Exercice 4 :

Codez une fonction reponse4 prenant en argument une liste d'entiers nombres. Elle renvoient une chaine de caracteres precisant, l'ensemble des diviseurs positifs d'au moins un element de nombres et le nombre d'elements qu'il divise. On ordonne par nombre d'apparitions decroissantes et par valeur du diviseur croissant. On aurait par exemple

- reponse4(nombres=[])
- print(reponse4(nombres=[1, 2, 3, 9]))

  1: 4 apparitions

  3: 2 apparitions

  2: 1 apparition

  9: 1 apparition

On va commencer par créer une liste ('apparitions') qui va contenir le nombre de multiples (dans la liste) pour chacun des nombres contenu dans la liste passée en paramètre

Entrée [3]:
```
def reponse4(liste:list):
    apparitions = []
    for nombre in liste:
        compteur = 0
        for nbre in liste:
            if nbre % nombre == 0:
                compteur += 1
        apparitions.append(compteur)
    return apparitions
```

Exemple

Entrée [4]:
```
print(reponse4([2,1,6,2,3]))
```

```
[3, 5, 1, 3, 2]
```

Nous allons maintenant ajouter une condition au cas où 0 appartienne à la liste

Entrée [6]:
```
def reponse4(liste:list):
    apparitions = []
    for nombre in liste:
        compteur = 0
        if nombre != 0:
            for nbre in liste:
                if nbre % nombre == 0:
                    compteur += 1
        apparitions.append(compteur)
    return apparitions
```

Exemple

Entrée [7]:
```
print(reponse4([2,0,1,6,2,3]))
```

```
[4, 0, 6, 2, 4, 3]
```

Nous allons maintenant créer un dictionnaire prenant comme clés, les éléments de la liste passée en paramètre et comme valeurs, les éléments de la liste 'apparitions'

Entrée [8]:
```
def reponse4(liste:list):
    apparitions = []
    for nombre in liste:
        compteur = 0
        if nombre != 0:
            for nbre in liste:
                if nbre % nombre == 0:
                    compteur += 1
        apparitions.append(compteur)
    dico = dict(zip(liste,apparitions))
    return dico
```

Exemple

Entrée [9]:
```
print(reponse4([2,0,1,6,2,3]))
```

```
{2: 4, 0: 0, 1: 6, 6: 2, 3: 3}
```

Nous allons à présent trier ce dictionnaire par ordre décroissant des valeurs et par ordre croissant des clés

Entrée [10]:
```
def reponse4(liste:list):
    apparitions = []
    for nombre in liste:
        compteur = 0
        if nombre != 0:
            for nbre in liste:
                if nbre % nombre == 0:
                    compteur += 1
        apparitions.append(compteur)
    dico = dict(zip(liste,apparitions))
    dico_trie = dict(sorted(dico.items(),key=lambda tr:(-tr[1],tr[0])))
    return dico_trie
```

Exemple

Entrée [11]:
```
print(reponse4([2,0,1,6,2,3,5]))
```

```
{1: 7, 2: 4, 3: 3, 5: 2, 6: 2, 0: 0}
```

Maintenant, nous allons rendre le dictionnaire trié en un objet itérable puis, le parcourir afin d'afficher le résultat tel que demandé dans l'exercice

Entrée [12]:
```
def reponse4(liste:list):
    apparitions = []
    for nombre in liste:
        compteur = 0
        if nombre != 0:
            for nbre in liste:
                if nbre % nombre == 0:
                    compteur += 1
        apparitions.append(compteur)
    dico = dict(zip(liste,apparitions))
    dico_trie = dict(sorted(dico.items(),key=lambda tr:(-tr[1],tr[0])))
    for cle,valeur in dico_trie.items():
        if valeur > 1:
            print("{}: {} apparitions".format(cle,valeur))
        else:
            print("{}: {} apparition".format(cle,valeur))
    return ""
```

Entrée [13]:
```
print(reponse4([2,0,1,6,2,3,5]))
```

```
1: 7 apparitions
2: 4 apparitions
3: 3 apparitions
5: 2 apparitions
6: 2 apparitions
0: 0 apparition
```

Nous allons maintenant gérer le cas où l'utilisateur entre une liste vide

Entrée [14]:
```
def reponse4(liste:list):
    if len(liste)==0:
        return "Votre liste est vide !!!"
    else:
        apparitions = []
        for nombre in liste:
            compteur = 0
            if nombre != 0:
                for nbre in liste:
                    if nbre % nombre == 0:
                        compteur += 1
            apparitions.append(compteur)
        dico = dict(zip(liste,apparitions))
        dico_trie = dict(sorted(dico.items(),key=lambda tr:(-tr[1],tr[0])))
        for cle,valeur in dico_trie.items():
            if valeur > 1:
                print("{}: {} apparitions".format(cle,valeur))
            else:
                print("{}: {} apparition".format(cle,valeur))
        return ""
```

Exemple

Entrée [15]:
```
print(reponse4([]))
```

```
Votre liste est vide !!!
```

Entrée [16]:
```
print(reponse4([2,0,1,6,2,3,5]))
```

```
1: 7 apparitions
2: 4 apparitions
3: 3 apparitions
5: 2 apparitions
6: 2 apparitions
0: 0 apparition
```

Pour terminer, nous allons gérer d'eventuelles erreurs pouvant survenir lors de l'exécution de notre fonction

Entrée [2]:
```
def reponse4(liste:list):
    try:
        if len(liste)==0:
            return "Votre liste est vide !!!"
        else:
            apparitions = []
            for nombre in liste:
                compteur = 0
                if nombre != 0:
                    for nbre in liste:
                        if nbre % nombre == 0:
                            compteur += 1
                apparitions.append(compteur)
            dico = dict(zip(liste,apparitions))
            dico_trie = dict(sorted(dico.items(),key=lambda tr:(-tr[1],tr[0]
            for cle,valeur in dico_trie.items():
                if valeur > 1:
                    print("{}: {} apparitions".format(cle,valeur))
                else:
                    print("{}: {} apparition".format(cle,valeur))
            return ""
    except:
        return " Désolé, une erreur est survenue !!! \n Veuillez vérifier le
```

Exemple

Entrée [18]:
```
print(reponse4([]))
```

```
Votre liste est vide !!!
```

Entrée [19]:
```
print(reponse4([2,0,1,6,2,3,5]))
```

```
1: 7 apparitions
2: 4 apparitions
3: 3 apparitions
5: 2 apparitions
6: 2 apparitions
0: 0 apparition
```

Entrée [3]:
```
print(reponse4(["a",2,3]))
```

```
Désolé, une erreur est survenue !!!
Veuillez vérifier le paramètre que vous avez entré.
```

Entrée [22]:
```
print(reponse4([-5,4,2,2,3,2,7.3,5,0]))
```

```
2: 5 apparitions
-5: 3 apparitions
5: 3 apparitions
3: 2 apparitions
4: 2 apparitions
7.3: 2 apparitions
0: 0 apparition
```

Entrée [23]:
```
print(reponse4([4,4,4,4]))
```

```
4: 4 apparitions
```

Entrée [24]:
```
print(reponse4([1]))
```

```
1: 1 apparition
```

Entrée [ ]:
```

```