GARNERO Camille

## Répartition des entraineurs

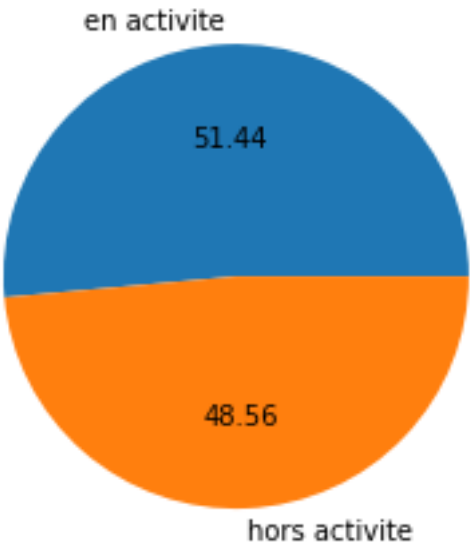
On souhaite savoir combien d'entraineurs sont encore en activité et combien ne le sont pas.

On regarde dans le dataframe si les entraineurs ont un currendTeamID ou non.

```
df = pd.read_json("coaches.json")
```

```
en_activite = len(df.loc[df['currentTeamId']==0])
hors_activite = len(df.loc[df['currentTeamId']!=0])
```

**Résultats :**




(en %)

## Répartition par pays

On souhaite savoir combien d'entraineur il y a dans chaque pays.

On regarde le nombre d'occurrence de chaque pays dans le dataframe

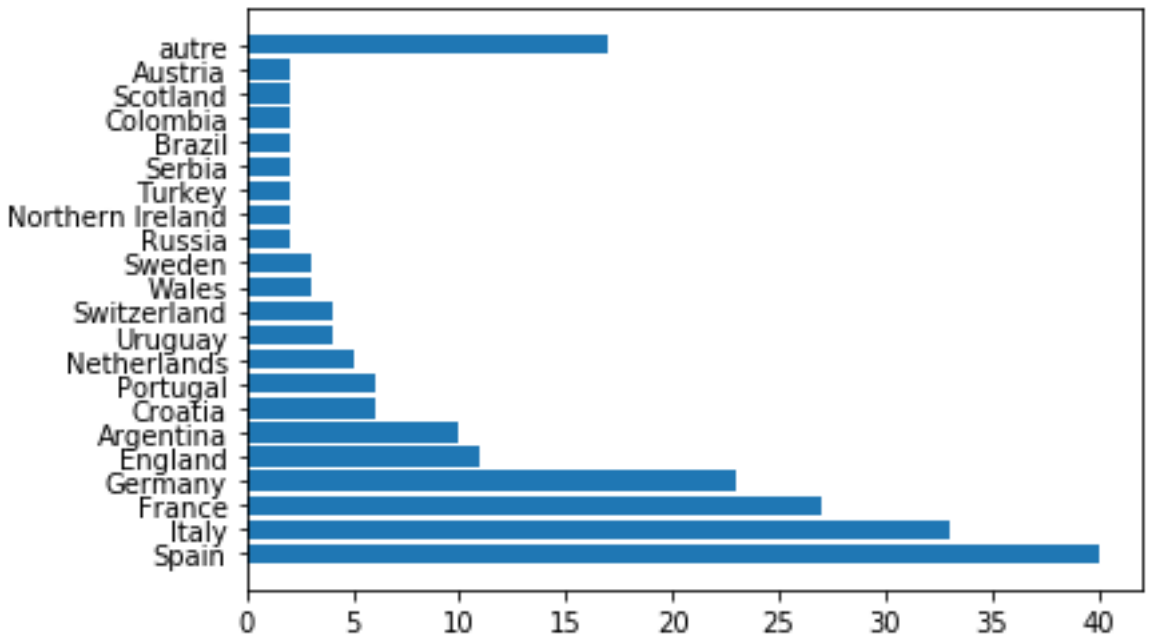
```
df = pd.read_json("coaches.json")
```

```
df_countries = df['passportArea'].value_counts()
```

On peut maintenant facilement voir combien d'entraineurs il y a dans chaque pays.

Pour des questions de lisibilité, les pays n'ayant qu'un seul entraineur seront rassemblés sous la dénomination 'autre '.

