BACCALAUREAT

SESSION 2022

Épreuve de l'enseignement de spécialité

NUMERIQUE et SCIENCES INFORMATIQUES

Partie pratique

Classe Terminale de la voie générale

Sujet n°13

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)

On s'intéresse au problème du rendu de monnaie. On suppose qu'on dispose d'un nombre infini de billets de 5 euros, de pièces de 2 euros et de pièces de 1 euro.

Le but est d'écrire une fonction nommée `rendu` dont le paramètre est un entier positif non nul `somme_a_rendre` et qui retourne une liste de trois entiers `n1`, `n2` et `n3` qui correspondent aux nombres de billets de 5 euros (`n1`) de pièces de 2 euros (`n2`) et de pièces de 1 euro (`n3`) à rendre afin que le total rendu soit égal à `somme_a_rendre`.

On utilisera un algorithme glouton : on commencera par rendre le nombre maximal de billets de 5 euros, puis celui des pièces de 2 euros et enfin celui des pièces de 1 euros.

Exemples :

```
>>> rendu(13)
[2,1,1]
>>> rendu(64)
[12,2,0]
>>> rendu(89)
[17,2,0]
```

EXERCICE 2 (4 points)

On veut écrire une classe pour gérer une file à l'aide d'une liste chaînée. On dispose d'une classe `Maillon` permettant la création d'un maillon de la chaîne, celui-ci étant constitué d'une valeur et d'une référence au maillon suivant de la chaîne :

```
class Maillon :
    def __init__(self,v) :
        self.valeur = v
        self.suivant = None
```

Compléter la classe File suivante où l'attribut `dernier_file` contient le maillon correspondant à l'élément arrivé en dernier dans la file :

```
class File :
    def __init__(self) :
        self.dernier_file = None

    def enfile(self,element) :
        nouveau_maillon = Maillon(...)
        nouveau_maillon.suivant = self.dernier_file
        self.dernier_file = ...

    def est_vide(self) :
```

```
        return self.dernier_file == None

    def affiche(self) :
        maillon = self.dernier_file
        while maillon != ... :
            print(maillon.valeur)
            maillon = ...

    def defile(self) :
        if not self.est_vide() :
            if self.dernier_file.suivant == None :
                resultat = self.dernier_file.valeur
                self.dernier_file = None
                return resultat
            maillon = ...
            while maillon.suivant.suivant != None :
                maillon = maillon.suivant
            resultat = ...
            maillon.suivant = None
            return resultat
        return None
```

