* Quelle est l’utilité d’une échelle dans l’informatique et dans d’autres domaines?

Les échelles sont utilisées pour**permettre aux travailleurs de disposer d'un appui sûr**. Dans l’informatique, un des principaux avantages des échelles est justement d’avoir des options quantifiées : on peut réaliser des statistiques sur les résultats, en calculant des moyennes, des écarts types, des différences entre populations… Quand un ordinateur mesure des grandeurs, il crée, par exemple, une échelle logarithmique, comme pour mesurer la luminosité de la lumière ou l’intensité d’un tremblement de terre. Mais il existe plein d’autres échelles. On peut citer l’échelle naturelle, l’échelle cartographique, graphique, numérique…

Ici on va se centrer sur une échelle de monnaie qui se base sur celle de l’Euro.

D’abord , il faut savoir qu’en simplifiant le système des prix, cette échelle de monnaie facilite le commerce, les échanges marchands.. Par exemple, un pantalon vaut 50 euros ou un carambar vaut 10 centimes en France. Par conséquent, la valeur d’un pantalon est égale à 500 fois la valeur d’un carambar. Cette échelle de monnaie est donc un instrument de mesure des marchandises qui simplifie le système des prix.

Maintenant, on va essayer d’étudier un contexte où il est possible d’utiliser cette échelle sous forme d’algorithme.

Pour cela on va utiliser un célèbre algorithme qui est l’algorithme glouton. C’est en fait un procédé qui consiste à utiliser d’abord le plus grand nombre d’une échelle jusqu’à ce que le contenu ou la somme soit assez petite pour utiliser le deuxième plus grand nombre de l’échelle jusqu’à ce que la somme soit encore plus petite pour utiliser le troisième plus grand nombre de l’échelle etc…

Le contexte est le suivant: on a la fonction *rendu\_monnaie* qui prend en paramètres deux nombres entiers positifs *s\_due* (la somme à rendre) et *s\_versee* (la somme versée par le client par ex). Cette fonction permet de procéder au rendu de la monnaie càd de la différence *s\_versee* *– s\_due* pour un achat effectué avec le système de pièces de l’Euro. On utilise pour cela un algorithme qui commence par rendre le maximum de pièces de la plus grande valeur et ainsi de suite. La fonction renvoie la liste des pièces qui composent le rendu. On précise que toutes les sommes sont exprimées en euros et que les valeurs possibles pour les pièces sont [1, 2, 5, 10, 20, 50, 100, 200].

Ainsi, prenons par exemple l’instruction *rendu\_monnaie (452, 500)*.  Avec 452 euros qui est à payer par le client et 500 euros la somme versée par le client. En effet, la somme à rendre est de 48 euros, 500-452. On a donc pris la plus grande valeur de la liste qui reste inférieure à 48. Ici on a commencé par 20 car 50>48. Donc on commence avec 20 qu’on soustrait après à la somme à rendre : 48-20=28. Puisque 28>20, on ajoute encore 20 à la liste puis on le soustrait à la somme à rendre. Donc 28-20=8. Là on change de pièces, on passe à 5 car 20>8 et 8>5. On fait ça jusqu’à ce que la somme soit nulle. On obtient au final, la liste [20, 20, 5, 2, 1]. Pour vérifier que le compte est bon on additionne tous les nombres de la liste et il faut que ce soit égal au rendu.

Maintenant, voyons ce que donne l’algorithme sur python :

def rendu\_monnaie (s\_due, s\_versee):  *#définir la fonction*

pieces = [1, 2, 5, 10, 20, 50, 100, 200] *#définir l’échelle Euro par une liste*

rendu = [] *#liste vide où vont paraitre les pièces de l’échelle qu’on va utiliser*

a\_rendre = s\_versee – s\_due *# la somme à rendre, ici on a vu c’était 48*

i = len(pieces) – 1 *#ici on veut commencer à la plus grande échelle donc*

*200. Alors on va définir l’indice de l’échelle à partir de la longueur de la liste.(-1 car l’indice commence à 0.)*

while a\_rendre > 0 : *#ensuite on arrive au point où le programme*

*commence à s’exécuter : tant que la somme à rendre ne sera pas totalement*

*rendue càd tant qu’elle ne sera pas nulle, on exécute les lignes à suivre*

if pieces[i] <= a\_rendre :

rendu.append(pieces[i]) *#si l’élément de l’échelle (200) est*

*inférieur à la somme à rendre, on ajoute cet élément à la liste* rendu *qu’on a créée au départ*

a\_rendre = a\_rendre - pieces[i] *#et on oublie pas de*

*soustraire la valeur de cet élément à la somme à rendre sinon le programme tournerait en boucle et serait*

else :  *sans fin*

i = i-1 *#si la condition n’est pas respectée, càd que l’élément*

*de l’échelle est plus grand que la somme à rendre , on passe à l’élément suivant donc on diminue l’indice de 1, donc on passe à l’indice de 100.*

return rendu #*enfin on retourne le rendu pour connaitre les pièces à*

*rendre*

Donc si on exécute ce programme avec les valeurs 452 et 500 par exemple, l’algorithme retournera cette liste : [20, 20, 5, 2, 1].

Pour conclure, on a vu que l’échelle de monnaie actuelle est très pratique en informatique comme dans d’autres domaines. C’est ce que l’on vient de démontrer à l’aide de l’algorithme glouton.

Merci de m’avoir écouté. Avez-vous des questions ?