**Résultats :**



## Carte des équipes

On cherche à placer les équipes du fichier teams.json sur un carte.

```
df_2 = pd.read_json("teams.json")
```

• On peut facilement associer chaque équipe à une ville de la manière suivante :

```
for city, club in zip(df_2['city'], df_2['name']):
```

• Pour obtenir les coordonnées, on utilise le module Nominatim

```
geolocator = Nominatim(user_agent="school_project")
```

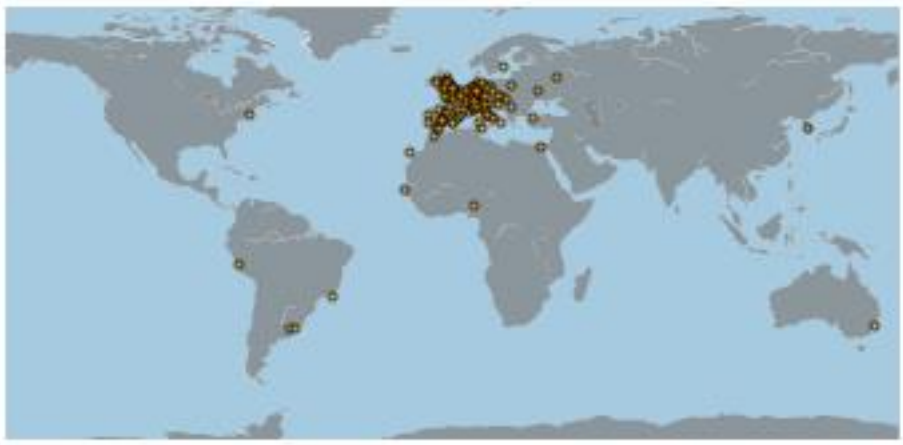
Pour chaque ville :

```
location = geolocator.geocode(str(city))
latitudes.append(location.latitude)
longitudes.append(location.longitude)
```

• Comme le dataframe comporte des erreurs d'encodage, certaines villes n'ont pas pu être trouvée :

```
M\u00f6nchengladbach n'a pas pu etre trouvee
Legan\u00e9s n'a pas pu etre trouvee
M\u00fcnchen n'a pas pu etre trouvee
Saint-\u00c9tienne n'a pas pu etre trouvee
K\u00f6ln n'a pas pu etre trouvee
A Coru\u00f1a n'a pas pu etre trouvee
Donostia-San Sebasti\u00e1n n'a pas pu etre trouvee
Tiran\u00eb (Tirana) n'a pas pu etre trouvee
Bucure\u015fti n'a pas pu etre trouvee
Bogot\u00e1 D.C. n'a pas pu etre trouvee
Ciudad de Panam\u00e1 n'a pas pu etre trouvee
Br\u00f8ndby n'a pas pu etre trouvee
San Jos\u00e9 n'a pas pu etre trouvee
T\u014dky\u014d (Tokyo) n'a pas pu etre trouvee
Tehr\u0101n (Teheran) n'a pas pu etre trouvee
Reykjav\u00edk n'a pas pu etre trouvee
Coyoac\u00e1n, Ciudad de M\u00e9xico (D.F.) n'a pas pu etre trouvee
Ar-Riy\u0101\u1e0d (Riyadh) n'a pas pu etre trouvee
```

**Résultats :**



## Evolution des scores

On veut pouvoir observer l'évolution des scores des 6 équipes suivantes : Liverpool, Manchester City, Manchester United, Chelsea, Arsenal, Tottenham.

Pour cela on utilise les dataframes teams.json et matches_England.json.

```
df_2 = pd.read_json("teams.json")
df_3 = pd.read_json("matches_England.json")
```

On créé une fonction qui trace le graphe à partir uniquement du nom du club

```
def show_evolution(club_name):
```

• La première étape est de récupérer l'identifiant de la ville grâce au dataframe teams.json

```
identifiant_club = df_2.loc[df_2['name'] == club_name, ['wyId']]
identifiant_club = identifiant_club.iat[0,0]
```

• On veut ensuite filtrer le dataframe matches_England.json afin de n'avoir que les lignes dans lesquelles notre équipe joue.

```
def filter_id(row):
    return str(identifiant_club) in row['teamsData'].keys()

filtre_bool = df_3.apply(filter_id, axis=1)
df_club_joue = df_3[filtre_bool]
```

• A partir de ce moment, on peut aisément récupérer les scores du match ainsi que sa date. Attention à bien convertir la date (initialement au format str) en datetime, afin de pouvoir trier par date.

Conversion :

```
try :
    date_object = datetime.datetime.strptime(date, '%B %d, %Y at %H:%M:%S PM GMT+2')
except ValueError:
    date_object = datetime.datetime.strptime(date, '%B %d, %Y at %H:%M:%S PM GMT+1')
```

Tri - matchs est un liste comprenant les couples (date, score) - :

```
sorted_maths = np.array(sorted(matchs, key=itemgetter(0)))
```

• Maintenant que la liste est triée, on peut l'afficher. On n'affiche pas les dates sur l'axe des abscisses car l'accumulation de dates rendraient le graphique illisible.

