AGR. Algorithmes. Avancés. Exercices. Corrigés

***Exercice 1* : Vrai / Faux**

Q.1 : VRAI. Voir cours.
Q.2 : FAUX. Par exemple, si le dernier élément de la liste est le plus petit.
Q.3 : FAUX. Voir cours.
Q.4 : VRAI. Voir cours.
Q.5 : VRAI. Voir cours.
Q.6 : VRAI. Voir cours.

***Exercice 2* : QCM**

Q.1 : Réponse 1. Voir cours.
Q.2 : Réponse 2. Si la liste est déjà triée, chaque élément n'est testé dans ce cas qu'une fois.
Q.3 : Réponse 3. Aide : Bien exécuter l'algorithme pas à pas.

```
def monte(liste) :
    for i in range(len(liste) - 1) :
        if liste[i] > liste[i+1] :
            liste[i], liste[i+1] = liste[i+1], liste[i]

    return liste

ma_liste = [12,5,13,8,11,6]
print(monte(ma_liste))
```

```
[5, 12, 8, 11, 6, 13]
```

Q.4 : Réponse 4. Aide : Chercher des contre-exemples.
Pour 1/ : 8 = 4 + 4 mais 5 + 1 + 1 + 1 avec l'algorithme glouton.
Pour 2/ : 6 = 3 + 3 mais 4 + 1 + 1 avec l'algorithme glouton.
Pour 3/ : 7 = 4 + 3 mais 5 + 1 + 1 avec l'algorithme glouton.

Q.5 : Réponse 1. On compare les mots lettre par lettre.
Remarque : les lettres majuscules sont comptées « plus grandes » que les lettres minuscules. Ainsi, un « A » sera après un « z » (Penser au code ASCII).

***Exercice 3* :**

1/ Il s'agit d'un tuple
2/ Voici un exemple de programme

```
# Système de rendu de monnaie très simplifié
syst = (1, 2, 5)
rendu = 0
combinaisons = []

for i in range(9) :  # Maximum : 9 pièces de 1 centime
    for j in range(5) : # Maximum : 4 pièces de 2 centimes
        for k in range(2) : # Maximum : 1 pièce de 5 centimes
            if i*syst[0] + j*syst[1] + k*syst[2] == 8 :
                combinaisons.append([i,j,k])

print(combinaisons)
```

```
[[0, 4, 0], [1, 1, 1], [2, 3, 0], [3, 0, 1], [4, 2, 0], [6, 1, 0], [8, 0, 0]]
```

3/ La meilleure solution est le triplet [1 , 1, 1].

Exercice 4 *:*

Voici un exemple de programme :

```
# Choix de la valeur
# Plus la valeur de l'objet est élevée, plus il est intéressant
def valeur(obj) :
    return obj[1]
```

```
# Implémentation de la stratégie gloutonne
def glouton(liste_obj, masse_max, choix) :
    liste_triee = sorted(liste_obj, key = choix, reverse = True)
    resultat = []
    valeur = 0
    masse = 0
    i = 0 # Compteur pour l'indice de la liste

    # Tant qu'il reste des objets et que la limite (masse)
    # n'est pas atteinte
    while liste_triee and masse < masse_max and i < len(liste_triee) :
        nom, val, nbre = liste_triee[i]

        # On reste sur le même objet tant qu'on atteint pas la masse limite
        while masse < masse_max :
            resultat.append(nom)
            masse += 1
            valeur += val
            nbre -= 1
            liste_triee[i][2] -= 1 # Un objet a été utilisé
            # Si on a chargé tous les objets disponibles
            # on repart au début de la liste et quitte la boucle
            if not nbre :
                del liste_triee[i]
                i = 0
                break;

    return resultat, valeur
```

```
# Liste d'objets : nom, rapport, nombre
objets = [ ['Objet 1', 9, 14], ['Objet 2', 16, 2], ['Objet 3', 4 , 5],
         ['Objet 4', 5, 1], ['Objet 5', 3 , 6], ['Objet 6', 10 , 8] ]

# La valeur est le meilleur choix (rapport pour un kg)
print("Choix par rapport : ", glouton(objets, 15, valeur))
```

```
Choix par rapport :  (['Objet 2', 'Objet 2', 'Objet 6', 'Objet 6', 'Objet 6', 'Objet 6', 'Objet 6',
t 6', 'Objet 1', 'Objet 1', 'Objet 1', 'Objet 1', 'Objet 1'], 157)
```

On obtient bien le résultat proposé pour une valeur de 157 euros.