# BACCALAURÉAT

## SESSION 2025

---

## Épreuve de l'enseignement de spécialité

# NUMÉRIQUE et SCIENCES INFORMATIQUES

## Partie pratique

## Classe Terminale de la voie générale

---

## Sujet n°36

---

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)

Dans cet exercice, on considère des phrases composées de mots.

- On appelle *mot* une chaîne de caractères composée avec des caractères choisis parmi les 26 lettres minuscules ou majuscules de l'alphabet.
- On appelle *phrase* une chaîne de caractères :
  - composée avec un ou 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 nombre 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 cette phrase.

```
>>> nombre_de_mots('Cet exercice est simple.')
4
>>> nombre_de_mots('Le point d exclamation est séparé !')
6
>>> nombre_de_mots('Combien de mots y a t il dans cette phrase ?')
10
>>> nombre_de_mots('Fin.')
1
```

## EXERCICE 2 (10 points)

Un arbre binaire de recherche est soit vide, représenté en Python par la valeur `None`, soit un nœud, contenant une étiquette et deux sous-arbres gauche et droit et représenté par une instance de la classe `Noeud` donnée ci-dessous.

On considère ici que les étiquettes des nœuds sont des entiers et que les arbres binaires de recherche considérés ne contiennent pas de doublons.

```
class Noeud:
    def __init__(self, etiquette):
        '''Méthode constructeur pour la classe Noeud.
        Crée une feuille d'étiquette donnée.'''
        self.etiquette = etiquette
        self.gauche = None
        self.droit = None

    def inserer(self, cle):
        '''Insère la clé dans l'arbre binaire de recherche
        en préservant sa structure.'''
        if cle < self.etiquette:
            if self.gauche != None:
                ...
            else:
                self.gauche = ...
        else:
            ...
                ...
            else:
                ... = Noeud(cle)
```

Compléter la méthode récursive `inserer` afin qu'elle permette d'insérer une clé dans l'arbre binaire de recherche non vide sur lequel on l'appelle.

Voici un exemple d'utilisation :

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