Algorithmes : Appartenance d'un élément à une liste non ordonné, complexité d'un algorithme.

Partie 1

On considère les deux algorithmes suivants :

```
Algorithme 1 :
   On considère une liste de nombres entiers
   et un entier naturel donné noté n

   Tant que la liste n'est pas vide, on tire au hasard un nombre de celle-ci
   Si ce nombre est égal à n , on renvoie True et on s'arrête, sinon on enlève ce nombre de la liste et on recommence
   Si la liste est vide, on renvoie False
```

```
Algorithme 2 :
   On considère une liste de nombres entiers
   et un entier naturel donné noté n

   On parcourt la liste du premier au dernier terme,
   si le terme rencontré est égal à n, on renvoie True et on s'arrête
   Si on a parcouru toute la liste, on renvoie False
```

et le programme python ci-dessous :

```
from random import randint
ma_liste=[9,12,1,17,3,9,10,8,5,20]

def algo1(n):
    while len(ma_liste)!= ........ :
        index=randint(0,.......................)
        if ma_liste[index]==n:
            return .................
        del ma_liste[index]
    return ...................

def algo2(n):
    for val ............................ :
        if val== .......... :
            return .........
    return ...........
```

Exercice 1

1. Compléter les deux fonctions pour qu'elles appliquent l'algorithme qui leur est associé.
2. Que font ces 2 algorithmes ?

   .....
   .....
3. Que permet de faire la fonction del de la ligne 9 ? .....
   .....

Partie 2 : efficacité d'un algorithme

En informatique, il est souvent nécessaire de comparer des algorithmes différents mais effectuant la même tâche pour déterminer lequel est le plus efficace.

> L'**analyse de la complexité d'un algorithme** consiste en l'étude formelle de la quantité de ressources (par exemple de temps ou d'espace) nécessaire à l'exécution de cet algorithme.
>
> https://fr.wikipedia.org/wiki/Analyse_de_la_complexit%C3%A9_des_algorithmes

1.Analyse de la complexité en termes d'opérations.

On peut considérer en première approche le nombre d'opérations (accès en mémoire, comparaisons, calculs) nécessaires pour effectuer un algorithme.

Exercice 2

On considère l'algorithme n°2 et la liste `ma_liste=[9,12,1,17,3,9,10,8,5,20]`

1. Combien d'opérations sont nécessaires pour que l'algorithme2 détermine que 9 appartient à `ma_liste` ? .....
2. Combien d'opérations sont nécessaires pour que l'algorithme2 détermine que 10 appartient à `ma_liste` ? .....
3. Combien d'opérations sont nécessaires pour que l'algorithme2 détermine que 15 n'appartient pas à `ma_liste` ? .....
4. Mêmes questions avec l'algorithme 1. .....
   .....
   .....
   .....
   .....
5. On considère à présent une liste de longueur n, déterminer quel est le nombre maximal d'opérations nécessaires pour déterminer si un nombre appartient ou non à cette liste avec ces 2 algorithmes.
   .....

   .....