

– TP3 Python –

Préambule

Premierement, pour surveiller les types des arguments passer dans les fonctions, j'utilise une fonction « décorateur » **checkType** (utilisant la fonction **wraps** du module intégré a python **functools**), qui va réécrire une fonction avec une nouvelle fonction qui validera les types d'arguments passés en paramètres et si demandé le type du résultat . On l'appelera à chaque fois que l'on définira une fonction (on peut demander à ne pas regarder le type du résultat) :

```
tp 3 - tp.py
1  from functools import wraps
2
3  def strictType(check_return=True):
4      ... # Create a new function wrapping the older
5      return wrapped
6
7  @strictType() # Call the decorator to overwrite
8  def nb_ingredients(data: dict, nId: str) -> int:
9      ...
```

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

Important

Des tests sont directement effectués si le script est directement exécuté , 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
1  if __name__ == "__main__":
2
3      # EXERCICE 1
4      ...
5
6      # EXERCICE 2
7      ...
```

Exercice n°1

Q1. `note_moyennes(notes: list) → float`

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

Q2. `moyennes_generale(data: list) → float`

Cette fonction appelle **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 itérant 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 génère un dictionnaire des moyennes d'élèves par leur identifiants, puis utilise la méthode **get** pour chercher (s'il existe) la moyenne correspondant a un identifiant.

Exercice n°2

Q1. `nb_ingredients(data: dict, nld: 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, nld: 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, nld: str) → list`

Retourne une liste de toutes les ingrédients de toutes les recettes, en générant une liste de tout les ingrédients dans toutes les recettes, puis en enlèvent les répétitions d'ingrédients avec la fonction **set** puis transforme le tout en liste avec la fonction **list**.

Q4. `table_ingredients(data: dict) → dict`

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

Q5. `ingredient_principal(data: dict) → str`

Cette fonction utilise un générateur itérant 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, nld: str) → dict`

Retourne une copie du dictionnaire de recettes grâce a un générateur qui exclue les recettes contenant l'ingrédient passé en paramètres (via un test).