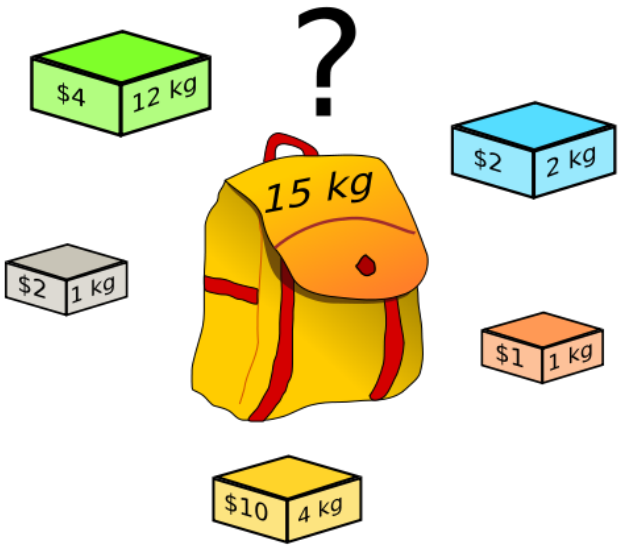
Problème du sac à dos

Le problème est de remplir un sac à dos pour transporter un maximum de valeurs. Le sac à dos peut emporter une charge maximum « M » et chacun des N articles transportés possède son propre poids P et une valeur définie V

Ex



CC BY-SA 2.5, https://commons.wikimedia.org/w/index.php?curid=985491

Dans l'exemple ci-dessus :

- le poids total doit être inférieur ou égal à 15
- le poids des articles : 1,2,4,12 kilogrammes
- les valeurs des articles 1,2,4,10

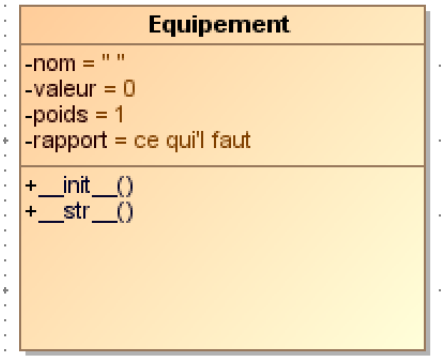
Algorithme glouton

Une solution efficace consiste à utiliser un algorithme glouton. L'idée est de calculer le rapport valeur / poids pour chaque objet et de trier l'objet sur la base de ce rapport calculé.

On prend l'objet avec le ratio le plus élevé et on ajoute cet objet et on recommence tout en vérifiant que le poids cumulé des différents objets ajoutés ne dépasse pas le poids transportable par le sac à dos.

3.1. Création de l'objet « Equipement »

Les attributs et les méthodes son présentées sur le diagramme de classe UML à droite.



« rapport » est le rapport de la valeur sur le poids (d'où les valeurs par défaut.

Pour la suite du programme et les tests vous créerez les objets suivants :

```
tente = Equipement("tente",4,12)
tapis = Equipement("tapis",10,4)
gourde = Equipement("gourde",2,2)
chaussures = Equipement("chaussures",2,1)
rechaud = Equipement("rechaud",1,1)
```

Et la liste

```
l = [tente,tapis,gourde,chaussures,rechaud]
```

__init__() constructeur

__str__() permet d'afficher les caractéristique d'un « Equipement »

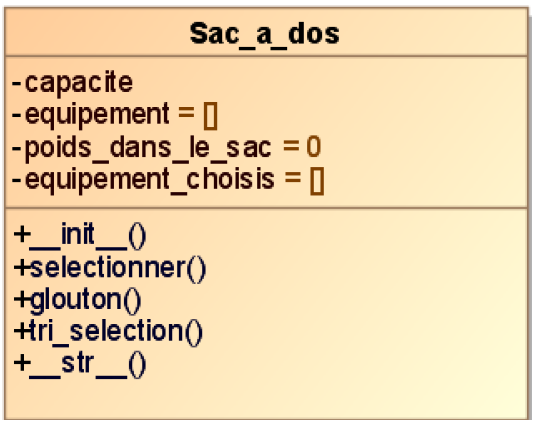
```
tente = Equipement("tente",4,12)


print(tente)
```

Affiche

```
tente 4 12 0.3333333333333333
```

3.2. Création de l'objet « Sac_a_dos »

La classe « sac_a_dos » est représentée par le diagramme de classe UML :



- capacite est le poids maximal transportable par le sac
- equipement est une liste d'objets « Equipement » qui peuvent être mis dans le sac liste vide au départ
- poids_dans_le_sac vaut 0 au départ
- equipement_choisis liste vide au départ qui va se remplir des équipements sélectionnés pour remplir le sac
- tri_selection() permet de trier la liste equipement par rapport (poids/valeur) croissant
- glouton renvoi la liste equipement_choisis
- sélectionner permet de sélectionner soit même les « Equipement »

3.2.1. Créer la classe de l'objet « Sac_a_dos »

Faire le cadre de l'objet dans les méthode autre que __init__() et __str__() mettez « pass »

Pour ne pas bloquer le programme. Tester avec un Sac_a_dos de capacite 33 kg et avec la liste d' « Equipement » précédente.

Créer un objet Sac Sac_a_dos

```
l = [tente,tapis,gourde,chaussures,rechaud]
Sac =  Sac_a_dos(33,l)
```

Vérifier ses attributs

3.2.2. Créer la méthode `def tri_selection(self) :`

Adapter le tri par sélection pour trier les « Equipement » par ordre de rapport (valeur/poids) croissant

Tester la

3.2.3. Créer la méthode `def gouton(self):`

En vous inspirant du rendu de pièce et en prenant le rapport (valeur/poids) comme référence établir le glouton qui remplit l'attribut equipement_choisis.

Tester la

3.2.4. Compléter la méthode `def selectionner(self) :`

```
def selectionner(self) :
    rep = "oui"
    while rep == "oui" :
        nom = input("Donner le nom de l'objet à mettre dans le sac'?")
        poids = int(input("Donner le poids de l'objet"))
        valeur  = int(input("Donner sa valeur"))
        eq = Equipement(??????,???????,???????)
        self.equipement.append(??????)
        rep = input("Voulez vous ajouter un autre équipement  : oui /non ?")
```

Vérifier que vous créer bien une liste d'objets Equipement

Pour bien faire il faudrait aussi soit accepter les Equipement déjà présent ou repartir sur une liste vide.

4. Effectuer la programmation finale du projet en affichant « équipement_choisis » de l'exemple