TP 6 – RECHERCHE DANS UNE LISTE

Préambule :

1. Ouvrir une session sur l'ordinateur en entrant Identifiant / Mot de passe ;
2. Lancer Spyder en cherchant Spyder dans la barre de recherche ;
3. Effacer le texte dans la fenêtre de gauche et écrire :

```
#NOM
#PRENOM
#TP6
```

Objectifs de ce TP :

- *Manipuler des listes ;*
- *Moyenne d'une liste ;*
- *Recherche de maximum / minimum ;*
- *Tri d'une liste.*

Toutes les réponses aux questions doivent écrites dans le programme qui sera à rendre dans le dossier Restitution de devoirs de votre classe avant 18h.

Rappels de cours sur l'utilisation des listes :

- Pour construire une liste, on énumère les éléments de la liste entre crochets, séparés par des virgules :

- Par exemple : L=[3,7,42,1,4,8,12]

- Il est également possible de créer une liste par concaténation de 2 listes :

```
c=[0,1]+[2,4]      #résultat : c=[0,1,2,4]
d=3*[0,1]          #résultat : d=[0,1,0,1,0,1]
```

- Les opérations « + » et « * » ne réalisent donc pas l'addition ou la multiplication élément par élément.

- Les éléments d'une liste sont numérotés ou indicés à partir de 0 : « L[2] » correspond donc au 3e élément de la liste L.

- Pour modifier la deuxième case de a et lui donner la valeur « 0 », il suffit d'écrire : « L[1] = 0 ».

- La fonction « len(L) » permet de connaître le nombre d'éléments de L.

- On peut aussi facilement accéder aux derniers éléments d'une liste : « L[len(L)] » ou « L[-1] » correspond au dernier élément de L par exemple.

- On peut également extraire tous les éléments situés entre les indices i (inclus) et j (exclus) d'une liste a avec la notation « a[i:j] ». Cette notation crée donc une liste de longueur j-i.

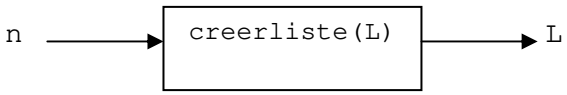
```
a=[3,7,42,1,4,8,12]
b=a[2:5]              #résultat : b = [42,1,4]
```

Exercice 1 : calcul de la moyenne d'une liste

1. Ecrire à la suite du code le commentaire `#Exercice 1`.
2. Taper la commande suivante dans la console : `from random import *`

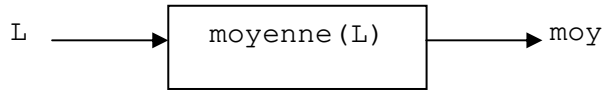
La fonction `randint(a,b)` renvoie un entier pris aléatoirement entre les valeurs a et b.

3. Créer une fonction `creerliste(n)` qui renvoie une liste de n entiers pris aléatoirement entre 0 et 100.



Indice : créer au départ une liste vide puis ajouter n éléments à l'aide d'une boucle for utilisant randint(0,100)

4. Créer à partir de la fonction précédente une liste de 100 entiers pris aléatoirement entre 0 et 100.
5. Créer une fonction `moyenne(L)` qui calcule la moyenne des éléments de la liste `L`.

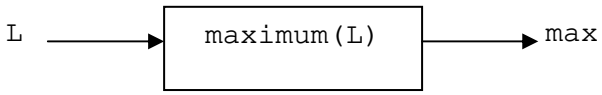


Indice : commencer par calculer la somme de tous les éléments de la liste par une boucle for puis calculer la moyenne.

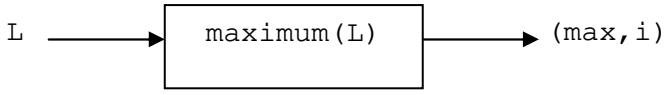
6. Tester votre fonction en comparant au calcul fait à la main.

