# INFORMATIQUE – DS n°2 – Correction

## 1 Equipe de volley

**1)** Fonction renvoyant la liste des joueurs d'une nationalité donnée

```
def joueurs_nationalite(pays):
    joueurs = []
    for joueur, infos in equipe.items():
        if infos[" Nationalite "] == pays:
            joueurs.append(joueur)
    return joueurs

print(joueurs_nationalite(" France "))
```

**2)** Fonction renvoyant la liste des nationalités présente dans l'équipe

```
def nationalite():
    liste_nat = []
     for infos in equipe.values():
        if (infos[" Nationalite "] in liste_nat) == False:
    liste_nat.append(infos[" Nationalite "])
    return liste_nat

print(nationalite())
```

**3)** Fonction renvoyant la taille moyenne des joueurs de l'équipe

```
def taille_moy():
    total_taille = 0
    nb_joueurs = len(equipe)
    for infos in equipe.values():
        total_taille += infos[" Taille "]
    if nb_joueurs > 0:
        return total_taille / nb_joueurs
    else:
        return 0

print(taille_moy())
```

## 2 Recherche des plus grands éléments d'une liste

**4)** Fonction renvoyant la position du maximum `k` et la valeur du maximum `maxi`. Il suffit de créer une variable `k` qui stocke la position du maximum. Ici, elle s'initialise à $1$ pour tenir compte du fait que les listes démarrent par la position $0$.

```
def maximum_pos(L):
    n = len(L)
    maxi = L[0]
    k=1
    for i in range(1,n):
        if L[i]> maxi:
            maxi = L[i]
            k=i+1
    return (k,maxi)

L=[1,6,3,1,6,2,1,3,3,3,4,2,3,1,3]
print(maximum_pos(L))
```

**5)** Si on utilise une inégalité large, le programme renverra la position de la dernière apparition du maximum alors qu'une inégalité stricte renvoie la position de la première occurrence du maximum.

**6)** Fonction renvoyant les deux premiers maximums et leurs positions. On initialise les deux premières valeurs des maximums et de leurs positions avec les deux premières valeurs de la liste. Si la deuxième valeur est plus grande que la première, on inverse. Puis on fait une boucle pour parcourir le tableau. Si un élément est plus grand que le premier maximum, cet élément devient premier maximum et l'ancien premier maximum devient deuxième maximum. Si un élément est compris entre le premier et le deuxième maximum, il devient deuxième maximum. On change à chaque fois les valeurs des positions en conséquence.

```
def premier_second_maximum(L):
    m1,m2,p1,p2 = L[0],L[1],0,1
    if m2 > m1 :
        m1,m2,p1,p2 = L[1],L[0],1,0
    for i in range (2,len(L)):
        e=L[i]
        if e>m1:
            m1,m2,p1,p2 = e,m1,i,p1
        elif e>m2:
            m2,p2 = e,i
    return (m1,p1),(m2 ,p2)

L=[1,6,3,1,6,2,1,3,9,3,4,2,3,1,3]
print(premier_second_maximum(L))
```

**7)** On trouve les résultats suivants pour les différentes listes proposées : `((9,3),(6,1))` ; `((3,0),(3,2))` ; `((3,1),(3,3))` ; `((6,2),(5,1))` ; `((6,0),(5,3))` ; `((7,1),(4,0))`

# 3 Tracé d'une courbe à partir de données expérimentales

**8)** La commande `open` avec l'option `"r"` permet d'ouvrir un fichier en mode lecture.

```
f=open("Mesure_axe_Emericc.txt","r")
```

**9)** La commande `readlines()` permet de convertir le fichier .txt en une liste où chaque élément correspond à une ligne.

```
ligne=f.readlines()
print(ligne)
```

**10)** La commande `rstrip(x)` appliquée à une chaîne de caractères, donc à chaque élément de la liste des lignes du fichier .txt, permet de retirer le caractère `x` à la fin de la chaîne.

```
for element in ligne:
    a=element.rstrip("\n")
    print(a)
```

Cela renvoie toutes les lignes sans le retour à la ligne `"\n"`.

**11)** Pour découper une chaîne de caractères selon un caractère donné (ici la tabulation `"\t"`, on utilise la commande `split`.

```
for element in ligne:
    a=element.rstrip("\n")
    b=x.split("\t")
    print(b)
```

On obtient chaque ligne du fichier .txt sans le retour à la ligne et, s'il y avait des tabulations sur une ligne, on obtient une liste pour cette ligne : cela correspond aux valeurs de positions dans notre cas.

**12)** Pour supprimer les lignes correspondant aux explications sur les paramètres, on peut utiliser la commande `pop(i)` qui retire la ligne `i` d'une liste ou bien la dernière ligne si on ne met pas d'argument. Si on repart de la question précédente, il faut remettre les valeurs `b` dans une liste `donnees` avec la méthode `append`.

```
donnees=[]
for element in ligne:
    a=element.rstrip("\n")
    b=x.split("\t")
    print(b)
    donnees.append(b)

for i in range(0,12):
    donnees.pop(0)
for i in range(0,9):
    donnees.pop()
```

**13)** À ce stade, nous avons une liste `donnees` où chaque élément est une liste de trois chaînes de caractères représentant chacune un nombre : le numéro de la mesure, la position selon $x$ et la position selon $y$. On peut commencer par extraire les deux dernières colonnes nous intéressant. Puis on fait une boucle qui va chercher la valeur de $x$ pour la mettre dans une nouvelle liste et de même pour $y$. On pense aussi à convertir la chaîne de caractères en un nombre de type `float`.

```
donnees=donnees[1:3]

x,y=[],[]
for in range(len(donnees)):
    xi=float(donnees[i][0])
    x.append(xi)
    yi=float(donnees[i][1])
    y.append(yi)
```

**14)** Cette question a été traitée juste au-dessus.

**15)** Il ne reste plus qu'à tracer la courbe avec le module `pyplot` de la bibliothèque `matplotlib`.

```
from matplotlib import pyplot as plt

plt.plot(x,y)
plt.show()
```

Ainsi, si on regroupe toutes les questions, on obtient le code suivant :

```
from matplotlib import pyplot as plt

f=open("Mesure_axe_Emericc.txt","r")
ligne=f.readlines()

donnees=[]
for element in ligne:
    a=element.rstrip("\n")
    b=x.split("\t")
    donnees.append(b)

for i in range(0,12):
    donnees.pop(0)
for i in range(0,9):
    donnees.pop()

donnees=donnees[1:3]

x,y=[],[]
for in range(len(donnees)):
    xi=float(donnees[i][0])
    x.append(xi)
    yi=float(donnees[i][1])
    y.append(yi)

plt.plot(x,y)
plt.show()
```