Numérique et science informatique
Classe de terminale 2024-25
Contrôle n° 1
Sujet A
Lundi 14 octobre 2024

I Récursivité

1. On a écrit la fonction suivante :

```
def somme(n) :
    """ prend en argument un entier positif
    et renvoie la somme des entiers de 0 à n inclus.
    Programmation récursive.
    """
    if n == 0 :
        return 0
    return n + somme(n-1)
```

a. Expliquer en une phrase pourquoi cette fonction met en œuvre une programmation récursive.

b. Qu'est-ce qu'un « cas de base » dans une fonction récursive ? Quel est le cas de base dans la fonction somme ?

c. Que s'affiche-t-il alors dans la console si on exécute le programme principal ci-dessous ?

```
for k in range(4) :
    res = somme(k)
    print("Pour", k, "le résultat est :", res)
```

d. Décrivez et expliquez, en une ou deux phrase(s), ce qui se passe si on exécute le programme principal suivant :

```
res = somme(-1)
print("Pour -1 le résultat est :", res)
```

e. Ecrivez une fonction `somme_iter(n)` qui donne toujours le même résultat que la fonction `somme` mais écrite dans une version non récursive à l'aide d'une boucle (on parle de programmation itérative).

2. On considère la suite (u_n) définie par récurrence de la façon suivante :

$$u_0 = 7 \text{ et, pour tout entier positif } n,\ u_{n+1} = -2u_n + 10$$

Recopiez et complétez le code de la fonction `U(n)` qui prend en argument un entier positif ou nul `n` et qui renvoie la valeur de terme de rang n de la suite (u_n).

```
def U(n) :
    """renvoie la valeur du terme de rang n
    Programmation récursive.
    """
    if n == 0 :
        return ...
    prec = U(...)
    return ...
```

II Variables globales et locales

3. Indiquez dans chacun des deux cas suivants ce qui s'affiche dans la console :

a. On exécute le code suivant :

```
def tripler(n) :
    n = 3*n
    return n

A = 11
B = tripler(A)
print("A :", A, "B :", B)
```

b. On exécute le code suivant :

```
def quadrupler() :
    global E
    E = 4*E
    F = 4*F

E = 20
F = 30
quadrupler()
print("E :", E, "F :", F)
```

4. Indiquez dans chacun des deux cas suivants ce qui s'affiche dans la console :

a. On exécute le code suivant :

```
def doubler_tableau1(tab) :
    for i in range(len(tab)) :
        tab[i] = 2*tab[i]
    return tab

T1 = [45, 7, 13, 23]
T2 = doubler_tableau1(T1)
print("T1 :", T1)
print("T2 :", T2)
```

b. On exécute le code suivant :

```
def doubler_tableau2(tab) :
    tab2 = [0]*len(tab)
    for i in range(len(tab)) :
        tab2[i] = 2*tab[i]
    return tab2

T3 = [25, 9, 11, 44]
T4 = doubler_tableau1(T3)
print("T3 :", T1)
print("T4 :", T2)
```

III Deux algorithmes de tri

5. On considère le tableau suivant :

`[28, 4, 99, 21, 48]`

a. Présentez les différentes étapes du tri de ce tableau selon l'algorithme de tri par sélection.

b. Présentez les différentes étapes du tri de ce tableau selon l'algorithme de tri par insertion.
Remarque : on utilisera un système de deux couleurs pour bien distinguer la partie triée de la partie non triée du tableau.

6. a. Rappeler la complexité des algorithmes de tri par sélection et de tri par insertion.

b. Une fonction a mis 3 ms a trié un tableau de 1000 éléments en utilisant un algorithme de tri par sélection. Combien de temps devrait-elle mettre sur le même ordinateur pour trier un tableau de 5000 éléments ?

IV Diviser pour régner

7. On considère ci-dessous le code incomplet de la fonction `dichotomie` qui met en œuvre la recherche dichotomique dans un tableau de nombres à l'aide d'une programmation récursive.

```
def dichotomie(val, tab, g, d) :
    """* 'tab' est un tableau (de type list)de nombres entiers
       * on suppose que' le tableau 'tab' est ...
       * 'val' est la valeur à trouver (de type int)
       * 'g' et 'd' sont ... (de type ...).
       * La fonction renvoie un indice du tableau
        où la valeur se trouve
        ou bien None si elle ne s'y trouve pas.
     """
    if ... : # la zone de recherche est vide
        return ...
    m = ... # indice du milieu de la zone de recherche
    if val < tab[m] :
        return dichotomie(..., ..., ..., ...)
    elif val > tab[m] :
        return dichotomie(..., ..., ..., ...)
    else :
        return ...
```

a. Recopier et compléter les 2ème et 4ème lignes de la *docstring*.

b. Recopier et compléter le corps de la fonction.

c. Proposer quelques tests pertinents de la fonction `dichotomie`.

8. On veut mettre en œuvre le tri fusion sur un tableau par programmation récursive. On a commencé à écrire le code suivant :

```
def tri_fusion(tab) :
    lg = len(tab)
    if ... :
        return ...
    tab1, tab2 = decouper(tab)
    tab1 = ...(tab1)
    tab2 = ...(tab2)
    return fusion(tab1, tab2)
```

a. Écrire la *docstring* des fonctions `decouper` et `fusion` en précisant les conditions sur les arguments donnés en entrée et le résultat renvoyé.

b. Recopier et compléter le code de la fonction `tri_fusion`.

c. Proposer un programme de test de la fonction `tri_fusion` (en veillant à proposer des appels corrects de cette fonction).

d. Quelle est la complexité de l'algorithme de tri par fusion ?

e. Pour trier un tableau, on a le choix entre deux fonctions `tri_insertion` et `tri_fusion` qui mettent respectivement en œuvre un tri par insertion et un tri par fusion. Laquelle choisira-t-on et pourquoi ?