Autotest n°4 – correction

Exercice 1 : Faire les exercices d'entraînement du site d'activités (thème 7).

Exercice 2 : *Principaux thèmes abordés : structure de données (tableaux, dictionnaires) et langages et programmation (spécification).*

Objectif de l'exercice :

Les Aventuriers du Rail© est un jeu de société dans lequel les joueurs doivent construire des lignes de chemin de fer entre différentes villes d'un pays.

La carte des liaisons possibles dans la région Occitanie est donnée à la fin de l'exercice. Cette carte contient également les liaisons possédées par le joueur 1 (en noir), et celles du joueur 2 (en blanc). Les liaisons en gris sont encore en jeu.

Codages des structures de données utilisées :

- **Liste des liaisons d'un joueur :** Toutes les liaisons directes (sans ville intermédiaire) construites par un joueur seront enregistrées dans une variable de type "**tableau de tableaux**".
  Le joueur 1 possède les lignes directes "Toulouse-Muret", "Toulouse-Montauban", "Gaillac-St Sulpice" et "Muret-Pamiers" (liaisons indiquées en noir dans l'annexe). Ces liaisons sont mémorisées dans la variable ci-contre.

```
liaisonsJoueur1 = [
  ["Toulouse","Muret"],
  ["Toulouse","Montauban"],
  ["Gaillac","St Sulpice"],
  ["Muret","Pamiers"]
]
```

  Remarque : Seules les liaisons directes existent, par exemple `["Toulouse","Muret"]` ou `["Muret","Toulouse"]`. Par contre, le tableau `["Toulouse","Mazamet"]` n'existe pas, puisque la ligne Toulouse-Mazamet passe par Castres.
- **Dictionnaire associé à un joueur :** On code la liste des villes et des trajets possédée par un joueur en utilisant un **dictionnaire de tableaux**. Chaque **clef de ce dictionnaire** est une ville de départ, et chaque **valeur** est un **tableau** contenant les villes d'arrivée possibles en fonction des liaisons possédées par le joueur.
  Le **dictionnaire de tableaux** du joueur 1 est donné ci-contre :

```
DictJoueur1 = {
  "Toulouse":["Muret","Montauban"],
  "Montauban":["Toulouse"],
  "Gaillac":["St Sulpice"],
  "St Sulpice":["Gaillac"],
  "Muret":["Toulouse","Pamiers"],
  "Pamiers":["Muret"]}
```

1) Expliquer pourquoi la liste des liaisons suivante n'est pas valide :

```
tableauliaisons = [["Toulouse","Auch"], ["Luchon","Muret"],
                   ["Quillan","Limoux"]]
```

   Solution : La liaison Luchon-Muret n'est pas valide, puisque les deux villes ne sont pas directement reliées.

2) Cette question concerne le joueur n°2 (Rappel : les liaisons possédées par le joueur n°2 sont représentées par un rectangle blanc dans l'annexe).
   a) Donner le tableau `liaisonsJoueur2`, des liaisons possédées par le joueur n°2.

```
liaisonsJoueur2 = [
    ["Tarbes", "St Gaudens"],
    ["Mazamet", "Castres"],
    ["Castres", "Toulouse"],
    ["Toulouse", "Castelnaudary"],
    ["Castelnaudary", "Carcassone"]
]
```

b) Recopier et compléter le dictionnaire suivant, associé au joueur n°2 :

```
DictJoueur2 = {
    "Toulouse": ["Castres","Castelnaudary"],
    ...
}
```

```
DictJoueur2 = {
    "Toulouse": ["Castres","Castelnaudary"],
    "Castres": ["Mazamet", "Toulouse"],
    "Mazamet": ["Castres"],
    "Castelnaudary": ["Toulouse", "Carcassonne"],
    "Carcassonne": ["Castelnaudary"],
    "Tarbes": ["St Gaudens"],
    "St Gaudens": ["Tarbes"]
}
```

3) À partir du tableau de tableaux contenant les liaisons d'un joueur, on souhaite construire le dictionnaire correspondant au joueur. Une première proposition a abouti à la fonction `construireDict` ci-dessous.

```
1  def construireDict(listeLiaisons):
2      """
3      listeLiaisons est un tableau de tableaux représentant la
4      liste des liaisons d'un joueur comme décrit dans le problème
5      """
6      Dict={}
7      for liaison in listeLiaisons:
8          villeA = liaison[0]
9          villeB = liaison[1]
10         if not villeA in Dict.keys():
11             Dict[villeA]=[villeB]
12         else :
13             destinationsA = Dict[villeA]
14             if not villeB in destinationsA:
15                 destinationsA.append(villeB)
16     return Dict
```

a) Écrire sur votre copie un **assert** dans la fonction `construireDict` qui permet de vérifier que la `listeLiaisons` n'est pas vide.
Solution : `assert len(listeLiaisons) > 0`

b) Sur votre copie, donner le résultat de cette fonction ayant comme argument la variable `liaisonsJoueur1` donnée dans l'énoncé et expliquer en quoi cette fonction ne répond que partiellement à la demande.
Solution : On obtient :

```
{
    "Toulouse":["Muret","Montauban"],
    "Montauban":["Toulouse"],
    "Gaillac":["St Sulpice"],
    "Muret":["Pamiers"],
}
```

c) La fonction `construireDict`, définie ci-dessus, est donc partiellement inexacte. Compléter la fonction `construireDict` pour qu'elle génère bien l'ensemble du dictionnaire de tableaux correspondant à la liste de liaisons données en argument. À l'aide des numéros de lignes, on précisera où est inséré ce code.

Solution : Il manque les liaisons allant de `villeB` à `villeA`. Il faut donc rajouter les lignes suivantes avant la ligne 16.

```
        if not villeB in Dict.keys():
            Dict[villeB]=[villeA]
        else :
            destinationsB = Dict[villeB]
            if not villeA in destinationsB:
                destinationsB.append(villeA)
```

Liaisons possédées par le joueur 1 (en noir) et le joueur 2 (en blanc)