Type de document	Cours et exercices	Classe	1re	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 — Run — Shell

```
#ceci est un commentaire, j'écris ce que je veux

#8 élevé à la puissance 2 (8*8)
print(8**2)

#je stocke dans une variable une valeur simple
#RAPPEL : variable = valeur
a = 2

#je l'affiche
print(a)

#je stocke dans b une expression : elle est évaluée
#et le résultat est placé dans la variable b
b = 2+3*5-8/2
print(b)

#un nombre modulo 2 qui renvoie 0 est pair
c = 35%2
print(c)

#on peut écraser une variable = changer sa valeur ou même son type
#ici, "toto" est de type chaîne de caractères
a = "toto"
print(a)

#la fonction type permet d'afficher le type d'une variable
d = True
#à la ligne suivante, j'imbrique deux fonctions : print() et type()
print(type(b))
```

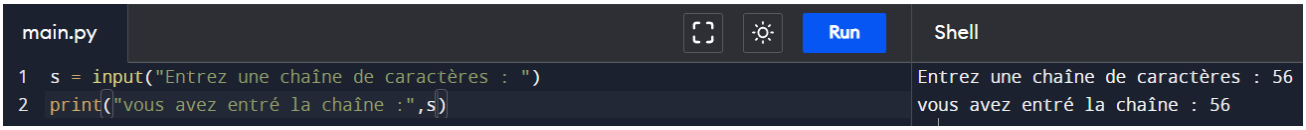
```
64
2
13.0
1
toto
<class 'float'>
>
```

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).

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.



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'existait 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.

main.py — Run

```
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)
```

Shell

```
avant échange, a vaut : 45.9 b vaut : toto
après échange, a vaut : toto b vaut : 45.9
>
```

main.py — Run

```
1  #initialisation des deux variables
2  a = input("entre une valeur : ")
3  b = input("entre une autre valeur : ")
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)
```

Shell

```
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
>
```

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