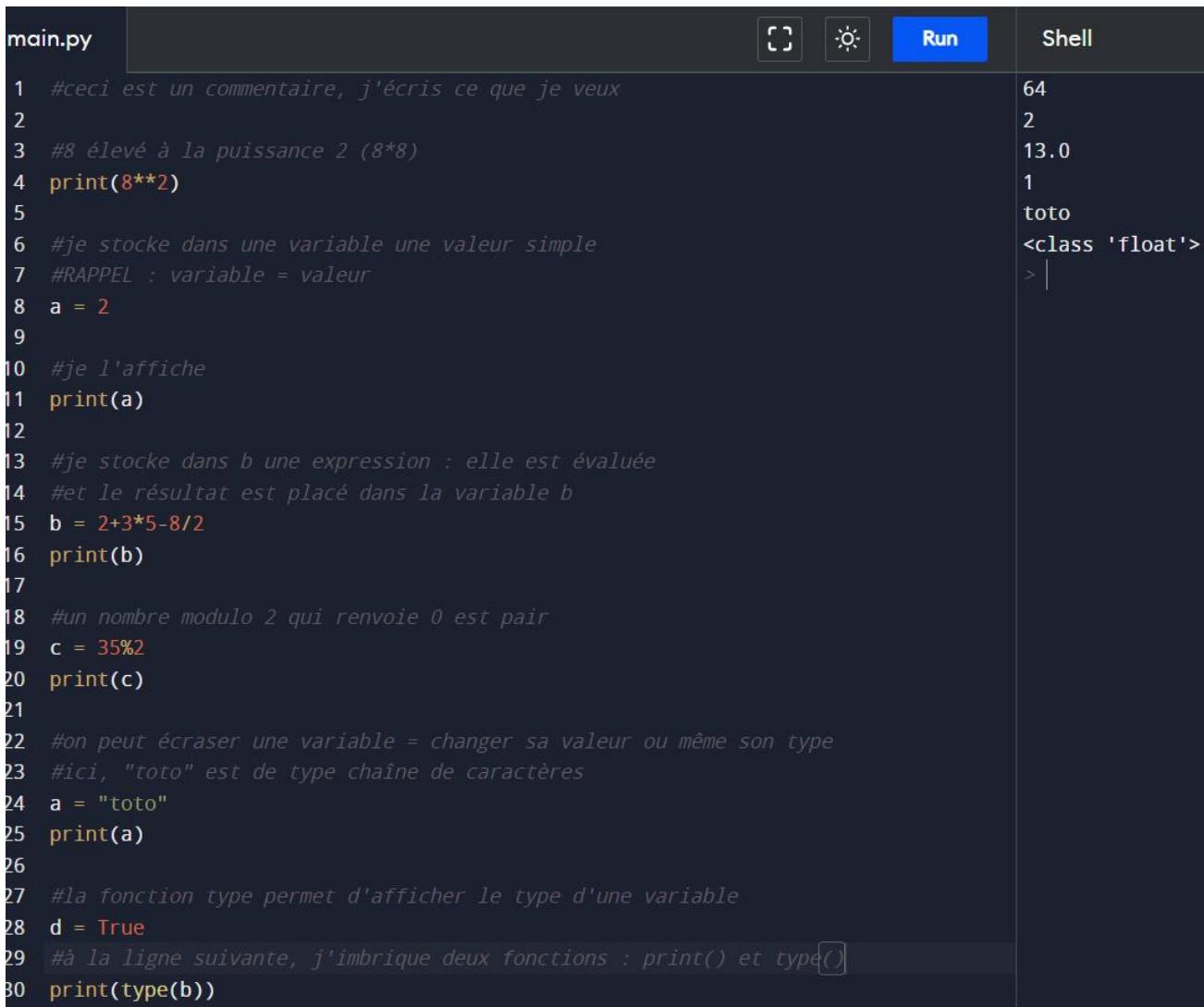


Type de document	Cours et exercices	Classe	1 ^{re}	Durée	2h	Date	06/10/2022
Thème et contenu(s)	Langages et programmation – Représentation des données						
Capacités attendues	Bases de Python : création de scripts simples, fonctions conditionnelles						
Prérequis	Connaître les 4 types de base : Boolean, Integer, Float, String						
Description	Cours et exercices, idéalement en salle informatique devant les machines						

On travaillera désormais exclusivement avec des fichiers .py, mais plus avec la console

I) Python comme calculatrice, premières fonctions

Utilisez l'environnement de votre choix pour refaire via un fichier Python les opérations que l'on faisait dans la console. Utilisez la fonction `print()` et entraînez-vous à écrire des commentaires. Sur les environnements en ligne, ce fichier est matérialisé de différentes manières : par exemple sur Programiz, c'est la partie gauche de l'écran par défaut et le fichier s'appelle **main.py**



```

main.py
1 #ceci est un commentaire, j'écris ce que je veux
2
3 #8 élevé à la puissance 2 (8*8)
4 print(8**2)
5
6 #je stocke dans une variable une valeur simple
7 #RAPPEL : variable = valeur
8 a = 2
9
10 #je l'affiche
11 print(a)
12
13 #je stocke dans b une expression : elle est évaluée
14 #et le résultat est placé dans la variable b
15 b = 2+3*5-8/2
16 print(b)
17
18 #un nombre modulo 2 qui renvoie 0 est pair
19 c = 35%2
20 print(c)
21
22 #on peut écraser une variable = changer sa valeur ou même son type
23 #ici, "toto" est de type chaîne de caractères
24 a = "toto"
25 print(a)
26
27 #la fonction type permet d'afficher le type d'une variable
28 d = True
29 #à la ligne suivante, j'imbrique deux fonctions : print() et type()
30 print(type(b))

