Les fondamentaux

Objectifs

- comprendre la notion de type de variable
- savoir transtyper une variable
- savoir afficher et saisir des données
- savoir concaténer des chaînes de caratères
- découvrir les structures de contrôle : CHOISIR, PARCOURIR, RÉPÉTER
- savoir définir et appeler une fonction

Connaissances et savoir-faire à maîtriser

1) Type d'une variable

Une variable d'un programme possède un **type** parmi les types principaux suivants :

- *chaîne de caratères*, de l'anglais *string*, abrégée **str** en Python
- *numérique entier*, noté **int** en Python
- *numérique à virgule*, on parle de *nombres flottants*, noté **float** en Python
- *booléen*, noté **bool** en Python qui ne peut prendre que deux valeurs `True` ou `False`

Remarque 1 : la fonction `type(variable)` renvoie le type de la variable passée en paramètre.

*Remarque 2 : il est possible de **transtyper** une variable c'est-à-dire de lui attribuer un type différent de celui d'origine en l'enveloppant dans le type désiré. Par exemple, l'instruction `str(1.0)` transtype un float en une string.*

2) Afficher des données

On affiche une chaîne de caratères ou une variable à l'écran en utilisant la fonction `print` .

```
var = 2
print(var) # affiche 2
print("bonjour") # affiche bonjour
print("V =", var) # affiche V = 2
```

Deux chaînes de caractères peuvent être assemblées en une seule (on dit **concaténées**) en utilisant l'opérateur +.

```
print("Hello" + "World") # affiche HelloWorld
var = 2
print("V = " + str(var)) # affiche V = 2
```

Remarque 3 : dans le dernier exemple, il est nécessaire de transtyper la variable numérique en une string avant la concaténation sous peine de générer une erreur lors de l'exécution.

3) Saisir des données

On demande à l'utilisateur de saisir des données en utilisant la fonction `input` . Celle-ci prend en paramètre le message à afficher à l'utilisateur et renvoit **toujours** un string. Par conséquent, pour les saisies **numériques**, il est nécessaire de transtyper la valeur de retour lors de l'appel à `input`.

```
moyT1 = float(input("Quelle est votre moyenne du premier trimestre ?"))
```

Dans ce dernier exemple, `moyT1` est de type float.

4) Les stuctures de contrôle

Lors de l'écriture d'un programme informatique, le programmeur dispose de commandes lui permettant de modifier le flux d'instructions à exécuter par la machine. Ces structures de contrôle du flux sont plus ou moins nombreuses suivant les langages et se répartissent en trois catégories.

a. Structure pour CHOISIR

Elle est utilisée lorsque l'on doit choisir quelle instruction exécuter en fonction du résultat d'un ou plusieurs tests conditionnels.

```
if moyenne == 10:
    print("Vous avez le bac de justesse")
elif moyenne < 10:
    print("Vous êtes peut-être au rattrapage")
else
    print("Vous avez peut-être le bac avec mention")
```

b. Structure pour PARCOURIR

Elle est aussi appelée structure de boucle bornée ou **boucle FOR** et est utilisée lorsque l'on doit exécuter un certain nombre de fois les mêmes instructions. En Python, on l'utilise souvent avec la fonction `range` qui permet d'énumérer tous les entiers d'un intervalle.

```
for i in range(1, 11):
    print(i)
```

Remarque 4 *: on notera un détail très important, l'énumération précédente liste les entiers de 1 à 10 et non de 1 à 11.*

c. Structure pour RÉPÉTER

Aussi appelée structure de boucle non bornée ou **boucle WHILE**, elle permet de répéter des instructions **tant que** le résultat d'un test conditionnel est vraie.

```
i = 1
while i < 11:
    print(i)
    i = i + 1
```


