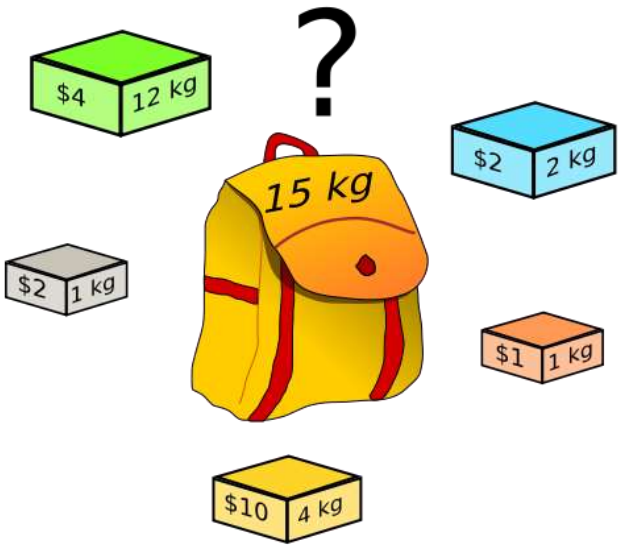
## 3. Problème du sac à dos

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






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 jusqu'à ce qu'on ne puisse plus en ajout

### 3.1. Création d'une liste de 2-uplets [valeurs, poids]

Créer une liste « articles » de 2-uplets 3.1. [valeurs, poids]

### 3.2. Ajouts du rapports valeur/poids

Définir la fonction valeurpoids qui ajoute pour chaque élément de articles le rapport valeur /poids

### 3.3. Trier la liste en fonction de ce rapport dans l'ordre décroissant ( utiliser sort ici)

Vous vous inspirerez fortement de :

https://www.w3schools.com/python/ref_list_sort.asp

### 3.4. Réalisation du glouton

On définit la fonction sac_a_dos. Elle prend comme paramètre la liste précédente et le poids du sac.

Elle retourne la liste des valeurs et leur total.

Elle sélectionner l'article à mettre dans le sac à dos avec avec le plus grand rapport valeur /poids en vérifiant que son poids peut être transporté. Puis elle retire son poids au poids total et elle recommence tant que l'opération est possible.