BACCALAURÉAT

SESSION 2025

Épreuve de l'enseignement de spécialité

NUMÉRIQUE et SCIENCES INFORMATIQUES

Partie pratique

Classe Terminale de la voie générale

Sujet n°46

DURÉE DE L'ÉPREUVE : 1 heure

Le sujet comporte 3 pages numérotées de 1 / 3 à 3 / 3
Dès que le sujet vous est remis, assurez-vous qu'il est complet.

Le candidat doit traiter les 2 exercices.

EXERCICE 1 (10 points)

Écrire une fonction `compte_occurrences` prenant en paramètres une valeur `x` et un tableau `tab` (de type `list`) et renvoyant le nombre d'occurrences de `x` dans `tab`.

L'objectif de cet exercice étant de parcourir un tableau, il est interdit d'utiliser la méthode `count` des listes Python.

Exemples :

```
>>> compte_occurrences(5, [])
0
>>> compte_occurrences(5, [-2, 3, 1, 5, 3, 7, 4])
1
>>> compte_occurrences('a', ['a','b','c','a','d','e','a'])
3
```

EXERCICE 2 (10 points)

On considère dans cet exercice un algorithme glouton pour le rendu de monnaie. Pour rendre une somme en monnaie, on utilise à chaque fois la plus grosse pièce possible et ainsi de suite jusqu'à ce que la somme restante à rendre soit nulle.

Les pièces de monnaie utilisées sont :

```
pieces = [1, 2, 5, 10, 20, 50, 100, 200]
```

On souhaite écrire une fonction `rendu_monnaie` qui prend en paramètres

- un entier `somme_due` représentant la somme à payer ;
- un entier `somme_versee` représentant la somme versée qui est supérieure ou égale à `somme_due` ;
- et qui renvoie un tableau de type `list` contenant les pièces qui composent le rendu de la monnaie restante, c'est-à-dire de `somme_versee - somme_due`.

Ainsi, l'instruction `rendu_monnaie(452, 500)` renvoie le tableau `[20, 20, 5, 2, 1]`.

En effet, la somme à rendre est de 48 euros soit `20 + 20 + 5 + 2 + 1`.

Le code de la fonction `rendu_monnaie` est donné ci-dessous :

```
def rendu_monnaie(somme_due, somme_versee):
    '''Renvoie la liste des pièces à rendre pour rendre la monnaie
    lorsqu'on doit rendre somme_versee - somme_due'''
    rendu = ...
    a_rendre = ...
    i = len(pieces) - 1
    while a_rendre > ...:
        while pieces[i] > a_rendre:
            i = i - 1
        rendu.append(...)
        a_rendre = ...
    return rendu
```

Compléter ce code et le tester :

```
>>> rendu_monnaie(700, 700)
[]
>>> rendu_monnaie(102, 500)
[200, 100, 50, 20, 20, 5, 2, 1]
```