1 Listes unidimensionnelles

1 Exercice 1

Définir les listes suivants par **compréhension**

- La liste des couples (x, y) d'entiers positifs tels que $x + y = 100$.
- La liste des couples (x, y) d'entiers positifs tels que $x < y$ croissante selon y avec $x, y \in [|0, 10|]$.
- La liste des entiers positifs inférieur à 100 qui ne sont ni multiple de 3 ni de 5

2 Exercice 2

créer les listes suivantes :

- $L1$: liste croissante des entiers naturels pairs strictement inférieurs à 20
- $L2$: tranche (slicing) du 3ème au 6ème élément inclus de la liste $L1$ précédente
- $L3$: liste $L1$ sans les premier et dernier éléments
- $L4$: liste décroissante des entiers naturels impairs strictement inférieurs à 10
- $L5$: liste décroissante des racines carrées des entiers naturels impairs inférieurs à 10.

3 Exercice 3

(Variance)

Écrire une fonction **variance(L)** qui prend en argument une liste L et qui renvoie la variance de cette liste.

$$var(x_1, \ldots, x_n) = \frac{1}{n} \sum_{k=1}^{n} |x_k - moy|. \quad \text{avec} \quad moy = \frac{1}{n} \sum_{k=1}^{n} x_k.$$

```
>>> L=[2,4,1,1,5]
>>> variance(L)
1.52
```

4 Exercice 4

(Fusion)

Écrire une fonction **fusion(L,M)** qui prend en argument deux listes L et M et qui renvoie leur union sans doublons.

5 Exercice 5

(Décalage)

Écrire une fonction **decalage(L)** qui prend en argument une liste L et qui renvoie une copie de la liste mais les valeurs sans décaler vers la droite par une position.

```
>>> L=[1, 2, 3, 4]
>>> decalage(L)
[4, 1, 2, 3]
```

Écrire une fonction **shift(L,k)** qui prend en argument une liste L et un entier positif k et qui renvoie une copie de la liste mais les valeurs sans décaler vers la droite par k-position.

```
>>> L=[1, 2, 3, 4]
>>> shift(L,3)
[2, 3, 4, 1]
```

6 Exercice 6

(Fibonacci)

La suite de Fibonacci est définie par :

$$F_0 = 1, F_1 = 1, \quad F_n = F_{n-1} + F_{n-2} \quad \forall n \geq 1.$$

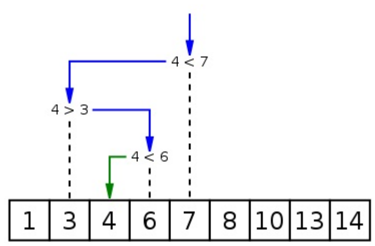
Écrire une fonction **Fibo(n)** qui prend en argument un entier positif n et qui renvoie une liste contenant les n-premiers termes de la suite.

```
>>> fibo(5)
[1, 1, 2, 3, 5, 8]
```

7 Exercice 7

(Recherche dichotomique)

Soit L une liste triées: On cherche à vérifier si un élément x appartient à L ou non. Pour cela on compare l'élément x avec la valeur de la case au milieu du L; si les valeurs sont égales, la tâche est accomplie, sinon on recommence dans la moitié de la liste pertinente.
Exemple : pour $x = 4$ et L=[1,3,4,6,7,8,10,13,14]



Écrire une fonction **rech_dico(L,x)** qui effectue une recherche dichotomique de x dans L et retourne True si $x \in L$.

8 Exercice 8

(Chiffrement de César)

Écrire une fonction **Cesar_encode(u,k)** qui prend en argument une chaine de caractères u et un entier k et qui permet de translater tous les caractères de u de k-caractères vers la droite.
Par exemple :

```
>>> Cesar_decode('Krusty',5)
Pwzxyd
```

En effet :

```
P est la cinquième lettre après K,
w est la cinquième lettre après r,
etc...
```

Chaque lettre a été décalée de 5 caractères vers la droite, et on boucle lorsqu'on a atteint la fin de l'alphabet (y est devenu d). Les caractères autres que des lettres (majuscules ou minuscules) ne sont pas transformés.
Indication :

```
# Get the ASCII number of a character
number = ord(char)

# Get the character given by an ASCII number
char = chr(number)
# The join() method is a string method and returns a string in which
#the elements of sequence have been joined by str separator.
>>> Lis=['S','A',' ','L','A','M']
>>> s=''.join(Lis)
>>> s
'SA LAM'
```

Écrire une fonction **Cesar_decode(u,k)** qui prend en argument une chaine de caractères u et un entier k et qui permet de translater tous les caractères de u de k-caractères vers la gauche.

```
>>> Cesar_decode('Pwzxyd',5)
Krusty
```

2 Listes bidimensionnelles

9 Exercice 9

① Écrire une fonction **somme(M1,M1)** qui retourne la somme de deux matrices $M1$ et $M2$.

② Écrire une fonction **produit(M1,M1)** qui retourne le produit de deux matrices $M1$ et $M2$.

③ Écrire une fonction **puissance(M,n)** qui retourne la puissance n-eme de la matrice M.