BACCALAURÉAT

SESSION 2023

Épreuve de l'enseignement de spécialité

NUMÉRIQUE et SCIENCES INFORMATIQUES

Partie pratique

Classe Terminale de la voie générale

Sujet n°40

DURÉE DE L'ÉPREUVE : 1 heure

Le sujet comporte 4 pages numérotées de 1 / 4 à 4 / 4
Dès que le sujet vous est remis, assurez-vous qu'il est complet.

Le candidat doit traiter les 2 exercices.

Exercice 1 (4 points)

Pour cet exercice :

- On appelle « mot » une chaîne de caractères composée de caractères choisis parmi les 26 lettres minuscules ou majuscules de l'alphabet,
- On appelle « phrase » une chaîne de caractères :
  - composée d'un ou de plusieurs « mots » séparés entre eux par un seul caractère espace `' '`,
  - se finissant :
    - soit par un point `'.'` qui est alors collé au dernier mot,
    - soit par un point d'exclamation `'!'` ou d'interrogation `'?'` qui est alors séparé du dernier mot par un seul caractère espace `' '`.

Voici deux exemples de phrases :

- `'Cet exercice est simple.'`
- `'Le point d exclamation est separe !'`

Après avoir remarqué le lien entre le nombre de mots et le nombres de caractères espace dans une phrase, programmer une fonction `nombre_de_mots` qui prend en paramètre une phrase et renvoie le nombre de mots présents dans celle-ci.

Exemples :

```
>>> nombre_de_mots('Cet exercice est simple.')
4

>>> nombre_de_mots('Le point d exclamation est separe !')
6

>>> nombre_de_mots('Combien de mots y a t il dans cette
phrase ?')
10

>>> nombre_de_mots('Fin.')
1
```

Exercice 2 (4 points)

La classe `Noeud` ci-dessous permet d'implémenter une structure d'arbre binaire de recherche (ABR). On considère que la clé d'un nœud est un entier et qu'elle est unique dans l'ABR.

```
class Noeud:
    def __init__(self, valeur):
        '''Méthode constructeur pour la classe Noeud.
        Paramètre d'entrée : valeur (int)'''
        self.valeur = valeur
        self.gauche = None
        self.droit = None

    def getValeur(self):
        '''Méthode accesseur pour obtenir la valeur du noeud
        Aucun paramètre en entrée'''
        return self.valeur

    def droitExiste(self):
        '''Méthode renvoyant True si le sous-arbre droit est non
vide
        Aucun paramètre en entrée'''
        return (self.droit is not None)

    def gaucheExiste(self):
        '''Méthode renvoyant True si le sous-arbre gauche est non
vide
        Aucun paramètre en entrée'''
        return (self.gauche is not None)

    def inserer(self, cle):
        '''Méthode d'insertion de clé dans un arbre binaire de
recherche
        Paramètre d'entrée : cle (int)'''
        if cle < ... :
            # on insère à gauche
            if self.gaucheExiste():
                # on descend à gauche et on recommence le test
initial
                 ...
            else:
                # on crée un fils gauche
                self.gauche =  ...
        elif cle >  ... :
            # on insère à droite
            if  ... :
                # on descend à droite et on recommence le test
initial
                 ...
            else:
                # on crée un fils droit
                 ...  = Noeud(cle)
```

Compléter la fonction récursive `inserer` afin qu'elle permette d'insérer un nœud dans l'arbre binaire de recherche proposé, à l'aide de sa clé.

Voici un exemple d'utilisation :

```
>>> arbre = Noeud(7)
>>>for cle in (3, 9, 1, 6):
       arbre.inserer(cle)

>>> arbre.gauche.getValeur()
3
>>> arbre.droit.getValeur()
9
>>> arbre.gauche.gauche.getValeur()
1
>>> arbre.gauche.droit.getValeur()
6
```