Exercice 2 : recherche du maximum d'une liste

1. Ecrire à la suite du code le commentaire `#Exercice 2`.
2. Créer une liste aléatoire de 20 entiers compris entre 0 et 20 à partir de `creerliste(n)`.
3. Ecrire une fonction `maximum(L)` qui renvoie la valeur du maximum de la liste :



4. Tester votre fonction.
5. Modifier la fonction `maximum(L)` pour renvoyer la valeur la plus grande `max` ainsi que sa place `i` dans la liste :

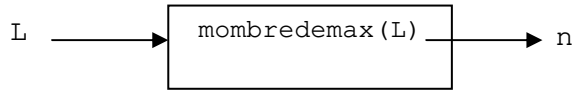


6. Tester votre fonction.

Exercice 3 : recherche de plusieurs maximum dans une liste

Le problème que l'on peut rencontrer est que le maximum d'une liste soit présent plusieurs fois dans la liste à différents rangs. Nous allons d'une part chercher combien de fois apparait le maximum et d'autre part chercher la valeur du deuxième maximum.

1. Ecrire à la suite du code le commentaire `#Exercice 3`.
2. Créer une liste de 100 éléments compris entre 0 et 20 à l'aide de la fonction `creerliste(n)`.
3. Trouver le maximum de la liste avec la fonction `maximum(L)`.
4. Créer une fonction `nombredemax(L)` qui renvoie le nombre de fois `n` ou le maximum de la fonction apparait.

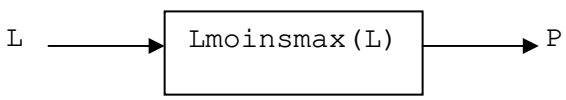


Etapes : trouver le maximum / tester chaque élément de la liste avec une boucle for / si l'élément est égal au maximum incrémenter un entier / renvoyer l'entier à la fin de boucle

5. Tester votre fonction.

Maintenant, pour trouver le deuxième maximum, nous allons créer une nouvelle liste à partir de `L` en y enlevant les maxima. Puis, en cherchant le maximum de cette nouvelle liste, nous aurons le deuxième maximum.

6. Créer la fonction `Lmoinsmax(L)` qui renvoie une liste P contenant tous les éléments de L à part les maxima.

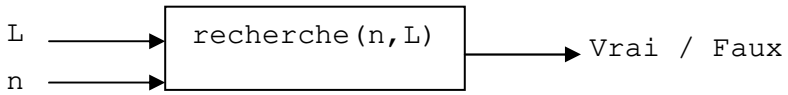


Etapes : trouver le maximum / tester chaque élément de la liste avec une boucle for / si l'élément n'est pas égal au maximum ajouter à P/ renvoyer P à la fin de la liste

7. Tester votre fonction.
8. Trouver le deuxième maximum de la liste `L` initiale.

Exercice 4 : Recherche dans une liste

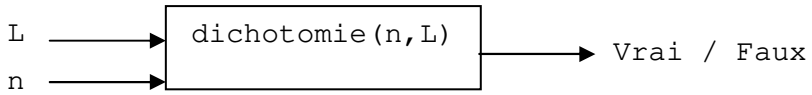
1. Ecrire à la suite du code le commentaire `#Exercice 4.`
2. Créer une liste L contenant 20 entiers pris aléatoirement entre 0 et 100.
3. Créer une fonction `recherche(n,L)` qui renvoie Vrai ou faux pour la présence de n dans la liste L.



Etapes : tester chaque élément de la liste par une boucle for / si l'élément est égal à n alors renvoyer vrai / renvoyer faux si n n'a pas été trouvé

Exercice 5 : Recherche dans une liste triée par dichotomie

1. Ecrire à la suite du code le commentaire `#Exercice 5.`
2. Créer une liste L contenant 20 entiers compris entre 0 et 100.
3. Trier la liste L avec la commande `L.sort().`
4. Créer un fonction `dichotomie(n,L)` qui recherche l'entier n dans la liste L en utilisant la méthode par dichotomie.



5. Tester votre fonction.

Rappel de la méthode dichotomie :

L'idée de la dichotomie est la suivante : on coupe la liste en deux et on recherche si la valeur x doit être recherchée dans la moitié gauche (inférieure) ou droite (supérieure). Il suffit pour cela de comparer x avec la valeur centrale de la liste, puis de réduire la recherche à la moitié de la liste, et ainsi de suite.