R1.01 Initiation au développement
Feuille de TP n°13
Utilisation avancée des fonctions min, max et sorted

Objectifs de la feuille

- Terminer le jeu du labyrinthe
- Manipuler les fonctions min, max et sorted avec key

Exercice 1 *Le jeu du labyrinthe*

Termine le jeu du labyrinthe (TP12)

Exercice 2 *Petites bêtes !*

Dans cet exercice, il faut compléter le fichier `pokemon.py`

Un pokedex est modélisé par une liste contenant des informations sur des pokemons. Ces informations sont données sous la forme d'un tuple (nom, familles, attaque, défense, poids).

Par exemple :

```
ma_liste = [
  ('Bulbizarre', {'Plante', 'Poison'}, 4, 3, 7),
  ('Herbizarre', {'Plante', 'Poison'}, 5, 5, 13),
  ('Jungko', {'Plante'}, 7, 1, 52),
  ('Abo', {'Poison'}, 4, 2, 6)]
```

Ainsi, `('Bulbizarre', {'Plante', 'Poison'}, 4, 3, 7)` modélise un pokemon qui se nomme Bulbizarre. Il appartient aux familles Plante et Poison. Sa valeur d'attaque est égale à 4, sa valeur de défense est égale à 3 et son poids est de 7 kg.

2.1 Écrire une fonction `plus_forte_attaque` qui prend en paramètre une telle liste de pokémons et qui renvoie toutes les informations dur le pokemon qui a la plus forte valeur d'attaque. Par exemple
`plus_forte_attaque(ma_liste)` doit renvoyer `('Jungko', 'Plante', 7, 1, 52)`

2.2 Écrire une fonction `tri_selon_defense` qui prend en paramètre une telle liste de pokémons et qui renvoie la liste des noms des pokemons triées selon l'ordre croissant de leur défense. Par exemple `tri_selon_defense(ma_liste)` doit renvoyer

`['Jungko', 'Abo', 'Bulbizarre', 'Herbizarre']`

2.3 La "force" d'un pokemon est la somme de son attaque et de sa défense. Écrire une fonction `plus_petite_force` qui prend en paramètre une telle liste de pokémons et qui renvoie le nom du pokemon qui a la plus petite force. Par exemple `plus_petite_force(ma_liste_pokemon)` doit renvoyer `'Abo'`

2.4 Écrire une fonction `tri_selon_diversite` qui prend en paramètre une telle liste de pokémons et qui renvoie la liste des pokemons triées selon leur nombre croissant de type d'attaque. Deux pokemons qui on le même nombre de type d'attaque seront triées dans l'ordre croissant de leur valeur d'attaque. Par exemple `tri_selon_diversite(ma_liste)` doit renvoyer

```
[('Abo', {'Poison'}, 4, 2, 6),
 ('Jungko', {'Plante'}, 7, 1, 52),
 ('Bulbizarre', {'Plante', 'Poison'}, 4, 3, 7),
 ('Herbizarre', {'Plante', 'Poison'}, 5, 5, 13),]
```

Exercice 3 *Petites bêtes ! (version 2)*

Dans cet exercice, il faut compléter le fichier `pokemon_v2.py` Pour les tests, on utilisera le fichier `test_pokemon` en remplaçant la première ligne par :

```
import pokemon_v2 as pokemon
```

Je décide de modifier la modélisation d'un pokedex en utilisant un dictionnaire dont les clefs sont les noms des pokemons et les valeurs associée des informations sur des pokemons. Ces informations sont données sous la forme d'un tuple (familles, attaque, défense, poids). Par exemple :

```
mon_dico = {
  'Bulbizarre': ({'Plante', 'Poison'}, 4, 3, 7),
  'Jungko': ({'Plante'}, 7, 1, 52),
  'Herbizarre': ({'Plante', 'Poison'}, 5, 5, 13),
  'Abo': ({'Poison'}, 4, 2, 6)}
```

3.1 On donne le code de la fonction `la_plus_forte_attaque` avec cette nouvelle modélisation. Pourquoi est-il nécessaire ici d'écrire la fonction `critere` à l'intérieur du code de la fonction `la_plus_forte_attaque` ?

```
def la_plus_forte_attaque(pokedex):
    def critere(nom):
        return pokedex[nom][1]
    le_plus_fort = max(pokedex, key = critere)
    (familles, attaque, defense, poids) = pokedex[le_plus_fort]
    return (le_plus_fort, familles, attaque, defense, poids)
```

3.2 Compléter le code des autres fonctions.

Remarque : comme d'habitude, vous avez le droit d'écrire et utiliser des fonctions auxiliaires pour faciliter l'écriture ou la lecture de votre code.

Exercice 4 *Qui a mon livre ?*

4.1 Implémente les fonctions de l'exercice 6 (Qui a mon livre ?) du TD 12

4.2 Implémente les fonctions de l'exercice 7 (Club sportif) du TD 12

N'oublie pas d'écrire des tests !

Exercice 5 *Petites bêtes ! (version 3)*

Dans cet exercice, il faut compléter le fichier `pokemon_v3.py` Pour les tests, on utilisera le fichier `test_pokemon` en remplaçant la première ligne par :

```
import pokemon_v3 as pokemon
```

5.1 Compléter le code des quatre fonctions proposées.