pas plus petit que 10")
print("fin du programme")
```



Vous pouvez remarquer que le bloc appartenant à l'instruction `if` (`print("x est plus petit que 10")` et le bloc appartenant à l'instruction `else` (`print("x n'est pas plus petit que 10")`) sont **indentés (décalés)** par rapport au reste du programme. Il est important de le faire pour que le programme marche!

Pour les exemples suivants, établir faire un parallèle entre le code et le schéma.

```
print("début du programme")
x = 10
y = 5
if x < y :
    print("x est plus petit que y")
    x = 15
else:
    print("x n'est pas petit que y")
y = 10
print("fin du programme")
```



En plus des instructions `if` et `if... else`, il existe une autre instruction : `if... elif... else`. Pour plus d'informations sur ce bloc rendez-vous sur Internet. Enfin, il est possible d'imbriquer les instructions entre elles, par exemple une instruction `if` peut être dans une autre instruction `if`. Si ce n'est pas clair, vous pouvez consulter ce site : https://www.cours-gratuit.com/tutoriel-python/tutoriel-python-comment-crer-des-instructions-imbriques-en-python#_Toc51882367

6 C'est à vous!

EXERCICE 1
Pour cet exercice, créer un fichier exercice_1.py

Pour ce premier exercice, nous allons faire un exemple simple avec une instruction `if... elif... else`. Le programme devra initialiser deux variables (les variables doivent contenir des nombres). Après ça, nous allons distinguer 3 cas :

- Si a < b : alors le programme devra afficher « a est plus petit que b »
- Sinon si a = b : alors le programme devra afficher « a est égal à b »
- Sinon : alors le programme devra afficher « a est plus grand que b »

EXERCICE 2
Pour cet exercice, créer un fichier exercice_2.py

En Python, l'opération `x%y` indique le resultat de la division de `x` par `y`. On peut résumer cet opérateur a :

Si y divise x alors `x%y == 0`
Si y ne divise pas x alors `x%y != 0`

Écrire un programme qui indique si un nombre en divise un autre.

EXERCICE 3
Pour cet exercice, créer un fichier exercice_3.py

En utilisant ce qui a été vu à l'exercice précédent, écrire un programme qui indique si un nombre est pair ou impair.

EXERCICE 4
Pour cet exercice, créer un fichier exercice_4.py

On souhaite écrire un programme qui va donner le prix à payer en fonction de l'âge du spectateur. On donne le tableau suivant :

Âge du spectateur	Prix à payer
10 ans et moins	Gratuit
11 - 14 ans	5€
15 - 17 ans	7€
18 - 29 ans	10€
30 - 54 ans	15€
55 - 99 ans	10€
100 ans et plus	Gratuit

Vous veillerez à utiliser une instruction `if... elif... else`.

EXERCICE 5
Pour cet exercice, il faudra modifier le fichier exercice_4.py

Des petits malins s'amusent à détraquer votre programme écrit précédemment : ils ont trouvé la faille ! En effet, les fraudeurs s'amusent à entrer un âge négatif afin de ne pas payer le cinéma. Corriger le programme afin que cet abus ne soit plus possible

RÉSUMÉ :

- Les variables ont des **types** en fonction de la valeur qu'elles contiennent.
- La commande Python `type(...)` permet d'afficher le type de la variable mise entre parenthèses.
- Il existe une structure conditionnelle qui permet de faire des tests et de n'exécuter qu'une partie du code en fonction du résultat de ce test.