Dans ce TP, nous commençons par écrire des fonctions renvoyant des listes aléatoires d'entiers, notamment.
Puis nous rechercherons un élément donné dans une liste u contenant n valeurs numériques, ou bien une lettre ou un mot dans une chaîne de caractères. Nous travaillerons également sur les maxima d'une liste de nombres.

I Générations de listes

Dans un premier temps, on importe les modules nécessaires, notamment pour générer du hasard.

import numpy as np et *import random as rd*.

1. Expliquer ce que fait la fonction suivante :

```
def listealeatoire(a,b,n) :
    """ Cette fonction..."""
    L=[ ]
    for k in range(n) :
        L.append(rd.randint(a,b))
    return L
```

2.
```
def codage(x) :
    """ Cette fonction..."""
    if x == 0  :
        return 'A'
    elif x == 1  :
        return 'C'
    elif x == 2  :
        return 'G'
    elif x == 3  :
        return 'T'
    else :
        return 'Erreur de codage'
```

3. Utiliser la fonction ci-dessus pour en créer une nouvelle : la fonction ***chaineADNaleatoire*(n)** qui renvoie une chaîne composée de n codons, chacun d'entre eux étant composé de 3 nucléotides.

II Recherche d'un élément donné

On va écrire des fonctions qui existent déjà avec les méthodes sur les listes. On peut faire les tests avec des listes de nombres ou des chaînes de caractères.

Exercice 1: Ecrire une fonction ***appartenance*(L,x)** qui étant donnée une liste L renvoie **True** si l'élément x appartient à la liste L et **False** sinon.
On écrira deux versions de cette fonction, la première en bouclant sur les indices de L, la deuxième en bouclant sur les éléments de L.
Testez ces fonctions et comparez avec l'instruction **in**.

Exercice 2: Ecrire une fonction ***occurence*(L,x)** qui étant donnée une liste L renvoie le nombre d'occurences de l'élément x si x appartient à la liste L. Elle renverra 0 si x n'appartient pas à L.
Testez cette fonction et comparez avec la méthode **count**

Exercice 3: Ecrire une fonction ***indiceelement*(L,x)** qui étant donnée une liste L renvoie le premier indice correspondant à l'élément x si x appartient à la liste L et **None** sinon.
Testez cette fonction avec des listes aléatoires de nombres entiers et comparez avec la méthode **index**

Exercice 4: Ecrire une fonction ***Listeposition*(L,x)** qui étant donnée une liste L renvoie la liste (éventuellement vide) de toutes les positions de l'élément x dans L.

III Recherche de maxima

Pour tester vos fonctions, commencez par prendre des petites listes, par exemple, L=$[1, 6, 2, 3, 4, 2, 6]$. Vous pouvez aussi utiliser la fonction de la partie 1 pour générer des listes avec beaucoup de valeurs.

Exercice 5: Ecrire une fonction ***maximum*(L)** qui, étant donnée une liste L d'entiers renvoie **None** si la liste L est vide et l'élément maximum de L sinon.
Comparer avec la fonction ***max*** de Python.

Exercice 6: Variations autour du maximum :

a Ecrire une fonction ***Indice_max(L)*** qui, étant donnée une liste de nombres L, renvoie l'indice d'un maximum de L et le maximum, en ne parcourant qu'une seule fois la liste L.

b Dans le cas où votre maximum est présent plusieurs fois dans la liste, quel indice renvoyez-vous ?

c Ecrire une fonction ***Positions_max(L)*** qui, étant donnée une liste de nombres L, renvoie la liste des positions du maximum dans L en utilisant la fonction ***Indice_max*** (2 parcours de L)

d Ecrire une fonction ***Positions_max_opt(L)*** qui, étant donnée une liste de nombres L, renvoie la liste des positions du maximum dans L sans utiliser la fonction ***Indice_max*** (1 parcours de L)

e Ecrire une fonction ***minimum*(L)** qui, étant donnée une liste L d'entiers renvoie **None** si la liste L est vide et l'élément minimum de L sinon.

On cherche à écrire une fonction qui renvoie le second maximum d'une liste L. On fait l'hypothèse que la liste possède au moins deux éléments différents. Attention : aucune des fonctions de cette exercice ne doit modifier la liste L.

Exercice 7: Ecrire une fonction ***min_et_max(L)*** de complexité linéaire qui renvoie le minimum et le maximum d'une liste de nombres L.

Exercice 8: Utiliser la fonction ***min_et_max*** pour écrire une fonction ***Second_max(L)*** qui renvoie le second maximum de L.

On veut améliorer l'algorithme précédent de façon à ne parcourir la liste qu'une seule fois.
On suppose que les deux premiers éléments de L sont distincts.

Exercice 9: Ecrire une fonction ***Second_max_2(L*)** qui renvoie le second maximum de L en ne parcourant L qu'une seule fois

Exercice 10: Ecrire une fonction ***Indice_second_max(L)*** qui renvoie une des positions du second maximum en ne parcourant L qu'une seule fois

IV Approfondissement :

Exercice 11: Ecrire une fonction ***Deux_plus_proches_valeurs***(L) qui, étant donnée une liste de nombres L contenant au moins deux éléments, renvoie les deux valeurs les plus proches.

Exercice 12: Ecrire une fonction ***Recherche_motif_1*** (mot,texte) qui, étant donnés deux chaînes de caractères (ou deux listes) mot et texte, renvoie True si le mot est présent et **False** sinon, slices autorisés.

Exercice 13: Ecrire une fonction ***Recherche_motif_2(mot,texte)*** qui, étant donnés deux chaînes de caractères (ou deux listes) mot et texte, renvoie True si le mot est présent et **False** sinon, slices non autorisés.

Rappel : Le terme anglais slice est associé à l'idée de découpage. En langage Python, cela permet d'extraire une tranche d'un objet itérable. Si L est une liste ou une chaine de caractères, L$[i : j]$ renvoie une nouvelle liste ou chaine de caractère composée des éléments d'indices python i à $j-1$ inclus de L. Par exemple :

```
>>> L = [1, 4, 8, 2, 6, 2]
>>> L[1 : 4]
[4, 8, 2]
```