2022 sujet 17

Exercice 1

```
def nombre_de_mots(phrase):
    compteur = 0
    for c in phrase:
        if c == " ":
            compteur += 1
    if phrase[-1] != ".":
        return compteur
    else:
        return compteur + 1
```

Exercice 2

```
# Attention !!!
# Il ne faut jamais mettre un objet mutable comme valeur par défaut
# d'un paramètre de fonction car il est évalué une seule fois
# lorsque la fonction est définie.
# La méthode parcours risque donc d'avoir un comportement erratique

class Noeud:
    '''
    Classe implémentant un noeud d'arbre binaire
    disposant de 3 attributs :
    - valeur : la valeur de l'étiquette,
    - gauche : le sous-arbre gauche.
    - droit : le sous-arbre droit.
    '''
    def __init__(self, v, g, d):
        self.valeur = v
        self.gauche = g
        self.droite = d

class ABR:
    '''
    Classe implémentant une structure
    d'arbre binaire de recherche.
    '''

    def __init__(self):
```

```
        '''Crée un arbre binaire de recherche vide'''
        self.racine = None

    def est_vide(self):
        '''Renvoie True si l'ABR est vide et False sinon.'''
        return self.racine is None

    def parcours(self, tab = []):
        '''
        Renvoie la liste tab complétée avec tous les éléments de
        l'ABR triés par ordre croissant.
        '''
        if self.est_vide():
            return tab
        else:
            self.racine.gauche.parcours(tab)
            tab.append(self.racine.valeur)
            self.racine.droite.parcours(tab)
            return tab

    def insere(self, element):
        '''Insère un élément dans l'arbre binaire de recherche.'''
        if self.est_vide():
            self.racine = Noeud(element, ABR(), ABR())
        else:
            if element < self.racine.valeur:
                self.racine.gauche.insere(element)
            else :
                self.racine.droite.insere(element)

    def recherche(self, element):
        '''
        Renvoie True si element est présent dans l'arbre
        binaire et False sinon.
         '''
        if self.est_vide():
            return False
        else:
            if element < self.racine.valeur:
                return self.racine.gauche.recherche(element)
            elif element > self.racine.valeur:
                return self.racine.droite.recherche(element)
            else:
                return True

a = ABR()
a.insere(7)
a.insere(3)
a.insere(9)
a.insere(1)
```

```
a.insere(9)
```