Python-ECG1-11-Listes3-Copy1

November 30, 2021

1 TP11 - Parcourir une liste – Corrigé

1.1 Exercice 1 - Variance

1) Écrire une fonction **moyenne(L)** qui, étant donnée une liste de nombres *L*, renvoie la moyenne de ses éléments.

```
In [1]: # version 1
        def moyenne(L):
            m = 0
            for k in range(len(L)):
                m = m + L[k]
            m = m/len(L)
            return m

        # Version 2
        def moyenne(L):
            m = 0
            for x in L:
                m = m + x
            m = m/len(L)
            return m
```

Tester avec la liste L = [3,5,1,3]

```
In [2]: L = [3,5,1,3]
        moyenne(L)
```

```
Out[2]: 3.0
```

2) Écrire une fonction **variance(L)** qui, étant donnée une liste de nombres L, renvoie la variance de ses éléments.

```
In [3]: # Version 1
        def variance(L):
            m = moyenne(L)
            v = 0
            for k in range(len(L)):
```

```
            v = v + (L[k]-m)**2
        v = v/len(L)
        return v

    # Version 2
    def variance(L):
        m = moyenne(L)
        v = 0
        for x in L:
            v = v + (x-m)**2
        v = v/len(L)
        return v
```

Tester avec la liste précédente.

```
In [4]: variance(L)
```

```
Out[4]: 2.0
```

1.2 Exercice 2 - maximum - minimum

1) Sur feuille.

2) Écrire une fonction **maximum(L)** qui renvoie la maximum d'une liste de nombres L.

```
In [5]: # Version 1
        def maximum(L):
            M = L[0]
            for k in range(1,len(L)) :
                if L[k] > M:
                    M = L[k]
            return M

        # Version 2
        def maximum(L):
            M = L[0]
            for x in L :
                if x > M:
                    M = x
            return M
```

Tester avec la liste L = [2,3,5,8,1,2,1]

```
In [6]: L = [2,3,5,8,1,2,1]
        maximum(L)
```

```
Out[6]: 8
```

3) Écrire de même une fonction **minimum(L)**.

```
In [7]: # Version 1
        def minimum(L):
            m = L[0]
            for k in range(1,len(L)) :
                if L[k] < m:
                    m = L[k]
            return m

        # Version 2
        def minimum(L):
            m = L[0]
            for x in L :
                if x < m:
                    m = x
            return m
```

On n'oublie pas de tester.

```
In [8]: minimum(L)

Out[8]: 1
```

1.3 Exercice 3 - Probabilités

1) Écrire des instructions python permettant de simuler 100 lancers d'une pièce équilibrée. Le résultat sera donné sous la forme d'une liste, par exemple ['P','P','F','P',...]

Rappel : la bibliothèque **numpy.random as rd** permet de faire de l'aléatoire.
rd.randint(a,b) renvoie un entier aléatoire entre a (inclus) et b (exclus).

```
In [19]: import numpy.random as rd
         L = []
         for k in range(100):
             alea = rd.randint(0,2)  # On va dire que 0=P et 1=F
             if alea == 0 :
                 L.append('P')
             else :
                 L.append('F')
         print(L)
```

```
['P', 'P', 'F', 'F', 'F', 'F', 'P', 'P', 'P', 'F', 'F', 'P', 'F', 'P', 'F', 'P', 'P', 'F', 'F'
```

2) Compter le nombre de "Pile" obtenus.

```
In [20]: # Version 1
         n = 0 # compteur
         for k in range(100):
             if L[k] == 'P':
```

```
            n = n+1
    print(n)

    # Version 2
    n = 0 # compteur
    for x in L:
        if x == 'P':
            n = n+1
    print(n)
```

```
52
52
```

3) Retrouver le résultat avec l'instruction **L.count(...)**.

L.count(obj) renvoie le nombre de fois où l'élément *obj* est présent dans la liste *L*.

```
In [23]: L.count('P')
```

```
Out[23]: 52
```

1.4 Exercice 4 - Second maximum

Écrire une fonction **second_max(L)** qui renvoie la valeur du second maximum de L, c'est-à-dire la deuxième plus grande valeur.

```
In [33]: # Version 1
         def second_max(L):
             M = maximum(L)
             while M in L:
                 L.remove(M)
             M2 = maximum(L)
             return M2

         # Version 2
         def second_max(L):
             M = maximum(L)
             # point de départ de SM : la première valeur < M
             i = 0
             while L[i] == M :
                 i = i+1
             SM = L[i]
             for k in range(i+1,len(L)):
                 if SM < L[k] < M:
                     SM = L[k]
             return SM
```

Tester toujours avec la liste L = [8,3,5,8,1,2,3]

```
In [32]: L = [8,3,5,8,1,2,3]
         second_max(L)

Out[32]: 5
```