```
#%%
##
"""
```

**RENDU DE MONNAIE AVEC CAISSE ILLIMITEE**

```
"""
euros1 = [50, 20, 10, 5, 2, 1]
euros2 = [50, 20, 10, 5, 2, 1, 0.50, 0.20, 0.10, 0.05, 0.02, 0.01]

#%%
##
'''
```

**VERSION AVEC MONTANTS SUPERIEURS A 1 EURO**

```
'''

def rend(montant, caisse):
    from copy import deepcopy
    utiliser = deepcopy(caisse)
    rendu = []
    reste = montant
    while reste:    # car bool([]) = F
        k = 0
        while reste < utiliser[k]:
            k += 1
        q = reste // utiliser[k]
        rendu.append([utiliser[k],q])
        reste = reste - utiliser[k] * q
        del(utiliser[:k])
    return rendu

print(rend(34, euros1))
```

```
#%%
##
‘’’
```

**VERSION AVEC LES MONTANTS INFERIEURS A 1 EURO**

Pour continuer à utiliser la division euclidienne, on convertit le montant

en entier à l'entrée, et on le reconvertit à la sortie.

On évite ainsi les problèmes du calcul flottant, approché par nature,

ce qui peut fausser les tests d'égalité.

```
‘’’

def rend2(montant, caisse):
    from copy import deepcopy
    rendu = []
    reste = int(montant * 100)
    utiliser = [int(100 * k) for k in caisse]
    while reste:
        k = 0
        while reste < utiliser[k]:
            k += 1
        q = reste // utiliser[k]
        rendu.append([utiliser[k],q])
        reste = reste - utiliser[k] * q
        del(utiliser[:k])
    for k in rendu:
        if k[0] % 100 == 0:
            k[0] = k[0]//100
        else:
            k[0] = round(k[0]/100,2)
    return rendu

print(rend2(34.17, euros2))
```

```
#%%
##
"""
```
**VOYAGEUR DE COMMERCE**
```
"""
villes5 = ['Clermont', 'Riom', 'Roanne', 'Thiers', 'Vichy']

distances5 = [[0, 15, 104, 42, 54], [15, 0, 108, 44, 39],
          [104, 108, 0, 59, 70], [42, 44, 59, 0, 35], [54,
39, 70, 35, 0]]
#%%
##
def circuit(vil, tab, n0):
    etapes = [vil[n0]] ; D = 0
    reste = [ k for k in range(len(vil)) if k != n0 ]
    dep = n0 # départ de l'étape courante
    while reste:
        i = reste[0] ; d = tab[i][dep]
        for k in reste:
            if tab[k][dep] < d:
                d = tab[k][dep]
                i = k
        etapes.append(vil[i])
        D += d
        dep = i
        reste.remove(i)
    etapes.append(vil[n0])
    D += tab[i][n0]
    return etapes, D
#%%
##
for i in range(5):
    print(circuit(villes5, distances5 , i))
```

```
#%%
##
''' FORCE BRUTE
Pour N villes : (N-1)! trajets donc (N2-1)! / (N1-1)!
quand on passe de N1 villes à N2.
N1 = 5 --> N2 = 10 : 9x8x7x6x5 = 15120 fois plus de
temps, donc 0,4 s.
N1 = 5 --> N2 = 20 : 19x18x...x6x5 = 5e15 fois plus de
temps, donc 1,27e11 s soit environ 4000 ans.
'''
#%%
## FORCE BRUTE pour 5 villes
def brute5(vil, tab, n0):
    v0 = n0 ; LD = [] ; LC = []
    reste1 = [ k for k in range(len(vil)) if k != v0 ]
    for v1 in reste1:
        D1 = tab[v0][v1]
        reste2 = [ k for k in reste1 if k != v1 ]
        for v2 in reste2:
            D2 = tab[v1][v2]
            reste3 = [ k for k in reste2 if k != v2 ]
            for v3 in reste3:
                D3 = tab[v2][v3]
                reste4 = [ k for k in reste3 if k != v3 ]
                v4 = reste4[0] ; D4 = tab[v3][v4]
                D5 = tab[v4][v0] ; D = D1 + D2 + D3 + D4 + D5
                LD.append(D)
                LD = list(set(LD)) # éliminer les doublons
                LD.sort()
    return LD, len(LD)

for i in range(5):
    print(brute5(villes5, distances5 , i))
```

''' **Bilan** : sur 12 longueurs de circuits possibles, de moyenne 288 km,

- au départ de Clermont, la distance trouvée est la 3e plus courte ;
- au départ de Riom, la distance trouvée est la 5e plus courte ;
- au départ de Roanne, la distance trouvée est la 3e plus courte ;
- au départ de Thiers, la distance trouvée est la 3e plus courte ;
- au départ de Vichy, la distance trouvée est la 5e plus courte.

L'AGLOU ne trouve donc jamais l'optimum dans ce contexte, mais donne des résultats meilleurs que la moyenne et la médiane de tous les cas possibles.

**Pour les 6 villes du sud**, parcours moins simple, l'AGLOU trouve l'optimum dans un cas mais une solution médiocre dans un autre : 35$^{e}$ plus courte sur 49 possibilités ; pour les cas restants, c'est mieux que la moyenne et la médiane.