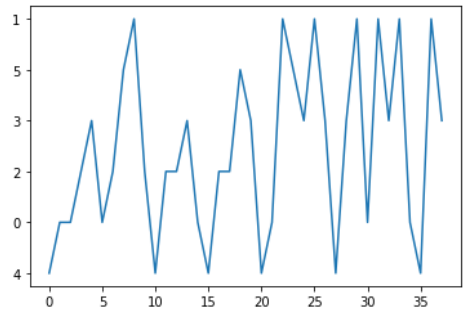
```
plt.plot(np.array(matchs)[:,1])
plt.show()
```

On peut utiliser la fonction sur les 6 villes concernées :

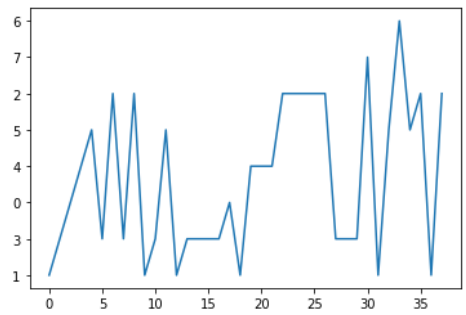
```
show_evolution('Liverpool')
show_evolution('Manchester City')
show_evolution('Manchester United')
show_evolution('Chelsea')
show_evolution('Arsenal')
show_evolution('Tottenham Hotspur')
```

**Résultats :**

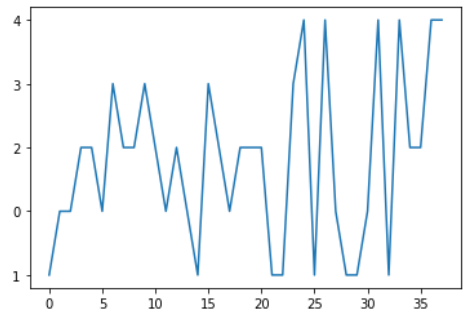
**Liverpool**



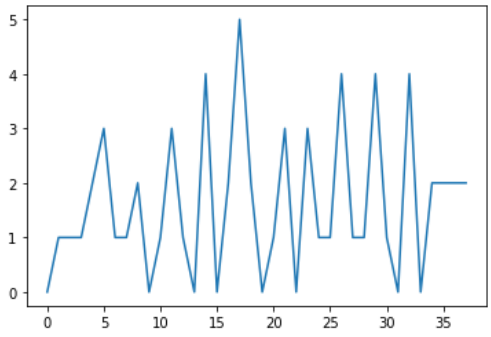

**Manchester City**



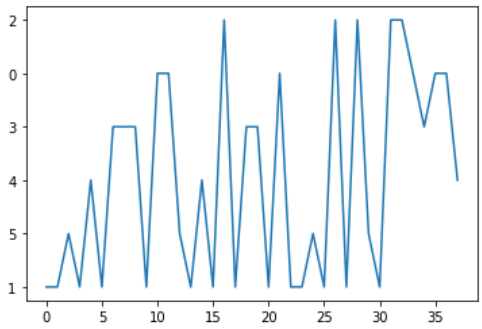

**Manchester United**



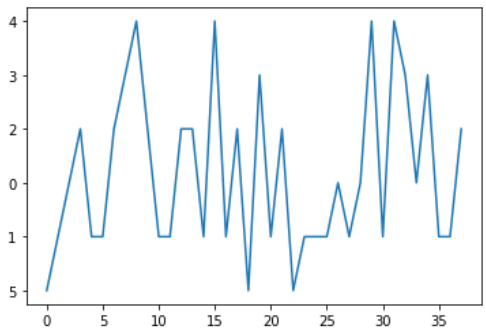
**Chelsea**




**Arsenal**




**Tottenham**



## Placement des buteurs

On veut afficher sur un stade de foot la position du tir marquant pour chaque but d'une équipe.

On dispose du fichier event_England.json

```
df_2 = pd.read_json("teams.json")
df_4 = pd.read_json("events_England.json")
```

Pour cela on utilise une fonction qui prend comme argument le nom du club

```
def show_buts(club_name):
```

• On récupère l'identifiant du club comme on l'a fait précédemment.

• On filtre le dataframe events_England.json en fonction
- de l'identifiant du club
- de si il y a but

```
df_goals = df_4.loc[(df_4['teamId'] == identifiant_club) & (df_4['subEventName'] == 'Goal kick')]
```

• Maintenant qu'on a que les lignes qui nous interessent, on récupère les coordonnées des tirs.

```
x, y = list(), list()
for i in df_goals['positions']:
    x.append(i[1]['x'])
    y.append(i[1]['y'])
```

On peut désormais tracer le résultat.

**Résultats :**

```
show_buts('Chelsea')
```