Quelques algorithmes importants sur les listes

Algorithmes Gloutons :

Le rendue de Monnaie :

Exercice 1.

1. $(20, 10, 5, 1)$
2. $(5, 5, 5, 5, 5, 5, 5, 1)$
3. $(20, 10, 5, 1)$ et $(20, 10, 2, 2, 2)$
4. Rendu impossible.
5. (a)
```
somme=50-14
ListeMontants=[20,10,5,2,1]
ListeNbPieces=[0,0,0,0,0]
for k in range(len(ListeMontants)) :
        ListeNbPieces[k]=somme//ListeMontants[k]
        somme=somme%ListeMontants[k]
print(ListeNbPieces)
```
   (b)
```
somme=50-14
ListeMontants=[5,2,1]
ListeNbPieces=[0,0,0]
for k in range(len(ListeMontants)) :
        ListeNbPieces[k]=somme//ListeMontants[k]
        somme=somme%ListeMontants[k]
print(ListeNbPieces)
```
   (c) On considère maintenant le script suivant :
```
def monnaie(somme,ListeMontants) :
    ListeNbPieces=[0 for x in ListeMontants]
    for k in range(len(ListeMontants)) :
        ListeNbPieces[k]=somme//ListeMontants[k]
    somme=somme % ListeMontants[k]
return somme,ListeNbPiece
```
6. $T = [18, 7, 1]$.
   - ⋆ Avec l'algorithme précédent $(18, 1, 1, 1)$.
   - ⋆ La solution $(7, 7, 7)$ nécessite moins de pièces.

Allocation de plages horaires :

Exercice 2.

1.
```
tableau_horaires=[[0,7,'C1'],[2,5,'C2'],[6,8,'C3'],[1,2,'C4'],[5,6,'C5'],[0,2,'C6'],
[4,7,'C7'],[0,1,'C8'],[3,6,'C9'],[1,3,'C10'],[4,5,'C11'],[6,8,'C12'],[0,2,'C13'],
[5,7,'C14'],[1,4,'C15']]
print(sorted(tableau_horaires,key=lambda fin: fin[1]))
```

2.
```
def planning(tab_horaires):
    nb_intervalles=len(tab_horaires)
    tab_horaires=(sorted(tab_horaires,key=lambda fin: fin[1]))
    tableau_planning=[tab_horaires[0]]
            j=0
            for i in range(1,nb_intervalles):
        if tab_intervalles[i][0]>=tab_intervalles[j][1]:
            tab_planning.append(tab_intervalles[i])
                    j=i
    return tab_planning
```

3. Le script précédent établit une liste d'horaires de conférences entre 8h et 17h d'une durée maximale chacune de 3h.