Devoir surveillé n° 1 – partie 1
informatique
MP
vendredi 5 septembre 2025
durée 1 heure

Questions de cours

1. Quelle instruction permet de tester si un entier n est pair ? multiple de 10 ?
2. `L` est une liste de nombres. Que fait le programme suivant et quelle est sa complexité ?

```
def inconnu(L):
    c = 0
    for valeur in L:
        if valeur == 18:
            c += 1
    return c
```

Exercice 1

1. Écrire une fonction `moyenne(X)` qui prend en paramètre `X`, une liste de nombres et qui calcule la moyenne de ces nombres.
   Par exemple : `moyenne([1, 2, 3, 4])` retourne 2.5
2. Écrire une fonction `variance(X)` qui calcule la variance d'une liste de nombres.
   Pour rappel, la variance des n nombres $x_1, x_2, \dots, x_n$ est la moyenne des carrés des écarts à la moyenne, c'est-à-dire

$$\frac{1}{n}\sum_{i=1}^{n}(x_i - \overline{x})^2 = \frac{1}{n}\sum_{i=1}^{n}x_i^2 - \overline{x}^2 \qquad \text{avec } \overline{x} = \frac{1}{n}\sum_{i=1}^{n}x_i$$

   Par exemple : `variance([1, 2, 3, 4])` retourne 1.25
3. Écrire une fonction récursive `calcul_somme(M)` qui prend en paramètre une liste imbriquée, de profondeur et de structure quelconques, dont tous les composants élémentaires sont des nombres, et calcule la somme de tous ces éléments.
   Par exemple : `calcul_somme([[[1, 2], [3, 4, 5]], 6, [7, 8], 9])` retourne 45
   Indication. Après import du module `numbers`, on pourra utiliser l'expression booléenne `isinstance(x, numbers.Real)` qui permet de tester si x est un nombre réel. Par exemple :
   `isinstance(1, numbers.Real) -> True`
   `isinstance(2.3e4, numbers.Real) -> True`
   `isinstance([1, 2, 3], numbers.Real) -> False`

Exercice 2

On se donne un tableau à une dimension, de longueur n dont les entrées sont indexées de 0 à $n-1$, ne contenant que des 0 et des 1, commençant et se terminant par un 0. On supposera qu'il ne contient pas que des 0.
Dans un tel tableau, on appelle **séquencel** une suite de 1 consécutifs précédés et suivis d'au moins un 0. Ci-dessous, on a un exemple d'un tel tableau de longueur 16, comportant 4 **séquence1**, dont une de longueur 4 entre les numéros 7 et 10, et une en 14 de longueur 1.
On se propose d'écrire des fonctions en Python permettant de calculer le nombre, la longueur et la position de telles **séquence1**.

Indice i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Tableau1[i]	0	0	1	1	1	1	0	1	1	1	1	0	1	0	1	0

1. Soit la fonction `f` écrite en Python, avec en paramètre d'entrée un tableau `t` défini précédemment.

```
def f(t):
    nb = 0
    n = len(t)
    for i in range(n-1):
        if t[i] < t[i+1]:
            nb = nb + 1
    return nb
```

   Expliquer ce que renvoie `f(Tableau1)` et préciser le fonctionnement de f.

2. Écrire sur votre copie les instructions manquantes pour renvoyer la position et la longueur de la plus longue **séquence1** du tableau (en cas d'égalité de longueur, celle de la première rencontrée). Par exemple, `g(Tableaul)` doit renvoyer la liste `[2, 5, 4]`.

```
def g(t):
    n = len(t)
    nb = f(t)
    assert nb != 0, 'hypothèse énoncé non satisfaite'

    # on stocke les positions des alternances 0-1
    positions = []
    for i in range(n-1):
        if t[i] < t[i+1]:
            ...............

    maxi = 0
    debut = 0
    fin = 0

    for indice in positions:
        j = indice
        longueur = 0
        while ............:
            longueur += 1
            j += 1
        if ...............:
            maxi = longueur
            debut = indice
            fin = j-1

    return [debut, fin, maxi]
```

Exercice 3

Une liste non vide d'entiers naturels est *bitonique* si elle contient une suite (éventuellement vide) croissante suivie d'une suite (éventuellement vide) décroissante. Une liste vide n'est pas bitonique.
Exemples :
Les listes suivantes sont bitoniques :

- 1, 4, 8, 12, 5, 2
- 12, 25, 40, 52
- 15, 8, 3

Les listes suivantes ne sont pas bitoniques :

- 1, 5, 10, 8, 6, 12
- 15, 12, 11, 9, 10, 14, 16

Écrire la fonction Python `bitonique(L)` qui détermine si une liste `L` est bitonique. Si c'est le cas, votre fonction devra retourner la valeur du point culminant (la valeur la plus haute) de la liste, et retournera la valeur -1 sinon.
Par exemple, dans la liste `[1, 4, 8, 12, 5, 2]` le point culminant est 12. Dans la liste `[12, 25, 40, 52]` c'est 52.