? À rendre mercredi 9 octobre 2024

Devoir Maison n°2 : Autour des dictionnaires – *Corrigé*

I Rappels de 1re année

Reprendre vos cours/TD de MPSI/PCSI sur les dictionnaires.
Et voir le polycopié de cours « Chapitre n°2. Les dictionnaires » partie I.

II Exercices

Exercice n°1

On suppose disposer d'un dictionnaire qui contient le nombre de médailles obtenues chaque jour par les français lors des JO de Paris 2024. Pour chaque élément, la clé est la date, et la valeur le nombre de médailles obtenu ce jour-là.

```
JO2024={'27/07/2024':4,'28/07/2024':4,...,'11/08/2024':2}
```

R1. Écrire une fonction `moyenne(d:dict)->float` qui prend en argument un dictionnaire `d` du type de celui défini ci-dessus et qui renvoie la valeur moyenne des valeurs.

Solution:

```
def moyenne(d:dict)->float:
    """
    renvoie la moyenne des valeurs du dictionnaire
    """
    S=0
    for c in d: # parcours des clés de d
        S=S+d[c] # on ajoute la température de la clé c
    return S/len(d)
```

R2. Écrire la ligne pour obtenir le nombre moyen journalier de médailles.

R3. Écrire la fonction `maxd(d:dict)->float` qui prend en argument un dictionnaire du type du précédent, et qui renvoie la valeur maximale (c'est-à-dire dans le cadre de l'exemple le nombre maximal de médailles obtenu un même jour).

On pourra utiliser `math.inf` de la bibliothèque `math` qui donne « l'infini » .

Solution:

```
def maxd(d:dict)->float:
    """
    renvoi la valeur maximale du dictionnaire d
    """
    m=-float('inf') # initialisation de la valeur maximale
    for c in d: # parcours du dictionnaire
        if d[c]>m: # valeur de clé c supérieure au max local
            m=d[c] # nouveau maximum
    return m
```

R4. Écrire la fonction `cle_max(d:dict)->str` qui prend en argument un dictionnaire du type du précédent, et qui renvoie la clé de valeur maximale (c'est-à-dire dans le cadre de l'exemple la date à laquelle le nombre maximal de médailles a été obtenu un même jour).

Solution:

```
def cle_max(d:dict)->'str':
    """
    renvoi la clé de valeur minimale du dictionnaire d
    """
    m=-float('inf') # valeur minimale
    for c in d: # parcours du dictionnaire
        if d[c]>m: # nouveau max
            m=d[c]
            cmax=c # mise à jour de la clé
    return cmax
```

On dispose maintenant d'un dictionnaire dont les clés sont les jours et les valeurs une liste de trois entiers qui sont le nombre de médailles d'or, d'argent et de bronze du jour :

```
JO2024={'27/07/2024':[1,2,1],'28/07/2024':[2,1,1],'29/07/2024':[2,5,1],
   ...,'11/08/2024':[0,2,0]}
```

Par exemple, le 29 juillet, 2 médailles d'or ont été remportées, 5 en argent et 1 en bronze.

R5. Écrire un script (on n'attend pas une fonction) qui permet d'obtenir une liste de trois éléments donnant le nombre total de chaque type de médailles.

Solution:

```
medailles=[0,0,0]
for c in JO2024:
    for i in range(3):
        medailles[i]=medailles[i]+JO2024[c][i]
```

R6. Écrire un script (on n'attend pas une fonction) qui permet d'obtenir la liste des jours où au moins une médaille d'or a été remportée.

Solution:

```
jour_or=[]
for c in JO2024:
    if JO2024[c][0]!=0:
        jour_or.append(c)
```

Exercice n°2 Occurrences des voyelles dans une chaîne de caractères

Écrire une fonction `analyseTexte(s:str)->dict` qui prend en argument une chaine de caractère `s` et renvoie un dictionnaire qui contient le nombre de fois où chaque caractère apparaît.

Par exemple, `analyseTexte('abracadabra')` renverra `{'a':5, 'b':2, 'r':2, 'c':1, 'd':1}`.

Elle devra avoir une complexité :

— linéaire en la longueur n de `s`,

— indépendante de k, le nombre de caractères distincts présents dans `s`.

De plus, cette fonction devra impérativement ne parcourir qu'une seule fois la chaîne de caractères. On admettra qu'un test d'appartenance d'une clé à un dictionnaire se fait à coût constant.

Solution:

```
def analyseTexte(s):
    """
    renvoi le dictionnaire des occurrences des caractères de de la chaine
   texte
    """
    occ={}
    for x in s:
        if x not in occ:
            occ[x]=1
        else: # x déjà dans occ
            occ[x]=occ[x]+1
    return occ
```