La Plateforme

Python

Job 00 : Installation

Pour réaliser les exercices de ce sujet, vous allez avoir besoin de Python 3

Si tu as une distribution Mac OS
Rendez vous sur ce lien et téléchargez l'installer pour windows 64bit de python 3.10.8

Si tu as une distribution Linux
Entrez la commande
sudo apt-get install python3

Sur une distribution windows :
Rendez vous sur ce lien et téléchargez l'installer pour windows 64bit de python 3.10.8
Lancez l'installation et cliquez sur "Install Now" et cochez la case **"Add Python to PATH"**

Pour faire les exercices du projet, installer un IDE serait une bonne idée. Nous vous recommandons de regarder du côté de Visual Studio Code

Job 01 : Les variables

Créez un script job01.py.
Déclarez une variable école, et attribuez lui la valeur “La Plateforme_” (avec les mêmes majuscules et le ‘_’ à la fin). A l’aide d’une fonction système, affichez cette variable dans le terminal.

Exemple

```
$ py job01.py
La Plateforme_
```

Job 02 : La saisie utilisateur

Créez un script job02.py qui dans un premier temps va afficher “Bonjour, comment t’appelles tu ? ”
L’utilisateur devra ensuite entrer son prénom.
Déclarez une variable prenom, qui prendra la valeur saisie par l’utilisateur.
A l’aide d’une fonction system, “Bonjour *prenom*” (prenom étant la valeur de la variable du même nom)

Exemple

```
$ py job02.py
Bonjour, comment t’appelles tu ? La Plateforme_
Bonjour La Plateforme_
```

Job 03 : Les types et conditions

Créez un script job03.py qui dans un premier temps va afficher la phrase “Bonjour, quel âge as tu ? ”
L’utilisateur devra ensuite entrer son âge.
Déclarez une variable “âge”, qui prendra la valeur saisie par l’utilisateur.
A l’aide d’une fonction system, affichez “Tu es mineur” si l’âge est inférieur à 18, et “Tu es majeur” si l’âge est supérieur ou égal à 18.

Exemple

```
$ py job03.py
Quel âge as tu ?
10
Tu es mineur
```

```
$ py job03.py
Quel âge as tu ?
18
Tu es majeur
```

Job 04 : Les boucles while

Créez un script job04.py
L'utilisateur devra entrer une valeur en input.
A l'aide d'une boucle while et d'une fonction system, affichez nombres se trouvant de 0 (inclus) à l'input (inlus).

Job 05 : Les boucles For

Créez un script job05.py
L'utilisateur devra entrer deux valeurs en input.
A l'aide d'une boucle for et d'une fonction system, affichez uniquement les nombres se trouvant entre l'input 1 et l'input 2**. Le programme doit toujours partir de l'input 1 et aller jusqu'à l'input 2.**
Si les deux sont égaux, le programme devra écrire "Valeurs égales".

Exemple

```
$ py job04.py
Valeur 1 : 1
Valeur 2 : 4
2
3


$ py job04.py
Valeur 1 : 4
Valeur 2 : 1
3
2


$ py job04.py
Valeur 1 : 4
Valeur 2 : 4
Valeurs égales
```

Job 06 : Les boucles infinies

Créez un script job06.py
Lorsque l'utilisateur va lancer le script, un prompteur devra s'afficher ">", et le programme devra attendre un input.

Si l'input entré est "Bonjour", le programme devra répondre "Bonjour à toi"
Si l'input entré est "Au revoir", le programme devra quitter

Exemple

```
$ py job05.py
>Bonjour
Bonjour à toi
>test
test
>Au revoir
```

Job 07 : Fizz Buzz

Écrire un programme qui parcourt les nombres entiers de 1 à 100. Pour les multiples de trois, afficher "Fizz" au lieu du nombre et pour les multiples de cinq afficher "Buzz". Pour les nombres qui sont des multiples de trois et cinq, afficher "FizzBuzz".

Job 08 : Rectangle

Écrire un programme rectangle.py qui attend deux inputs : la largeur puis la hauteur. Votre programme devra ensuite dessiner un rectangle avec les éléments suivants :

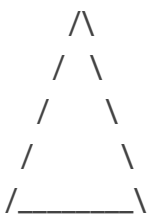
- des “|” pour dessiner les côtés droite et gauche
- des “-” pour dessiner le haut et le bas
- des espaces pour remplir le rectangle

Exemple pour un input de 10 en largeur et 3 en hauteur

```
|--------|
|        |
|--------|
```

Job 09 : Triangle

Écrire un programme triangle.py qui affiche dans le terminal un triangle avec des '_', des '\' et des '/'
en fonction de l'input entré, qui représentera la hauteur.
Exemple si l'input entré est 5



Job 10 : Ecriture et lecture de fichier

Écrire un programme job10.py qui demande à l'utilisateur d'entrer un texte. Le programme devra récupérer ce texte, et l'écrire dans un fichier "**output.txt**".
Écrire un programme qui lit le contenu de fichier "**output.txt**", et qui l'écrit dans le terminal.

