Nouvelle Calédonie – 2023 – sujet2 - Correction

Exercice 2 (4 points)

1.

a. Spécifications de la fonction `palindrome`

```
def palindrome(chaine):
    """
    Paramètre :
        chaine (str) : mot dont on veut tester s'il est un
palindrome.
    Retour :
        bool : True si le mot est un palindrome, False sinon.
    Rôle :
        Tester si une chaîne de caractères s'écrit de la même façon
        à l'endroit et à l'envers.
    """
```

👉 *Commentaire : une docstring précise toujours les paramètres, leur type, le type de retour et le rôle de la fonction. Toujours avec 3 guillemets.*

---

b. Complexité en temps

Pour un mot de longueur n, l'algorithme compare les caractères deux à deux.

Nombre de comparaisons ≈ n/2

Complexité : **O(n)**

👉 *Commentaire : on ne garde que le terme dominant. La complexité est linéaire car le nombre de comparaisons est proportionnel à n.*

---

c. Définition d'une fonction récursive

Une fonction récursive est une fonction **qui s'appelle elle-même**.

Elle comporte :

- un **cas de base** (condition d'arrêt) ;
- un **appel récursif** sur un problème plus petit.

👉 *Commentaire : sans condition d'arrêt, une fonction récursive provoquerait une récursion infinie.*

---

d. Fonction `palindrome_recursive`

Complétion des lignes 2, 4 et 5 :

```
def palindrome_recursive(chaine):
    if len(chaine) <= 1:
        return True
    if chaine[0] != chaine[-1]:
        return False
    return palindrome_recursive(chaine[1:-1])
```

👉 *Commentaire :*

- *on compare le premier et le dernier caractère puis on applique la fonction sur la sous-chaîne privée de ces deux caractères.*
- `chaine[-1]` *est équivalent à* `chaine[len(chaine) -1]`
- `palindrome_recursive(chaine[1:-1])` *est équivalent à* `palindrome_recursive(chaine[1:len(chaine) -1])`
- *Bien se souvenir que dans un slice* `[a:b]`*, la première valeur ,* `a` *est incluse et la deuxième,* `b` *exclue de l'intervalle.*

⚠️ *Il n'y a pas de question e. C'est une erreur du sujet.*

2.

a. Fonction `test_mot`

```
def test_mot(mot_mystere, mot_propose):
    resultat = ""
    for i in range(len(mot_mystere)):
        if mot_propose[i] == mot_mystere[i]:
            resultat += mot_mystere[i]
        else:
            resultat += "-"
    return resultat
```

Exemple :

```
>>> test_mot('MOMIE','MINCE')
'M---E'
```

👉 *Commentaire : on compare les lettres à la même position dans les deux mots.*

---

b. Pourquoi le programme `jeu_motus` ne donne pas toujours le résultat attendu ?

Le problème classique dans ce type de programme est :

- le compteur d'essais mal géré (incrémenté au mauvais moment) ;
- la condition d'arrêt incorrecte ;
- ou la vérification de victoire (variable `gagne`) placée après l'incrémentation.

Par exemple, si le programme incrémente le compteur avant de tester la victoire, le nombre de coups affiché peut être faux.

👉 *Commentaire : dans une boucle contrôlée par un nombre d'essais, l'ordre des instructions (test de victoire / incrémentation) est essentiel.*
