

## Corrigé du TP Informatique 12

### Exercice 1

1. On saisit :

```
def rendu(v,S):
    """rendu(v:int)->list
    v : valeur à rendre
    en sortie : la liste des pièces ou billets
    """
    res=[]
    reste=v
    ind=0
    while reste>0:
        if S[ind]>reste:
            ind+=1
        else:
            res.append(S[ind])
            reste-=S[ind]
    return res
```

2. Le système de la zone euro est dit canonique : l'algorithme glouton proposé donne la solution optimale. Mais un système pris au hasard n'a pas de raison de l'être.
- Si l'on considère le système proposé dans la question 2, pour rendre 6, l'algorithme glouton renvoie  $6 = 4 + 1 + 1$  alors que la solution optimale est  $3 + 3$ .

### Exercice 2

1. Un programme possible :

```
def trieff(L) :
    return sorted(L,key=lambda couple : couple[0]/couple[1],reverse=True)
```

2. Un programme possible :

```
def remplissage(L,P) :
    res,w,v=[],0,0
    liste=trieff(L)
    for i in range(len(liste)) :
        if w+liste[i][1]<=P :
            res.append(liste[i])
            w=w+liste[i][1]
            v=v+liste[i][0]
    return res,w,v
```

3. L'instruction `remplissage(L,15)` renvoie

```
Out[11]: ([[10, 9], [1, 2]], 11, 11)
```

Mais cette solution n'est pas optimale, on peut remplir le sac avec les objets (2, 5), (10, 9) pour un poids total de  $14 \leq 15$  et une valeur de  $12 > 11$ .

### Exercice 3

Un programme utilisant une boucle `for` comme pour le sac à dos.

```
def Zeckendorf(n):
    # construction des termes de la suite jusqu'à F_k <= n < F_{k+1}
    fibo=[0,1]
    while fibo[-1]<=n :
        fibo.append(fibo[-1]+fibo[-2])
    # stratégie gloutonne en redescendant la liste
    res,m=[],0
    for i in range(len(fibo)-2,0,-1) :
        if m+fibo[i]<=n :
            res=[fibo[i]]+res
            m=m+fibo[i]
    # test de validité du résultat
    test=sum(res)==n
    return res,test
```

Un programme utilisant une boucle `while` comme pour le rendu de monnaie.

```
def Zeckendorf(n) :
    # construction des termes de la suite jusqu'à F_k <= n < F_{k+1}
    fibo=[0,1]
    while fibo[-1]<=n :
        fibo.append(fibo[-1]+fibo[-2])
    # stratégie gloutonne en redescendant la liste
    res,m,indice=[fibo[-2]],n-fibo[-2],-2
    while m>0 :
        while fibo[indice]>m :
            indice-=1
        res=[fibo[indice]]+res
        m-=fibo[indice]
    # test de validité du résultat
    test=sum(res)==n
    return res,test
```