```

II) Expressions booléennes

On utilise aussi Python pour évaluer des expressions non-numériques, c'est-à-dire dont le résultat est de type booléen : **True** ou bien **False**. On peut pour cela faire appel aux opérateurs de comparaison (`<`, `>`, `<=`, `>=`, `==`, `!=`) et/ou aux opérateurs logiques (**or**, **not()**, **and**, `^`). Remarque : `not()` est une fonction, elle s'applique à tout ce qui est entre parenthèses (elle symbolise la « barre » de la notation algébrique qui peut s'employer pour inverser un seul élément ou bien une expression).

```

main.py
1 print(2<=3)
2
3 #on initialise deux variables a et b
4 a = 5
5 b = 4
6
7 #on teste si elles sont égales
8 print(a==b)
9
10 #on crée une variable c qui contient le résultat de l'évaluation
11 #de l'expression logique a!=b
12 c = a != b
13
14 #on l'affiche, en mélangeant du texte et des valeurs de variables
15 print("c contient la valeur : ",c)
16
17 #on affiche le type de c
18 print("c est de type : ", type(c))
19
20 #on montre que la représentation des flottants est parfois ambiguë
21 print(1+2==3)
22 print(0.1+0.2==0.3)
23
24 #opérateurs logiques
25 print(not(True))
26 print(not(False))
27 print(True or False)
28 print(False and False)
29 print(True and False)
30 print(True and True)

```

The screenshot shows a Python code editor with a script named `main.py`. The code contains various print statements demonstrating different aspects of Python's logical operators and type handling. Colored arrows point from specific print statements to their corresponding output values in the shell pane, highlighting the results of floating-point comparisons and the behavior of the `not`, `or`, and `and` operators.

III) La fonction `input()` pour communiquer avec l'utilisateur

Cette fonction possède comme **paramètre optionnel une chaîne de caractères** qui sera affichée, par exemple pour demander à l'utilisateur d'entrer quelque chose au clavier. Toute entrée au clavier est vue comme une chaîne de caractères par Python. **Pour stocker la valeur entrée par l'utilisateur, on a besoin d'utiliser une variable.** La capture suivante est un exemple d'utilisation de la fonction `input()`. Elle stocke le texte entré par l'utilisateur dans la variable `s`.

```

main.py
1 s = input("Entrez une chaîne de caractères : ")
2 print("vous avez entré la chaîne : ",s)

```

The screenshot shows a Python code editor with a script named `main.py`. The code uses the `input()` function to get user input and store it in the variable `s`. The output shows the user entering the value `56` and the program printing it back.

IV) Échange de deux variables

Pour échanger le contenu de deux variables, **on doit obligatoirement passer par une variable « tampon »** qui stocke temporairement le contenu d'une des deux variables. Si le tampon n'existe pas, en plaçant `b` dans `a` on perdrait définitivement le contenu de `a`. Les codes suivants montrent :

- un échange de variables connues à l'avance (définies par le programmeur)
- un échange de variables entrées par l'utilisateur grâce à la fonction `input()`

Comme toujours, à gauche se trouve le programme principal (fichier `main.py`), et à droite le résultat de l'exécution de ce programme.

<pre> main.py</pre>	   Run	Shell <pre> 1 #initialisation des deux variables 2 a = 45.9 3 b = "toto" 4 5 #test d'affichage en mélangeant texte et variables 6 print("avant échange, a vaut : ",a,"et b vaut : ",b) 7 8 #stockage de a dans une variable tampon 9 tmp = a 10 11 #on met b dans a 12 a = b 13 14 #tmp contient la valeur de a 15 #on le place donc dans b 16 b = tmp 17 18 #affichage 19 print("après échange, a vaut : ",a,"et b vaut : ",b)</pre>
---------------------	--	---

<pre> main.py</pre>	   Run	Shell <pre> entre une valeur : 9 entre une autre valeur : 45 avant échange, a vaut : 9 et b vaut : 45 après échange, a vaut : 45 et b vaut : 9 > </pre>
---------------------	---	--

On pourrait résumer ceci par l'**algorithme** suivant (un algorithme est une représentation abstraite d'un programme, indépendante des langages de programmation, plus proche du langage naturel) :

- Créer une variable a qui reçoit une entrée clavier
- Créer une variable b qui reçoit une entrée clavier
- Créer une variable tampon et y mettre a
- Mettre b dans a
- Mettre tampon dans b