|  |
| --- |
| TD SAC A DOS Algorithmes Gloutons |

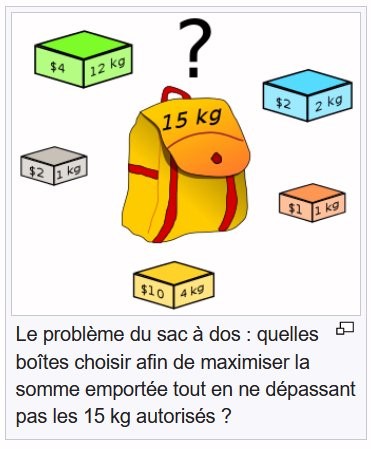
TD Sac à dos 1ère 6 NSI

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... |

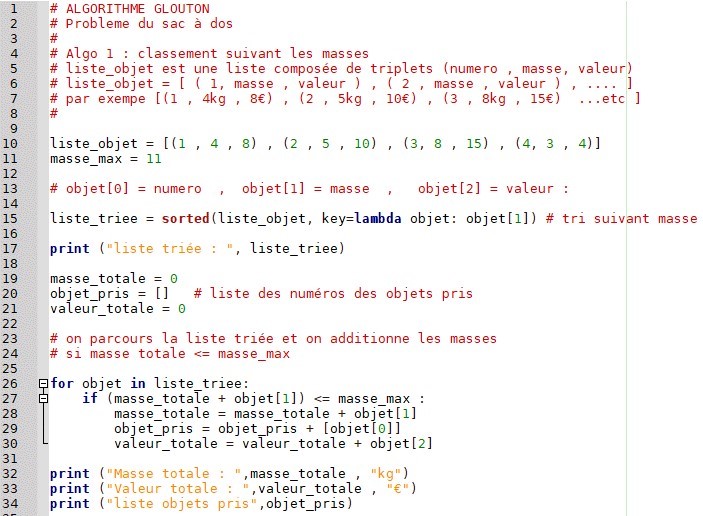


Il a retrouvé dans ses vieux cours plusieurs programmes, il aimerait bien les tester pour trouver le meilleur compromis, mais les termites ont mangé des morceaux de feuilles. Sauras-tu aider ti'Payet à reconstituer les programmes à partir des éléments dont tu disposes ?

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 parcours 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)



Taper et tester le

programme.

Vérifier en faisant

le calcul sur papier

que le programme

donne une solution

cohérente.

La solution est-ell

e optimale ?

(

Rechercher sur p

apier la solution

optimale)



Cours HATTEMER OMJS Page 1 sur 2 TD Sac à dos 1ère NSI

# 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

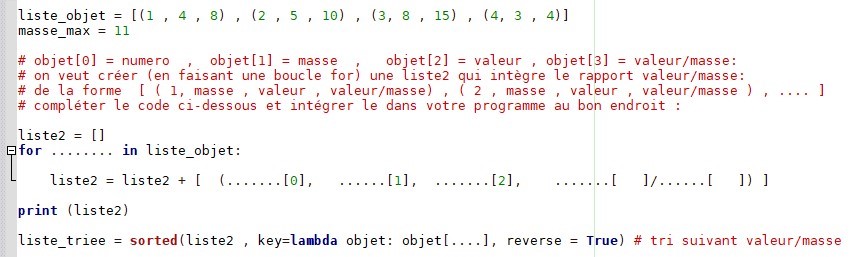
Notez l’option « reverse » de la fonction de tri python ‘sorted’ qui permet d’inverser l’ordre de tri pour avoir ici du plus grand au plus petit.

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)

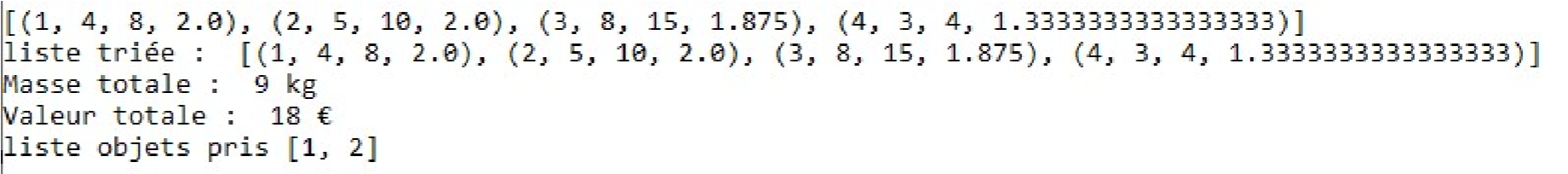
La suite de l’algorithme ne change pas : Tester et valider par une vérification sur papier.

|  |  |  |  |
| --- | --- | --- | --- |
| |  | | --- | | Résultats du programme : | | |  | | --- | | Vos calculs : | |
| |  | | --- | | Validation prof : | |

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



Vous devriez obtenir après exécution le résultat suivant :



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  4990  1945  321  1142    7390  2945  10372  4136  3114  1107    2744  1022  3102  1101  2890 7280  2624  962  3020  1060  805  2310  2078  689    3926  1513  3878 9656  13504  32708  4830  1865  2034  667  4766  1833  16553  40006 | |  | | --- | | Comparez les résultats avec la Solution optimale : poidsTotal = 30996 valeurTotale = 12248  7390+2744+7280+3926+9656= 30996kg  2945+1022+2890+1513+3878=12248 | |

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.

Cours HATTEMER OMJS Page 2 sur 2