Job 11 : Traitement d'un fichier 1.0

Créer un programme job11.py qui parcourt le contenu du fichier "**domains.xml**" et qui compte le nombre d'extension de domaines qui s'y trouvent (.com, .net, etc ...).

Job 12 : Traitement d'un fichier 2.0

Créer un programme job12.py qui parcourt le contenu du fichier "**data.txt**" et qui compte le nombre nombre de mots (sans caractère spéciaux) qui s'y trouvent.

Job 13 : Traitement d'un fichier 3.0

Créer un programme job13.py qui demande à l'utilisateur de renseigner un nombre entier. Le programme devra alors parcourir le contenu du fichier "**data.txt**" compter le nombre de mots de la taille renseignée qui s'y trouvent.

Job 14 : Fonctions

Créer un programme job14.py. Le programme devra contenir une fonction myFunction() qui écrit dans le terminal “je suis une fonction”.

Job 15 : Fonctions et paramètres

Créer un programme job15.py. Le programme devra contenir une fonction qui prend en paramètres une variable *prenom* et une variable *nom*.
La fonction écrira “Bonjour *prenom nom*” dans le terminal.

Job 16 : Fonctions et paramètres 2.0

Créer un programme job16.py. Le programme devra contenir une fonction qui prend en paramètres un nombre de paramètres indéfini.
La fonction écrira tous les paramètres dans le terminal.

Job 17 : Listes

Créer un programme job17.py. Le programme devra contenir une fonction qui prend en paramètres un nombre de paramètres indéfini (uniquement des nombres).
La fonction devra :

- Remplir une liste *myList* contenant tous ces paramètres.
- Parcourir et afficher dans le terminal uniquement les nombres pairs de la liste.

Job 18 : Trier avec sort

Créer un programme job18.py. Le programme devra contenir une fonction qui prend en paramètres un nombre de paramètres indéfini (uniquement des nombres).
La fonction devra :

- Remplir une liste *myList* contenant tous ces paramètres.
- Trier par ordre croissant la liste à l'aide de la fonction sort
- Afficher la liste dans un terminal

Job 19 : Trier ~~avec sort~~

Créer un programme job19.py. Le programme devra contenir une seule fonction qui prend en paramètres un nombre de paramètres indéfini (uniquement des nombres).
La fonction devra :

- Remplir une liste *myList* contenant tous ces paramètres.
- Trier par ordre croissant la liste~~ à l'aide de la fonction sort~~
- Afficher la liste dans un terminal

Job 20 : Valeur de retour

Créer un programme job20.py. Le programme devra contenir une seule fonction nommée myAddition qui prend en paramètres deux nombres et qui retourne le résultat de leur addition.
Vous devrez appeler cette fonction et assigner à une variable le résultat qu'elle retourne.

Job 21 : Trier ~~avec sort~~ 2.0

Créer un programme job21.py. Reprenez l'exercice 19, mais modifiez le de façon à utiliser deux fonctions :

- Une fonction *mySort*, qui prendra en paramètre une liste. Elle retourne une liste avec les valeurs de celle passée en paramètre, triés par ordre croissant.
- Une fonction *myDisplay* qui prendra en paramètre une liste. Elle affichera dans le terminal le contenu de cette liste.

Job 22 : Chaîne de caractères

Créer un programme job22.py. Il devra prendre un premier input qui sera une chaine de caractère, puis un deuxième. Si le deuxième input est :

- "upper" : une fonction "myUper" devra être appellée. Cette fonction prend en paramètre une chaine de caractère, et retourne cette chaîne en majuscule.
- "lower" : une fonction "myLower" devra être appellée. Cette fonction prend en paramètre une chaine de caractère, et retourne cette chaîne en minuscule.
- "title" : une fonction "myTitle" devra être appellée. Cette fonction prend en paramètre une chaine de caractère, et retourne cette chaîne avec une majuscule au début de chaque mot.
- "clean" : une fonction "myClean" devra être appellée. Cette fonction prend en paramètre une chaine de caractère, et retourne cette chaîne en enlevant tous les espaces inutiles (en début et en fin de chaine de caractère, et les doubles espaces).

Aucune fonction system n'est autorisée. Vous pouvez cependant les tester afin d'étudier leur fonctionnement.
A la fin de votre programme, la chaine de caractère retournée par la fonction devra être affichée sur le terminal.

Rendu

Le projet est à rendre sur github **https://github.com/prenom-nom/starter-python**

Vous devrez y mettre chacun de vos scripts, directement à la racine de ce dossier.

Base de connaissances

- Les variables Python
- Les boucles Python

- Les conditions Python
- L'indentation en Python

- Lecture et écriture de fichier Python