

**Problématique générale :** Ti' Payet doit traverser l'océan indien avec un bateau et il a droit à une charge maximale de 31181 kg. Il doit transporter un maximum de containers dont la masse et la valeur sont variables. Il compte revendre au meilleur prix ses marchandises.

Comment optimiser le remplissage du bateau afin de ne pas dépasser la capacité totale de 31181 kg tout en ayant la plus grande valeur possible ?



Ti'Payet se rend compte que le problème est similaire à un problème étudié en classe d'informatique quand il était plus jeune : C'est le problème du sac à dos : quelles boîtes choisir afin de maximiser la somme emportée tout en ne dépassant pas les 15 kg autorisés ? Mais cela fait bien longtemps qu'il n'a plus codé et il a besoin de ton aide...

Il a retrouvé dans ses vieux cours plusieurs programmes, il aimerait bien les tester pour trouver le meilleur compromis, mais les termites ont mangé des morceaux de feuilles. Sauras-tu aider ti'Payet à reconstituer les programmes à partir des éléments dont tu disposes ?

Ti'Payet propose de réaliser un premier algorithme :

- 1- Algo1 : Classement suivant les masses** Algorithme à coder : a - On classe les objets dans une liste de la masse la plus faible à la masse la plus élevée b - On parcourt la liste en additionnant les valeurs et les masses . c - On s'arrête dès que l'on ne peut plus ajouter de masse (dépassement capacité)

```

1  # ALGORITHME GLOUTON
2  # Probleme du sac à dos
3  #
4  # Algo 1 : classement suivant les masses
5  # liste_objet est une liste composée de triplets (numero , masse , valeur)
6  # liste_objet = [ ( 1, masse , valeur ) , ( 2 , masse , valeur ) , .... ]
7  # par exemple [(1, 4kg , 8€) , (2 , 5kg , 10€) , (3 , 8kg , 15€) ...etc ]
8  #
9
10 liste_objet = [(1 , 4 , 8) , (2 , 5 , 10) , (3 , 8 , 15) , (4 , 3 , 4)]
11 masse_max = 11
12
13 # objet[0] = numero , objet[1] = masse , objet[2] = valeur :
14
15 liste_triee = sorted(liste_objet, key=lambda objet: objet[1]) # tri suivant masse
16
17 print ("liste triée : ", liste_triee)
18
19 masse_totale = 0
20 objet_pris = [] # liste des numéros des objets pris
21 valeur_totale = 0
22
23 # on parcourt la liste triée et on additionne les masses
24 # si masse totale <= masse_max
25
26 for objet in liste_triee:
27     if (masse_totale + objet[1]) <= masse_max :
28         masse_totale = masse_totale + objet[1]
29         objet_pris = objet_pris + [objet[0]]
30         valeur_totale = valeur_totale + objet[2]
31
32 print ("Masse totale : ",masse_totale , "kg")
33 print ("Valeur totale : ",valeur_totale , "€")
34 print ("liste objets pris",objet_pris)

```



Taper et tester le programme.

Vérifier en faisant le calcul sur papier que le programme donne une solution cohérente.

La solution est-elle optimale ?  
(Rechercher sur papier la solution optimale)

2- Algo2 : Classement suivant les valeurs

Le classement se fait maintenant de la valeur la plus forte à la plus faible :

Modifier la ligne 15 du programme comme suit ( remplacer le « ? » par la bonne valeur ) :

```
liste_triee=sorted(liste_objet, key=lambda objet: objet[?], reverse=True) #tri suivant valeur
```



Il faut remplacer la valeur ? par 2.

Notez l'option « reverse » de la fonction de tri python 'sorted' qui permet d'inverser l'ordre de tri pour avoir ici du plus grand au plus petit.

Le critère de tri (key) est une fonction lambda, c'est un peu spécial, nous la verrons plus en détail ultérieurement. Pour le moment, sachez juste que objet ici est juste un paramètre (il pourrait avoir n'importe quel nom)

La suite de l'algorithme ne change pas : Tester et valider par une vérification sur papier.

