Python
Problème de recherche : la caisse automatique

Une entreprise programme le logiciel d'une caisse automatique. La caisse dispose des pièces et billets suivants (en centimes) : 500, 200, 100, 50, 20, 10, 5, 2 et 1. Elle doit rendre la monnaie **avec le moins de pièces possible**.
Le développeur propose la méthode suivante, dite « gloutonne » : *à chaque étape, on rend la plus grande pièce possible qui ne dépasse pas la somme restant à rendre.*

La mission

1. Écrire en Python une fonction `rendu(somme)` qui renvoie la liste des pièces rendues par la méthode gloutonne.
2. Un pays imaginaire utilise seulement des pièces de 1, 3 et 4 centimes. La méthode gloutonne rend-elle toujours le moins de pièces possible ? Trouver la plus petite somme pour laquelle elle se trompe.

Pour aller plus loin

Même question pour un système de pièces de 1, 5 et 12 centimes.

Coups de pouce

Bloqué ? Des indices progressifs sont disponibles en ligne. N'en ouvrir qu'un à la fois, et seulement après avoir cherché !

Éléments de correction

1. La fonction

```
def rendu(somme, pieces=[500, 200, 100, 50, 20, 10, 5, 2, 1]):
    liste = []
    for p in pieces:          # des plus grandes aux plus petites
        while somme >= p:
            liste.append(p)
            somme = somme - p
    return liste

print(rendu(387))   # [200, 100, 50, 20, 10, 5, 2]
```

Avec les pièces en euros, on peut vérifier (par exemple en testant toutes les sommes jusqu'à 1 000 centimes avec un programme qui cherche le minimum) que la méthode gloutonne donne toujours le nombre minimal de pièces.

2. Pièces de 1, 3 et 4

`rendu(6, [4, 3, 1])` renvoie `[4, 1, 1]` : 3 pièces. Or $6 = 3+3$: 2 pièces suffisent. On vérifie que pour 1, 2, 3, 4 et 5 la méthode gloutonne est optimale ($5 = 4+1$). **La plus petite somme où elle se trompe est 6.**

Conclusion : la méthode gloutonne est rapide et simple, mais elle n'est optimale que pour certains systèmes de pièces (dont l'euro).

Pour aller plus loin

Avec 1, 5 et 12 : pour 15, la méthode gloutonne rend $12+1+1+1$ (4 pièces) alors que $5+5+5$ en utilise 3. On vérifie que les sommes de 1 à 14 sont bien traitées : **la plus petite somme problématique est 15**.

Ce qu'on a utilisé du chapitre

- Fonctions, listes, boucles `for` et `while`, tests.
- Tester un programme sur de nombreux cas ; chercher un contre-exemple.

Pour aller plus loin (enrichissement)

Pour trouver à coup sûr le nombre minimal de pièces dans n'importe quel système, on utilise la *programmation dynamique*, étudiée en NSI et dans le supérieur.