On pourra tester le fonctionnement de la classe en utilisant les commandes suivantes dans la console Python :

```
>>> F = File()
>>> F.est_vide()
True
>>> F.enfile(2)
>>> F.affiche()
2
>>> F.est_vide()
False
>>> F.enfile(5)
>>> F.enfile(7)
>>> F.affiche()
7
5
2
>>> F.defile()
2
>>> F.defile()
5
>>> F.affiche()
7
```

```python
"""
Author: Pascal Padilla
Source: correction de l'exercice 1 du sujet 13 des épreuves pratiques NSI 2022
"""

from doctest import testmod


def rendu(somme_a_rendre):
    """Rendu de monnaie sous forme de tableau contenant
    le nombre de billets de 5¿, de pièces de 2¿ et de 1¿

    Args:
        somme_a_rendre (int): valeur dont on doit faire la monnaie

    Returns:
        list: tableau de 3 cases contenant :
                * le nb de billets de 5¿
                * le nb de pièces de 2¿
                * le nb de pièces de 1¿

    Tests et exemples:
    >>> rendu(13)
    [2, 1, 1]
    >>> rendu(64)
    [12, 2, 0]
    >>> rendu(89)
    [17, 2, 0]
    """
    ########################
    # Première méthode     #
    #   avec trois boucles #
    ########################

    # somme d'argent qu'il reste à rendre
    reste = somme_a_rendre

    # BOUCLE
    # invariant de boucle:
    #   nb_5_euros × 5 + reste = somme_a_rendre
    # condition d'arrêt:
    #   reste < 5
    nb_5_euros = 0
    while not reste < 5:
        nb_5_euros += 1
        reste -= 5

    # BOUCLE
    # invariant de boucle:
    #   nb_2_euros × 2 + nb_5_euros × 5 + reste = somme_a_rendre
    # condition d'arrêt:
    #   reste < 2
    nb_2_euros = 0
    while not reste < 2:
        nb_2_euros += 1
        reste -= 2

    # la somme qui reste est égale au nb de pièces de 1¿ ;)
    nb_1_euros = reste

    # return [nb_5_euros, nb_2_euros, nb_1_euros]

    ###################################
    # Deuxième méthode                #
    #   avec 3 divisions euclidiennes #
    ###################################
    reste = somme_a_rendre

    # le nombre de billets de 5 ¿ est égal au quotient de la
    #    division euclidienne de ce qui reste à rendre par 5
    nb_5_euros = reste // 5

    # le reste de la monnaie à rendre est égal
```

```
    #    au reste de la division euclidienne
    reste = reste % 5

    # idem avec le nb de pièce de 2 euros
    nb_2_euros = reste // 2
    reste = reste % 2

    # la somme qui reste est égale au nb de pièces de 1¿ ;)
    nb_1_euros = reste

    return [nb_5_euros, nb_2_euros, nb_1_euros]


# tests de l'énoncé
testmod()
```

```
"""
Author: Pascal Padilla
Source: correction de l'exercice 2 du sujet 13 des épreuves pratiques NSI 2022

Remarques:
Je ne veux pas vous embrouiller, mais pour mieux comprendre
cet exercice, je me suis forcé à remplacer tous les
'suivant' du code par des 'precedent'.
En effet, dans ma tête en faisant monMaillon.suivant
je me dis que je vais vers la fin de la file : vers son dernier élément.
Et en faisant monMaillon.precedent, j'ai l'impression de remonter
la file et de me diriger vers son premier élément.
...
Par exemple, avec les notations officielle de l'énoncé,
on remonte la file du dernier maillon vers le premier à l'aide
d'une boucle qui calcule 'maillon.suivant'.
Personnellement, je remonte une file du dernier vers le premier
en faisant plutôt une boucle 'maillon.precedent'
...
Bon, ok, je chipote mais je n'arrivais pas à comprendre tant que
je n'avais pas réalisé cela... Et pour cause : c'est juste le contraire !
"""

class Maillon:
    """
    Maillon composé d'une valeur et d'un lien vers son
    maillon suivant.
    """
    def __init__(self, v):
        self.valeur = v
        self.suivant = None

class File:
    """ File de maillons possédant comme attribut un lien
    vers son dernier maillon (le dernier maillon arrivé)
    """
    def __init__(self):
        self.dernier_file = None

    def enfile(self, element):
        # création d'un nouveau maillon qui deviendra le dernier
        # ce maillon contient la valeur element
        nouveau_maillon = Maillon(element)

        # on définit le maillon suivant du nouveau
        # maillon comme étant le dernier maillon de la file
        # ceci permet "d'accrocher" le nouveau maillon à la file
        nouveau_maillon.suivant = self.dernier_file

        # on définit le nouveau maillon comme étant le dernier
        self.dernier_file = nouveau_maillon

    def est_vide(self):
        return self.dernier_file == None

    def affiche(self):
        """ Affiche toutes les valeurs des maillons de
        la file en commençant par le dernier arrivé
        """
        maillon = self.dernier_file

        # condition d'arrêt de la boucle :
        #   il n'y a plus de maillon à afficher
        while maillon != None:
            print(maillon.valeur)
            # on remonte dans la file
            maillon = maillon.suivant

    def defile(self):
        if not self.est_vide():
            # cas du dernier maillon qui n'a pas de suivant
            # la file ne contient donc qu'un seul élément
            if self.dernier_file.suivant == None:
```

```
                # enregistrer la valeur du dernier maillon...
                resultat = self.dernier_file.valeur
                # ...avant de l'effacer
                # la file est donc maintenant vide
                self.dernier_file = None
                # renvoyer le résultat enregistré
                return resultat

            # on va remonter dans la file en commençant par le dernier
            # maillon (dont on est certain qu'il possède un suivant)
            maillon = self.dernier_file

            # remonter dans la file de maillon en maillon
            # jusqu'à atteindre le deuxième maillon de la file
            # (c'est à dire le maillon dont son suivant est le premier)
            # (c'est à dire le maillon dont le suivant de son suivant est vide)
            # (et oui !)
            while maillon.suivant.suivant != None:
                maillon = maillon.suivant

            # ok : maillon contient le deuxième élément


            # le résultat à renvoyer est donc la valeur de
            # son maillon suivant
            # on enregistre ce résultat
            resultat = maillon.suivant.valeur

            # puis on efface la connection entre le premier
            # maillon et le deuxième
            # ceci permet de "décrocher" le premier maillon
            # et du coup, le deuxième maillon devient le nouveau premier maillon
            maillon.suivant = None

            # on renvoie la valeur enregistrée qui est bien celle de
            # l'ancien premier maillon
            return resultat

        # cas de la file vide
        # dans ce cas, on renvoie le vide
        return None


# tests de l'énoncé
F = File()
print( F.est_vide() )

F.enfile(2)
F.affiche()
print( F.est_vide() )

F.enfile(5)
F.enfile(7)
F.affiche()

print( F.defile() )
print( F.defile() )
F.affiche()
```