**Résultats du programme**

liste triée: [(3, 8, 15), (2, 5, 10), (1, 4, 8), (4, 3, 4)]

Masse totale: 11 kg

Valeur totale: 19 €

liste des objets pris: [3, 4]

**Vos calculs :**

liste donnée:

(1, 4, 8), (2, 5, 10), (3, 8, 15), (4, 3, 4)

k k k k

k=kg

8>5>4>3

donc position:

1. (3, 8, 15)

2. (2, 5, 10)

3. (1, 4, 8)

4. (4, 3, 4)

**Validation prof :**3- Algo 3 : classement en fonction du rapport valeur/masse (du plus fort au plus faible)

On met le script juste avant l'affichage des valeurs, avant la boucle for du programme originalement donné.

```
liste_objet = [(1, 4, 8), (2, 5, 10), (3, 8, 15), (4, 3, 4)]
masse_max = 11

# objet[0] = numero , objet[1] = masse , objet[2] = valeur , objet[3] = valeur/masse:
# on veut créer (en faisant une boucle for) une liste2 qui intègre le rapport valeur/masse:
# de la forme [ ( 1, masse , valeur , valeur/masse ) , ( 2 , masse , valeur , valeur/masse ) , .... ]
# compléter le code ci-dessous et intégrer le dans votre programme au bon endroit :

liste2 = []
for objet in liste_objet:
    liste2 = liste2 + [ (objet[0], objet[1], objet[2], objet[2]/objet[1]) ]

print (liste2)

liste_triee = sorted(liste2, key=lambda objet: objet[3], reverse = True) # tri suivant valeur/masse
```

On laisse les valeurs d'initialisation tout en haut du document

3, objet = valeur renplies par Moi

Vous devriez obtenir après exécution le résultat suivant :

```
[(1, 4, 8, 2.0), (2, 5, 10, 2.0), (3, 8, 15, 1.875), (4, 3, 4, 1.3333333333333333)]
liste triée : [(1, 4, 8, 2.0), (2, 5, 10, 2.0), (3, 8, 15, 1.875), (4, 3, 4, 1.3333333333333333)]
Masse totale : 9 kg
Valeur totale : 18 €
liste objets pris [1, 2]
```

Vérifier par un calcul la conformité des résultats.

Tester votre programme avec les 19 containers de ti'payet et comparer avec la solution optimale

Masse\_max = 31181kg □ variable à mettre à jour dans le programme

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
	40006	4766	532034	4830	32708	38789656	2078	132310	3020	2624	28909280	2744	3114	10372	7390	1142	454990	2945	5

Comparez les résultats avec la Solution optimale : poidsTotal

= 30996 valeurTotale =

12248

7390+2744+7280+3926+9656= 30996kg

2945+1022+2890+1513+3878=12248

Je n'ai pas put faire cet exercice car les valeurs sont illisibles.

Conclusion : Ces 3 programmes font partie de la famille des algorithmes « gloutons »

Les algos gloutons ne donnent pas (ou rarement) les solutions optimales mais présentent l'intérêt d'être rapide à l'exécution sans utiliser trop de ressources mémoire. Ils sont aussi généralement facile à coder et peuvent convenir comme première approche simple à beaucoup de problèmes.

Vérifier par un calcul la conformité des résultats.

Tester votre programme avec les 19 containers de ti'payet et comparer avec la solution optimale

**Masse\_max = 31181kg** → variable à mettre à jour dans le programme

Valeur masse
1945 4990
321 1142
<b>2945 7390</b>
4136 10372
1107 3114
<b>1022 2744</b>
1101 3102
<b>2890 7280</b>
962 2624
1060 3020
805 2310
689 2078
<b>1513 3926</b>
<b>3878 9656</b>
13504 32708
1865 4830
667 2034
1833 4766
16553 40006

Comparez les résultats avec la  
Solution optimale :  
poidsTotal = 30996  
valeurTotale = 12248  
 $7390+2744+7280+3926+9656= 30996\text{kg}$   
 $2945+1022+2890+1513+3878=12248$