# Devoir maison N°4

## Exercice 1

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

### Présentation

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 en annexe 1. Dans l'annexe 2 les liaisons possédées par le joueur 1 sont 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-StSulpice" et "Muret-Pamiers" (liaisons indiquées en noir dans l'annexe 2 de l'exercice 2). 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"]}
```

**Q1. Expliquer** pourquoi la liste des liaisons suivante n'est pas valide :

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

Pour rappel, les liaisons possédées par le joueur n°2 sont représentées par un rectangle blanc dans l'annexe 2.

**Q2. Donner** le tableau liaisonsJoueur2, des liaisons possédées par le joueur n°2.

**Q3. Recopier** et **compléter** le dictionnaire suivant, associé au joueur n°2 :

```
dictJoueur2 = {"Toulouse":["Castres","Castelnaudary"],
                        ...
                       }
```

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

**Q4.** Sur votre copie, **donner** le résultat de cette fonction ayant comme argument la variable liaisonsJoueur1 définie dans l'énoncé et **expliquer** en quoi cette fonction ne répond que partiellement à la demande.

La fonction `construireDict`, définie ci-dessus, est donc partiellement inexacte.

**Q5. 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.

## Exercice 2

Cet exercice est à réaliser sur ordinateur. Le fichier réponse Python devra être déposé sur le dépôt Moodle associé à ce devoir maison.

**Q1. Écrire** et **tester** sur ordinateur une fonction `rechercheMinMax` qui prend en paramètre un tableau de nombres `tab` non triés et non vide. Cette fonction renvoie la plus petite et la plus grande valeur du tableau sous la forme d'un dictionnaire à deux clés 'min' et 'max'. Les tableaux seront représentés sous forme de liste Python.

Exemples :

```
>>> tableau = [0, 1, 4, 2, -2, 9, 3, 1, 7, 1]
>>> rechercheMinMax(tableau)
{'min': -2, 'max': 9}
>>> tableau = [2]
>>> rechercheMinMax(tableau)
{'min': 2, 'max': 2}
```

## Annexe 1 : Extrait des liaisons possibles en Occitanie


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