1. Trier des données

Nous reprenons notre fichier de joueurs de rugby du Top14.

```
import csv
f = open('data/Top14.csv', "r", encoding = 'utf-8')
donnees = csv.DictReader(f)
joueurs = []
for ligne in donnees:
    joueurs.append(dict(ligne))

f.close()
```

1.0.1 Créer une fonction filtre

L'objectif est de créer une fonction `joueursEquipe(equipe)` qui renvoie une liste contentant tous les joueurs de l'équipe `equipe`. Le paramètre `equipe` sera donnée sous forme de chaîne de caractères. La valeur renvoyée sera de type liste.

```
def joueursEquipe(equipe):
    ret = []

    for k in joueurs :
        if k['Equipe'] == equipe :
            ret.append(k)

    return ret

joueursEquipe("Clermont")
```

Exemple d'utilisation :

```
len(joueursEquipe("Clermont"))
```

```
43
```

Définir de la même manière une fonction `joueursPoste(poste)`.

```

```

1.1 Utilisation d'une fonction de tri

Comment classer les joueurs suivant leur taille ? La fonction `sorted(liste)` est efficace sur les listes : elle renvoie une nouvelle liste triée dans l'ordre croissant.

```
mylist = [4,2,8,6]
mynewlist = sorted(mylist)
print(mynewlist)
```

```
[2, 4, 6, 8]
```

Mais comment trier un dictionnaire ?

```
test = sorted(joueurs)
```

Il est normal que cette tentative échoue : un dictionnaire possède plusieurs clés différentes. Ici, plusieurs clés peuvent être des critères de tri : la taille, le poids.

1.1.1 un exemple de tri de dictionnaire

```
Simpsons = [{"Prenom" : "Bart", "age estimé": "10"},
            {"Prenom" : "Lisa", "age estimé": "8"},
            {"Prenom" : "Maggie", "age estimé": "1"},
            {"Prenom" : "Homer", "age estimé": "38"},
            {"Prenom" : "Marge", "age estimé": "37"}]
```

```
def age(personnage):
    return int(personnage["age estimé"])
```

```
age(Simpsons[0])
```

```
10
```

La création de cette fonction `age()` va nous permettre de spécifier une clé de tri, par le paramètre `key` :

```
triSimpsons = sorted(Simpsons, key = age)
```

```
triSimpsons
```

```
triSimpsons = sorted(Simpsons, key = age, reverse = True)
```

```
triSimpsons
```

1.2 Exercice

1. Trier les joueurs du top14 par taille.
2. Trier les joueurs de Clermont par taille.
3. Trier les joueurs de Clermont suivant leur Indice de Masse Corporelle (IMC)

```
def taillePlayer(player) :
    return int(player['Taille'])
```

```
joueurs_taille_croissant = sorted(joueurs, key = taillePlayer)
print(joueurs_taille_croissant)
```

```
def IMC(player):
    masse = int(player['Poids'])
    taille_m = int(player['Taille']) / 100
    return masse / taille_m**2
```

```
IMC(joueurs[0])
```

```
36.42987249544626
```

```
joueursASM = [k for k in joueurs if k['Equipe'] == 'Clermont']
```

```
joueursASM_tri = sorted(joueursUBB, key = IMC)
```

```
for k in joueursASM_tri:
    print(k['Nom'], IMC(k))
```

1.3 Recherche des joueurs de profil physique similaire

1.3.1 Distance entre deux joueurs

Construire une fonction `distance(joueur1,joueur2)` qui renvoie la somme des carrés des différences de tailles et de poids entre les joueurs `joueur1` et `joueur2` : $$ d = (p_1-p_2)^2 + (t_1-t_2)^2$$

```
def distance(joueur1,joueur2):
    p1 = int(joueur1['Poids'])
    p2 = int(joueur2['Poids'])
    t1 = int(joueur1['Taille'])
    t2 = int(joueur2['Taille'])
    return (p1-p2)**2+(t1-t2)**2
```

1.3.2 Distance des joueurs avec Baptiste Serin

Retrouvons d'abord le numéro de Baptiste Serin dans notre classement de joueurs :

```
for k in range(len(joueurs)) :
    if joueurs[k]['Nom'] == 'Baptiste SERIN' :
        print(k)
```

```
530
```

```
joueurs[530]
```

Nous pouvons maintenant classer les joueurs suivant leur distance morphologique à Baptiste SERIN :

```
def distanceSerin(joueur2):
    return distance(joueurs[530],joueur2)
```

```
distanceSerin(joueurs[530])
```

```
0
```

```
joueurs_VS_Serin = sorted(joueurs, key = distanceSerin)
```

```
joueurs_VS_Serin
```

```

```