*Remarque 5 : encore un détail important, si on oublie d'incrémenter la valeur de i, le programme entre dans une **boucle infinie** qui l'empêche de se terminer correctement.*

d. Les fonctions

Une fonction permet de **factoriser** du code, autrement dit d'exécuter une même séquence d'instructions plusieurs fois en réalisant un **appel de fonction**.
On définit une fonction par l'utilisation du mot-clé `def` suivi du nom de la fonction. Celle-ci peut prendre des variables en **paramètres** et peut retourner une valeur via l'emploi du mot-clé `return`.

```
def f(x):
    return x**2

for x in range(1, 11):
    print(f(x))
```

Remarque 6 : la fonction f définit ci-dessus prend une valeur numérique de x en paramètre et retourne la valeur de x^2 (on notera la notation double étoile pour élever à la puissance). Une boucle FOR permet alors d'afficher le tableau de valeurs de x^2 pour des valeurs entières de x variant de 1 à 10.

Remarque 7 : une fonction peut ne prendre aucun paramètre, elle peut aussi ne rien retourner comme on le verra dans l'exercice 3.

Remarque 8 : une fonction peut ne rien faire via l'emploi du mot-clé `pass`.

Exercices à réaliser

Pour chaque exercice, ouvrez le fichier mentionné dans l'énoncé et lisez les consignes qui reprennent ou complètent les notions de ce document. Quelques conseils :

- respecter les niveaux d'indentation
- utiliser des noms de variables explicites
- utiliser des commentaires afin de documenter votre programme via le symbole #
- user et abuser des `print` pour débugger, les mettre en commentaire lorque votre programme fonctionne
- bien analyser les messages d'erreur de Python qui vous renseignent sur la nature de l'erreur lors de l'exécution

Exercice 1
Ouvrir le fichier `TP-fondamentaux-exo1.py`, lire les consignes et répondre aux questions suivantes :

1. Demander à l'utilsateur de saisir son Nom/Prénom puis afficher un message de bienvenue.
2. Demander à l'utilisateur de saisir ses trois moyennes de trimestre puis calculer sa moyenne générale.
3. Indiquer enfin à l'utilisateur s'il a le bac (avec ou sans mention) ou s'il est au rattrapage.

Exercice 2
Ouvrir le fichier `TP-fondamentaux-exo2.py`, lire les consignes et répondre aux questions suivantes :

1. Utiliser une boucle FOR pour afficher la table de multiplication de votre choix.
2. Utiliser deux boucles FOR imbriquées pour afficher TOUTES les tables de multiplication de 1 à 10.

Exercice 3

Ouvrir le fichier `TP-fondamentaux-exo3.py` et lire les consignes. L'objectif de cet exercice est d'afficher à l'utilisateur en continu un menu listant les possibilités qui s'offrent à lui tant que celui-ci n'a pas décidé de quitter l'application.

1. Créer une fonction `afficherMenu()` qui affiche le menu ci-dessous et qui retourne le choix de l'utilisateur.

```
********** Bienvenue **********
Que voulez-vous faire ?
1 - Afficher une table de multiplication de votre choix
2 - Afficher toutes les tables de multiplication de 1 à 10
3 - Quitter l'application
Votre choix :
```

2. Utiliser une boucle WHILE pour afficher le menu tant que l'utilisateur n'a pas décidé de quitter l'application.
3. Compléter les fonctionnalités du menu en utilisant le code de l'exercice précédent. Vous veillerez à définir une fonction pour chaque fonctionnalité.

Compétences à valider

Les fondamentaux		☀	⛅	🌧	⛈
Type	J'ai compris la notion de type de variable				
	Je sais transtyper une variable				
Afficher	Je sais afficher des messages à l'utilisateur : chaînes de caractères et/ou variables				
	Je sais afficher un message en utilisant la concaténation et le transtypage				
Saisir	Je sais demander à l'utilsateur de saisir des données				
	Je sais transtyper les données saisies en vue de les traiter correctement				
Contrôler	Je sais utiliser une structure de type CHOISIR				
	Je sais utiliser une structure de type PARCOURIR				
	Je sais utiliser une structure de type RÉPÉTER				
	Je sais définir et appeler une fonction				