BACCALAUREAT

SESSION 2022

Épreuve de l'enseignement de spécialité

NUMERIQUE et SCIENCES INFORMATIQUES

Partie pratique

Classe Terminale de la voie générale

Sujet n°35

DUREE DE L'EPREUVE : 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 (4 points)

Ecrire une fonction qui prend en paramètre un tableau d'entiers non vide et qui renvoie la moyenne de ces entiers. La fonction est spécifiée ci-après et doit passer les assertions fournies.

```
def moyenne (tab):
    '''
        moyenne(list) -> float
        Entrée : un tableau non vide d'entiers
        Sortie : nombre de type float
        Correspondant à la moyenne des valeurs présentes dans le
        tableau
    '''

assert moyenne([1]) == 1
assert moyenne([1,2,3,4,5,6,7]) == 4
assert moyenne([1,2]) == 1.5
```

EXERCICE 2 (4 points)

Le but de l'exercice est de compléter une fonction qui détermine si une valeur est présente dans un tableau de valeurs triées dans l'ordre croissant.

L'algorithme traite le cas du tableau vide.

L'algorithme est écrit pour que la recherche dichotomique ne se fasse que dans le cas où la valeur est comprise entre les valeurs extrêmes du tableau.

On distingue les trois cas qui renvoient `False` en renvoyant `False,1` , `False,2` et `False,3`.

Compléter l'algorithme de dichotomie donné ci-après.

```
def dichotomie(tab, x):
    """
        tab : tableau trié dans l'ordre croissant
        x : nombre entier
        La fonction renvoie True si tab contient x et False sinon
    """
    # cas du tableau vide
    if ...:
        return False,1

    # cas où x n'est pas compris entre les valeurs extrêmes
    if (x < tab[0]) or ...:
        return False,2
```

```
    debut = 0
    fin = len(tab) - 1
    while debut <= fin:
        m = ...
        if x == tab[m]:
            return ...
        if x > tab[m]:
            debut = m + 1
        else:
            fin = ...
    return ...
```

Exemples :

```
>>> dichotomie([15, 16, 18, 19, 23, 24, 28, 29, 31, 33],28)
True
>>> dichotomie([15, 16, 18, 19, 23, 24, 28, 29, 31, 33],27)
(False, 3)
>>> dichotomie([15, 16, 18, 19, 23, 24, 28, 29, 31, 33],1)
(False, 2)
>>> dichotomie([],28)
(False, 1)
```

```
"""
Author: Pascal Padilla
Source: correction de l'exercice 1 du sujet 35 des épreuves pratiques NSI 2022

Remarques:

"""


def moyenne(tab):
    '''
        moyenne(list) -> float
        Entrée : un tableau non vide d'entiers
        Sortie : nombre de type float
        Correspondant à la moyenne des valeurs présentes dans le
        tableau

    Tests et exemples:
    >>> moyenne([1])
    1.0
    >>> moyenne([1,2,3,4,5,6,7])
    4.0
    >>> moyenne([1,2])
    1.5
    '''
    # nombre de cases du tableau :
    nb_valeurs = len(tab)

    # initialisation invariant
    #  -> somme contient la somme des valeurs de tab[0 .. i-1]
    somme = 0

    # initialisation
    #  -> i = 0
    # condition d'arrêt
    #  -> i == n
    for i in range(0, nb_valeurs):
        somme += tab[i]

    # après la boucle
    #  -> tout le tableau est parcouru
    #  -> donc somme contient la somme des valeurs de TOUT le tableau

    # calcul de la moyenne
    return somme/nb_valeurs


# vérification avec des assertions
assert moyenne([1]) == 1
assert moyenne([1,2,3,4,5,6,7]) == 4
assert moyenne([1,2]) == 1.5

# vérification avec des affichagges
print(moyenne([1]))
print(moyenne([1,2,3,4,5,6,7]))
print(moyenne([1,2]))

# vérification avec doctest
from doctest import testmod
testmod()
```

```
"""
Author: Pascal Padilla
Source: correction de l'exercice 2 du sujet 35 des épreuves pratiques NSI 2022

Remarques:
    * version itérative de la recherche dichotomique
    * le renvoie `return False, 1` renvoie en fait un tuple : `return (False, 1)
    * la distinction des trois cas False (avec 1, 2 e 3) permet
    de bien mettre en évidence pourquoi l'élément n'y est pas :
      -> cas 1 : tableau vide
      -> cas 2 : valeur hors limite
      -> cas 3 : recherche effectuée mais infructueuse
"""



def dichotomie(tab, x):
    """
        tab : tableau trié dans l¿ordre croissant
        x : nombre entier
        La fonction renvoie True si tab contient x et False sinon
    """
    # cas du tableau vide
    if tab == [] :
        # renvoie et donc fin de la fonction
        return False, 1

    # cas où x n'est pas compris entre les valeurs extrêmes
    if (x < tab[0]) or (x > tab[len(tab)-1]):
        # renvoie et donc fin de la fonction
        return False, 2

    debut = 0
    fin = len(tab) - 1

    # condition d'arrêt :
    #  -> la zone de recherche tab[debut .. fin] est un tableau vide
    #     et donc debut devient supérieur à fin
    while debut <= fin:
        # valeur médiane :
        m = (debut + fin) // 2

        # l'élément cherché est égale à la valeur médiane : trouvé !
        if x == tab[m]:
            # renvoie et fin de la fonction
            return True

        # recherche dans la partie supérieure à la case médiane
        if x > tab[m]:
            debut = m + 1

        # recherche dans la partie inférieure à la case médiane
        else:
            fin = m - 1

    # SORTIE de boucle
    #  la zone de recherche est vide
    #  et l'élément n'a pas encore été trouvé

    # renvoie infructueux associé au cas numéro 3
    return False, 3


# vérification par des assertions
assert dichotomie([15, 16, 18, 19, 23, 24, 28, 29, 31, 33],28) == True
assert dichotomie([15, 16, 18, 19, 23, 24, 28, 29, 31, 33],27) == (False, 3)
assert dichotomie([15, 16, 18, 19, 23, 24, 28, 29, 31, 33],1) == (False, 2)
assert dichotomie([],28) == (False, 1)

# vérification par des affichages

print(dichotomie([15, 16, 18, 19, 23, 24, 28, 29, 31, 33],28))
print(dichotomie([15, 16, 18, 19, 23, 24, 28, 29, 31, 33],27))
```

```
print(dichotomie([15, 16, 18, 19, 23, 24, 28, 29, 31, 33],1))
print(dichotomie([],28))
```