Informatique - DS2

Proposition de corrigé

Problème

Présentation du problème

Une usine produit des **bobines** de film plastique thermo-rétractable pour l'emballage.
Chaque **commande** est constituée d'une liste de « laizes », à savoir les largeurs en cm de chaque bobine commandée.

Exemple : `[100,100,120,40,60,110,60,90,100]`.

(en réalité une commande contient des centaines de laizes)

La machine qui fabrique le film plastique délivre des **rouleaux** d'une longueur de 280 cm.
Chaque rouleau est ensuite tronçonné pour réaliser les bobines correspondant aux laizes d'une commande.
On cherche à caser une commande dans autant de rouleaux que nécessaire.
Par exemple, pour honorer la commande ci-dessus on peut utiliser quatre rouleaux :

- premier rouleau : `[100, 100, 40]` ;
- deuxième rouleau : `[120, 60, 60]` ;
- troisième rouleau : `[110, 90]` ;
- quatrième rouleau : `[100]`.

Sur les 280 cm d'un rouleau, la longueur non utilisée pour une commande est considérée comme une perte.

L'objectif de ce problème est d'automatiser la répartition des laizes d'une commande dans des rouleaux selon le procédé suivant qui vise à réduire les pertes :

- créer un rouleau en y entrant la plus grande laize de la commande, puis la plus grande laize restante possible, et ainsi de suite jusqu'à ce qu'aucune laize de la commande ne puisse enter dans le rouleau ;
- créer un nouveau rouleau et reproduire le même algorithme pour les laizes de la commande qui n'ont pas été déjà intégrées ;
- poursuivre ainsi la création de rouleaux jusqu'à ce que la commande soit entièrement casée.

Dans l'exemple donné on obtiendrait ainsi la répartition :

- premier rouleau : `[120, 110, 40]` ;
- deuxième rouleau : `[100, 100, 60]` ;
- troisième rouleau : `[100, 90, 60]`.

Ce qui économise un rouleau par rapport à la répartition précédente.

Structuration des données

Une **commande** sera modélisée par une liste de nombres.
Le remplissage d'un **rouleau** sera également modélisé par une liste de nombres (dont la somme est inférieure ou égale à 280).
La **répartition** d'une commande en rouleaux sera modélisée par une liste de listes de nombres.

Découpage du programme en fonctions python

Première fonction

Nom : `ordonner`
Paramètres d'entrée : la commande
Action : ranger les laizes de la commande dans l'ordre décroissant.
Données renvoyées : la commande rangée

Deuxième fonction

Nom : `rouleau`
Paramètres d'entrée : une liste de laizes dans l'ordre décroissant
Action : remplir le rouleau avec les laizes de la liste selon le procédé décrit dans la présentation du problème.
Données renvoyées : la liste des laizes entrées dans le rouleau, la liste des laizes restantes.

Troisième fonction

Nom : `rouleaux`
Paramètres d'entrée : la commande
Action : répartir la commande selon le procédé décrit dans la présentation du problème.
Données renvoyées : la liste des rouleaux.

Questions

1. Ci-dessous une proposition de listing python pour implémenter la première fonction :

```
1 def ordonner(C):
      fini = False
3     while fini==False:
          fini=True
5         for k in range(len(C)-1):
              if C[k]<C[k+1]:
7                 C[k],C[k+1]=C[k+1],C[k]
                  fini=False
9     return(C)
```

Décrire le fonctionnement du programme en commentant chaque ligne.
(on pourra écrire sur le sujet, ou utiliser les numéros de ligne.)

On reconnaît un algorithme de tri à bulle.

Ligne 1 : `def ordonner(C):` :
Définit la fonction en lui donnant un nom (`ordonner`) et en nommant son unique paramètre d'entrée : `C` ;

Ligne 2 : `fini = False` :
Affecte la variable booléenne `fini` à la valeur `False` ;

Ligne 3 : `while fini==False:` :
Lance une boucle conditionnelle réalisée tant que la variable fini est à la valeur False

Ligne 4 : `fini = True` :
Affecte (temporairement) la variable booléenne `fini` à la valeur `True` ;

Ligne 5 : `for k in range(len(C)-1):` :
Lance un boucle séquentielle qui sera réalisée pour toute les valeurs de l'entier k entre 0 et le nombre d'éléments de la liste `C` moins 1 ;

Ligne 6 : `if C[k]<C[k+1]:` :
Teste la condition : « l'élément de la liste `C` de rang `k` est strictement inférieur à son suivant ;

Ligne 7 : `C[k],C[k+1]=C[k+1],C[k]` :
Permute les deux éléments de rang `k` et `k+1` ;

Ligne 8 : `fini = False` :
Parce qu'il y a eu un changement d'ordre dans la liste, on considère que le tri n'est peut-être pas fini ;

Ligne 9 : `return(C)`
Si on est sorti de la boucle conditionnelle, c'est parce qu'aucun élément de la liste `C` n'est dans le mauvais ordre par rapport à son suivant, donc que la liste `C` est rangée dans l'ordre décroissant.
La fonction renvoie la liste rangée dans l'ordre décroissant.

2. Proposer un listing python pour implémenter la deuxième fonction.
Quelques rappels d'instructions :
créer une liste vide : `liste=[]`
ajouter un élément à la fin d'une liste : `liste.append(élément)`

Proposition de listing :

```
def rouleau(C):
    # C comme commande
    L=280 # L comme longueur du rouleau
    rouleau=[] # crÃ©e un rouleau vide
    R=[] # R comme restant
    for l in C: # l comme laize
        if l <= L: # il reste assez de longueur de rouleau
            rouleau.append(l) # on rÃ©alise la laize dans ce
   rouleau
            L=L-l # on raccourcit le rouleau
        else:
            R.append(l) # on passe la laize dans la liste non
    utilisÃ©e
    return(rouleau,R) # on renvoie le rouleau rÃ©alisÃ© et
   les laizes non utilisÃ©es
```

3. Proposer un listing python pour implémenter la troisième fonction.
Quelques rappels d'instructions :
affecter à deux variables deux données renvoyées par une fonction :
`variable1,variable2=fonction(paramètres)`

Proposition de listing :

```
def rouleaux(C):   # C comme commande
    C=ordonner(C) # ordonne les bobines
    rouleaux=[] # crÃ©e une liste vide de rouleaux
    while C!=[]: # tant qu'il reste des laizes Ã  caser
        nouveau_rouleau,C=rouleau(C)
        # affecte Ã  nouveau_rouleau et C les deux donnÃ©es
        # retournÃ©es par la fonction rouleau
        rouleaux.append(nouveau_rouleau)
        # ajoute le nouveau rouleau dans la liste des
   rouleaux
    return(rouleaux) # renvoie la liste des rouleaux
```