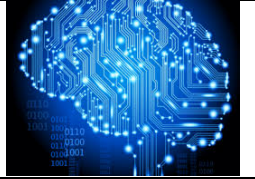


Terminale NSI	Programmation	Module 1&4 :	
TD. N°2	Révision : Langage et programmation.		

1. Les nombres de Armstrong

Un nombre de Armstrong est un entier positif égal à la somme des cubes de ses chiffres

Exemple : $153 = 1 + 125 + 27$

Écrire une fonction qui retourne les nombres de Armstrong inférieur à un entier.

Exemple d'exécution :

Veillez entrer un entier :2000

```
>>>1
```

```
153
```

```
370
```

```
371
```

```
407
```

2. Diviseurs d'un entier

Écrire une fonction qui retourne une liste de diviseurs d'un entier.

Exemple d'exécution :

Veillez entrer un entier : 126

```
>>>[1, 2, 3, 6, 7, 9, 14, 18, 21, 42, 63, 126]
```

3. Nombres amicaux

a et b deux entiers positifs sont dits amicaux ou amiables ou aimables si chacun des deux nombres est égal à la somme des diviseurs stricts(diviseurs autres que lui-même) de l'autre .

Exemple :

- la somme des diviseurs de 220 (en excluant 220) : $1+2+4+5+10+11+20+22+44+55+110=284$
- la somme des diviseurs de 284 (en excluant 284) : $1+2+4+71+142=220$

Écrire une fonction qui renvoie True si deux nombres sont amicaux

Exemple d'exécution :

Veillez entrer un entier n1 : 220

Veillez entrer un entier n2 : 284

```
>>>True
```

4. Supprimer les éléments en double

Écrire une fonction qui supprime les éléments en double d'une liste.

Exemple d'exécution :

liste de départ : L=[1,2,5,8,6,2,5,9,1,8,8]

```
>>>affichage : [1, 2, 5, 8, 6, 9]
```

1. Des listes

Les variables `liste_eleves` et `liste_notes` ayant été préalablement définies et étant de même longueur, la fonction `meilleures_notes` renvoie la note maximale qui a été attribuée, le nombre d'élèves ayant obtenu cette note et la liste des noms de ces élèves. Compléter le code Python de la fonction `meilleures_notes` ci-dessous.

```
liste_eleves = ['a','b','c','d','e','f','g','h','i','j']
liste_notes = [1, 40, 80, 60, 58, 80, 75, 80, 60, 24]

def meilleures_notes():
    note_maxi = 0
    nb_eleves_note_maxi = ...
    liste_maxi = ...

    for compteur in range(...):
        if liste_notes[compteur] == ...:
            nb_eleves_note_maxi = nb_eleves_note_maxi + 1
            liste_maxi.append(liste_eleves[...])
        if liste_notes[compteur] > note_maxi:
            note_maxi = liste_notes[compteur]
            nb_eleves_note_maxi = ...
            liste_maxi = [...]
    return (note_maxi,nb_eleves_note_maxi,liste_maxi)
```

Une fois complété, le code ci-dessus donne

```
>>> meilleures_notes()
(80, 3, ['c', 'f', 'h'])
```

2. Recherches

La fonction `recherche` prend en paramètres deux chaînes de caractères `gene` et `seq_adn` et renvoie `True` si on retrouve `gene` dans `seq_adn` et `False` sinon. Compléter le code Python ci-dessous pour qu'il implémente la fonction `recherche`.

```
def recherche(gene, seq_adn):
    n = len(seq_adn)
    g = len(gene)
    i = ...
    trouve = False
    while i < ... and trouve == ... :
        j = 0
        while j < g and gene[j] == seq_adn[i+j]:
            ...
        if j == g:
            trouve = True
        ...
    return trouve
```

Exemples :

```
>>> recherche("AATC", "GTACAAATCTTGCC")
True
>>> recherche("AGTC", "GTACAAATCTTGCC")
False
```