Lucas Maillet
1ere n°1

– TP3 Python –

Préambule

Premierement, pour surveiller les types des arguments passer dans les fonctions, je crée une fonction « décoratrice » **checkType** qui va réécrire la fonction à appeller avec une nouvelle function qui validera le nombre et les types d'arguments passés en paramètres avant d'appeller la function. On l'appelera à chaque fois que l'on définira une fonction (il est nécessaire d'y définir les types des paramètres) :

tp 3 - tp.py

```python
def checkType(ctx):
    ... # Create a new function wrapping the function
    return wrapped

@checkType # Call the wrapper to overwrite the function
def note_moyenne(notes:  list) -> float:
    ...
```

Ce n'est sans doute pas la technique la plus optimisée mais elle reste pratique pour la lisibilité du code.

Important

Des tests sont directements effectués si le script est directement executé , via l'astuce du test de la variable « __name__ » du module.

D'ailleurs, l'optimisation du code à souvent été sacrifié pour la construction par compréhension, comme demandé.

Aussi toutes les fonctions ont été documenté.

tp 3 - tp.py

```python
if __name__ == "__main__":

    # EXERCICE 1
    ...

    # EXERCICE 2
    ...
```

Exercice n°1

Q1. note_moyennes(notes: list) → float

Cette fonction fait la somme de chaque éléments d'une liste avec la function **sum** puis divise cette somme par sa taille avec la function **len**, et retourne le tout arrondi à 2 chiffres significatifs avec la fonction **round**.

Q2. moyennes_generale(data: list) → float

Cette fonction appele **note_moyennes** avec une liste des moyennes des élèves créer par un générateur qui itère sur la liste passé en arguments.

Q3. top_etudiant(data: list) → float

Cette fonction utilise un générateur itérrant sur la liste d'élèves pour créer un dictionnaire d' élèves par la moyenne de l'élève puis retourne l'élève avec la moyenne la plus grande via une fonction **lambda** utilisant la fonction **max**.

Q4. recherche_moyenne(data: list) → float

Cette fonction itère sur la liste d'élèves et retourne la moyenne d'un élève quand son identifiant correspont a celui passé en paramètres.

Exercice n°2

Q1. nb_ingredients(data: dict, nId: str) → int

Retourne la taille de la liste d'ingrédients d'une recette avec son nom dans le dictionnaire passer en paramètres.

Q2. recette_avec(data: dict, nId: str) → list

Retourne une liste de toutes les recettes créer dans un générateur si l'ingrédient passé en paramètres se trouve dans la liste d'ingrédients de la recette.

Q3. tous_ingredients(data: dict, nId: str) → list

Retourne une liste de toutes les ingrédients de toutes les recettes si l'ingrédient n'est pas déjà dans la liste, sans générateurs mais juste avec des boucles bornées.

Q4. table_ingredients(data: dict) → dict

Utilise un générateur créant un dictionnaire de listes de recettes par un ingrédient y étant nécéssaire, liste d'ingrédients elle même générée par un générateur si l'ingredient est présent dans la recette pour chaque recette du dictionnaire de recettes.

Q5. ingredient_principal(data: dict) → str

Cette fonction utilise un générateur itérrant sur le dictionnaire de recettes pour créer un dictionnaire d'ingrédient par le nombre de fois que l'ingrédient est utilisé puis retourne l'ingrédient le plus utilisé avec une fonction lambda utilisant la fonction max.

Q6. recettes_sans(data: dict, nId: str) → dict

Retourne une copie du dictionnaire de recettes graçe a un générateur qui exclue les recettes contenant l'ingrédient passé en paramètres.