BACCALAUREAT

SESSION 2022

Épreuve de l'enseignement de spécialité

NUMERIQUE et SCIENCES INFORMATIQUES

Partie pratique

Classe Terminale de la voie générale

Sujet n°31

DUREE DE L'EPREUVE : 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 (4 points)

Écrire en langage Python une fonction `recherche` prenant comme paramètres une variable `a` de type numérique (`float` ou `int`) et un tableau `t` (type `list`) et qui renvoie le nombre d'occurrences de `a` dans `t`.

Exemples d'utilisations de la fonction `recherche` :

```
>>> recherche(5,[])
0
>>> recherche(5,[-2, 3, 4, 8])
0
>>> recherche(5,[-2, 3, 1, 5, 3, 7, 4])
1
>>> recherche(5,[-2, 5, 3, 5, 4, 5])
3
```

EXERCICE 2 (4 points)

La fonction `rendu_monnaie_centimes` prend en paramètres deux nombres entiers positifs `s_due` et `s_versee` et elle permet de procéder au rendu de monnaie de la différence `s_versee - s_due` pour des achats effectués avec le système de pièces de la zone Euro. On utilise pour cela un algorithme qui commence par rendre le maximum de pièces de plus grandes valeurs et ainsi de suite. La fonction renvoie la liste des pièces qui composent le rendu.

Toutes les sommes sont exprimées en centimes d'euros. Les valeurs possibles pour les pièces sont donc `[1, 2, 5, 10, 20, 50, 100, 200]`.

Ainsi, l'instruction `rendu_monnaie_centimes(452, 500)` renverra la liste `[20, 20, 5, 2, 1]`.
En effet, la somme à rendre est de 48 centimes soit 20 + 20 + 5 + 2 + 1.

Le code de la fonction est donné ci-dessous :

```
def rendu_monnaie_centimes(s_due, s_versee):
    pieces = [1, 2, 5, 10, 20, 50, 100, 200]
    rendu = ...
    a_rendre = ...
    i = len(pieces) - 1
    while a_rendre > ... :
        if pieces[i] <= a_rendre :
            rendu.append(...)
            a_rendre = ...
        else :
            i = ...
    return rendu
```

Compléter ce code pour qu'il donne :

```
>>> rendu_monnaie_centimes(700,700)
[]
>>> rendu_monnaie_centimes(112,500)
[200, 100, 50, 20, 10, 5, 2, 1]
```

```python
"""
Author: Pascal Padilla
Source: correction de l'exercice 1 du sujet 31 des épreuves pratiques NSI 2022

Remarques:
"""

def recherche(element, tab):
    """
    Tests et exemples:
    >>> recherche(5,[])
    0
    >>> recherche(5,[-2, 3, 4, 8])
    0
    >>> recherche(5,[-2, 3, 1, 5, 3, 7, 4])
    1
    >>> recherche(5,[-2, 5, 3, 5, 4, 5])
    3
    """
    # BOUCLE
    # -> invariant: occurrence contient le nb d'apparition
    #               de element dans la zone déjà parcourue de tab
    # -> initialisation: au départ, rien n'est parcouru
    occurrence = 0

    # -> condition d'arrêt: après la dernière case du tableau
    for elt_courant in tab:
        # maintien de l'invariant si la valeur element est trouvée
        if elt_courant == element:
            # incrémenter le compteur `occurrence`
            occurrence = occurrence + 1

    # fin de BOUCLE
    # tout le tableau est parcouru et occurrence est correct
    return occurrence


# Vérification par des assertions
assert recherche(5,[]) == 0
assert recherche(5,[-2, 3, 4, 8]) == 0
assert recherche(5,[-2, 3, 1, 5, 3, 7, 4]) == 1
assert recherche(5,[-2, 5, 3, 5, 4, 5]) == 3

# Vérification par des affichages
print(recherche(5,[]))
print(recherche(5,[-2, 3, 4, 8]))
print(recherche(5,[-2, 3, 1, 5, 3, 7, 4]))
print(recherche(5,[-2, 5, 3, 5, 4, 5]))

# Vérification par doctest
from doctest import testmod
testmod()
```

```
"""
Author: Pascal Padilla
Source: correction de l'exercice 1 du sujet 31 des épreuves pratiques NSI 2022

Remarques:
    * algo glouton classique:
        un algorithme glouton est un algorithme qui fait un choix
        glouton à chaque tour de boucle.
        Ce choix peut éviter pleins de calculs.
        Ici, le choix glouton c'est de rendre, quand c'est possible
        la pièce la plus grande. Et si ne ce n'est pas possible,
        de rendre la pièce d'un montant juste inférieur.
"""


def rendu_monnaie_centimes(s_due, s_versee):
    pieces = [1, 2, 5, 10, 20, 50, 100, 200]
    # -> initialisation: tableau vide (rien à rendre)
    rendu = []

    # -> initialisation de la somme à rendre
    a_rendre = s_versee - s_due

    # -> invariant: i est l'indice de la piece qu'on essaye de rendre
    i = len(pieces) - 1

    # -> condition d'arrêt: il n'y a plus rien à rendre
    while a_rendre > 0 :
        # il est possible de rendre une piece courante piece[i]
        if pieces[i] <= a_rendre :
            # ajout de la pièce dans le tableau de l'argent rendu
            rendu.append(pieces[i])
            # mise à jour de la somme à rendre
            a_rendre = a_rendre - pieces[i]

        # la pièce courante est plus grande que la somme à rendre
        # on passe donc à une pièce d'un montant juste inférieur
        else :
            i = i - 1

    # fin BOUCLE
    # il n'y a plus rien à rendre
    # et rendu contient les pièces rendues
    return rendu



# Vérification par des assertions
assert rendu_monnaie_centimes(452, 500) == [20, 20, 5, 2, 1]
assert rendu_monnaie_centimes(700,700) == []
assert rendu_monnaie_centimes(112,500) == [200, 100, 50, 20, 10, 5, 2, 1]

# Vérification par des affichages
print(rendu_monnaie_centimes(452, 500))
print(rendu_monnaie_centimes(700,700))
print(rendu_monnaie_centimes(112,500))
```