Programmation Python

Fiche TDM

Solemane Coulibaly (solemane.coulibaly@gmail.com)

decembre 2024

Partie 1

Écrire un programme qui permet de :

1. extraire d'une liste d'entiers les valeurs consécutives qui se suivent, par exemple

```
[3,8,9,10,9,15,16] -> [(8,9),(9,10),(15,16)]
```

2. trouver la liste des indices d'une valeur dans une liste, par exemple, 3 dans `[2,3,4,3,6,3]` renvoie `[1,3,5]`
3. supprime toutes les occurrences d'une valeur spécifique dans une liste, en modifiant la liste originale sans créer de nouvelle liste. Appelons doublon dans une liste tout élément y figurant au moins deux fois. Ainsi la liste `[3, 1, 4, 1, 5, 9,2, 6, 5, 3, 5]` contient trois doublons : 1 et 3 qui y figurent deux fois, et 5 qui y figure trois fois. Le résultat sera [4, 9,2, 6]
4. comptez (dans un dictionnaire) le nombre d'occurrence de chaque élément contenu dans une liste. Par exemple `["a","c","c","b","a"]` -> `{"a":2, "b":1, "c":2}`

Partie 2

Exercice 1

Écrire un programme une fonction qui calcule la somme des éléments d'une liste de nombres. Et un autre qui permet de multiplier tous les éléments d'une liste de nombres. Tester les deux fonctions.

Exercice 2

Écrire une fonction qui prends en arguments deux nombres entiers a et b et qui renvoie un tuple formé de : - Le quotient q de la division euclidienne de a par b - Le reste r de la division euclidienne de a par b

Exercice 3

Ecrire une fonction qui trouve le deuxième plus grand élément dans une liste.

```
entrée : [92, 16, 25, -7, 100, 150, 0, 200, -300]
en sortie : 150
```

Exercice 4

Écrivez une fonction Python `ecart_type(valeurs)` qui calcule l'écart type d'une liste de valeurs. L'écart type est donné par la formule :

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2}$$

où μ est la moyenne des valeurs et N est le nombre de valeurs.

Exercice 5

1. Écrivez une fonction `creer_etudiant` qui prend en paramètres le nom, l'âge, la matière principale et une liste de notes, et retourne un dictionnaire représentant l'étudiant.

Voici la signature de cette fonction :

```
def creer_etudiant(nom, age, matiere_principale, notes):
return {
    "nom": ???,
    "âge": ???
    ....
}
```

Le paramètre `moyenne` de la fonction calcule la moyenne des notes de l'étudiant dans une matière.

2. Écrivez une fonction `afficher_etudiant` qui prend en paramètre un dictionnaire représentant un étudiant et affiche ses informations de manière lisible.

Exercice 6

1. Créez une liste contenant plusieurs dictionnaires représentant des étudiants.

```
liste_etudiants = [
    creer_etudiant("Toto", 20, "Informatique", [15, 16, 17]),
    creer_etudiant("Mory", 22, "Mathématiques", [12, 19, 17]),
    creer_etudiant("Amina", 23, "Physique", [14, 15, 16])
]
```

2. Écrivez une fonction `moyenne_generale` qui prend en paramètre une liste d'étudiants et retourne la moyenne générale de tous les étudiants.

Voici la signature de cette fonction :

```
def moyenne_generale(liste_etudiants):
    ???
```

3. Écrivez une fonction `meilleur_etudiant` qui prend en paramètre une liste d'étudiants et retourne le dictionnaire de l'étudiant ayant la meilleure moyenne.

Exercice 7

Dans un fichier intitulé `app.py`, reprenez les différentes solutions des exercice 1 à 4. Réaliser ensuite un appel

```
#Le fichier app.py
```

```
def demander_nom():
    nom = input("Comment vous appelez-vous ? ")
    return nom
def afficher(*args):
    print(*args)
```

Le nom du module correspond au nom du fichier. Le programme principal dans le fichier `init.py` peut importer et utiliser les fonctions définies dans le module `app.py` et les exécuter.

```
#Le fichier init.py
```

```
from entree_sortie import demander_nom, afficher
    nom = demander_nom()
    afficher("Bonjour", nom)
```

Completer le fichier `app.py`, en ajoutant les différentes fonctions des exercice 1 à 4. Réaliser ensuite un appel de chaque fonction dans un fichier `init2.py`