Marseille, 05/12/2018 - Achref El Mouelhi

TP 1 : Collections, modules et fonctions Lambda avec Python

Exercice

Étant donnée la liste suivante représentant les données personnelles des employés d'une entreprise ainsi que leurs villes d'intervention :

```
personnes = [
    { "type": 'user', "nom": 'Max Mustermann', "age": 25, "villes": ['Marseille', 'Lyon', 'Paris'] },
    { "type": 'admin', "nom": 'John Wick', "age": 45, "villes": ['Paris'] },
    { "type": 'user', "nom": 'Kate Muller', "age": 23, "villes": ['Nantes', 'Lyon', 'Lille', 'Nice'] },
    { "type": 'admin', "nom": 'Bruce Willis', "age": 64, "villes": ['Paris', 'Nantes'] },
    { "type": 'user', "nom": 'Jack Wilson', "age": 35, "villes": ['Marseille', 'Lyon', 'Montpellier'] },
    { "type": 'admin', "nom": 'Carol Smith', "age": 23, "villes": ['Marseille', 'Nice', 'Montpellier'] }
];
```

Répondez aux questions suivantes en utilisant `map`, `filter` et `reduce`.

1. Écrivez une fonction Lambda `get_by_type(type: str) -> Iterable:` qui retourne la liste des personnes selon le type passé en paramètre.
2. Écrivez une fonction Lambda `get_by_ville(ville: str) -> Iterable:` qui retourne la liste des personnes ayant dans `villes` la ville passée en paramètre.
3. Écrivez une fonction Lambda `get_other_ville_than(ville: str) -> Iterable:` qui retourne la liste des personnes n'ayant pas dans `villes` la ville passée en paramètre.
4. Écrivez une fonction Lambda `count_villes() -> Iterable:` qui retourne un tableau d'objets : chaque objet contient le nom d'une personne ainsi que le nombre d'éléments dans la liste villes (voir ci-dessous).

   ```
   [
       {'nom': 'Max Mustermann', 'nbrVilles': 3},
       {'nom': 'John Wick', 'nbrVilles': 1},
       {'nom': 'Kate Muller', 'nbrVilles': 4},
       {'nom': 'Bruce Willis', 'nbrVilles': 2},
       {'nom': 'Jack Wilson', 'nbrVilles': 3},
       {'nom': 'Carol Smith', 'nbrVilles': 3}
   ]
   ```

5. Écrivez une fonction Lambda `get_by_villes_number(nbr: int) -> Iterable:` qui retourne les nom des personnes dont le nombre de villes d'intervention correspond au paramètre `nbr`.
6. Écrivez une fonction Lambda `count_character_in_villes() -> Iterable:` qui retourne un tableau d'objets : chaque objet contient le nom d'une personne ainsi que le nombre total de caractères de ses villes (voir ci-dessous).

   ```
   [
       {'nom': 'Max Mustermann', 'total': 18},
       {'nom': 'John Wick', 'total': 5},
       {'nom': 'Kate Muller', 'total': 19},
       {'nom': 'Bruce Willis', 'total': 11},
       {'nom': 'Jack Wilson', 'total': 24},
       {'nom': 'Carol Smith', 'total': 24}
   ]
   ```

7. Écrivez une fonction Lambda `count_by_ville(ville: str) -> int:` qui retourne le nombre de personnes qui interviennent dans la ville passée en paramètre (ici c'est 3 pour `Marseille` par exemple).

8. Écrivez une fonction Lambda `find_having_max_ville() -> int:` qui permet de retourner le nombre max de villes d'intervention qu'un employé a pu avoir (ici c'est 4).
9. Écrivez une fonction Lambda `get_all_villes() -> Iterable:` qui permet de retourner un tableau de toutes les villes (sans doublons).
10. Écrivez une fonction Lambda `get_avg_age(ville: str) -> int:` qui retourne la moyenne d'age (d'employés) de la ville passée en paramètre.
11. Écrivez une fonction Lambda `get_avg_age_by_ville() -> Iterable:` qui retourne la moyenne d'age de chaque ville d'intervention sous forme d'un tableau d'objet.

```
[
    {'ville': 'Marseille', 'moyenneAge':
       27.666666666666668},
    {'ville': 'Lyon', 'moyenneAge': 27.666666666666668},
    {'ville': 'Paris', 'moyenneAge': 44.666666666666664},
    {'ville': 'Nantes', 'moyenneAge': 43.5},
    {'ville': 'Lille', 'moyenneAge': 23.0},
    {'ville': 'Nice', 'moyenneAge': 23.0},
    {'ville': 'Montpellier', 'moyenneAge': 29.0}
]
```

12. Écrivez une fonction Lambda `get_younger_by_ville() -> Iterable:` qui retourne l'age de la personne la plus jeune par ville (voir ci-dessous).

```
[
    {'ville': 'Marseille', 'plusJeuneAge': 23},
    {'ville': 'Lyon', 'plusJeuneAge': 23},
    {'ville': 'Paris', 'plusJeuneAge': 25},
    {'ville': 'Nantes', 'plusJeuneAge': 23},
    {'ville': 'Lille', 'plusJeuneAge': 23},
    {'ville': 'Nice', 'plusJeuneAge': 23},
    {'ville': 'Montpellier', 'plusJeuneAge': 23}
]
```