BACCALAUREAT

SESSION 2022

Épreuve de l'enseignement de spécialité

NUMERIQUE et SCIENCES INFORMATIQUES

Partie pratique

Classe Terminale de la voie générale

Sujet n°16

DUREE DE L'EPREUVE : 1 heure

Le sujet comporte 2 pages numérotées de 1 / 2 à 2 / 2
Dès que le sujet vous est remis, assurez-vous qu'il est complet.

Le candidat doit traiter les 2 exercices.

EXERCICE 1 (4 points)

Écrire une fonction `maxi` qui prend en paramètre une liste `tab` de nombres entiers et renvoie un couple donnant le plus grand élément de cette liste, ainsi que l'indice de la première apparition de ce maximum dans la liste.

Exemple :

```
>>> maxi([1,5,6,9,1,2,3,7,9,8])
(9,3)
```

EXERCICE 2 (4 points)

Cet exercice utilise des piles qui seront représentées en Python par des listes (type `list`). On rappelle que l'expression `T1 = list(T)` fait une copie de `T` indépendante de `T`, que l'expression `x = T.pop()` enlève le sommet de la pile `T` et le place dans la variable `x` et, enfin, que l'expression `T.append(v)` place la valeur `v` au sommet de la pile `T`.

Compléter le code Python de la fonction `positif` ci-dessous qui prend une pile T de nombres entiers en paramètre et qui renvoie la pile des entiers positifs dans le même ordre, sans modifier la variable `T`.

```
def positif(T):
    T2 = ...(T)
    T3 = ...
    while T2 != []:
        x = ...
        if ... >= 0:
            T3.append(...)
    T2 = []
    while T3 != ...:
        x = T3.pop()
        ...
    print('T = ',T)
    return T2
```

Exemple :

```
>>> positif([-1,0,5,-3,4,-6,10,9,-8 ])
T = [-1, 0, 5, -3, 4, -6, 10, 9, -8]
[0, 5, 4, 10, 9]
```

```python
"""
Author: Pascal Padilla
Source: correction de l'exercice 1 du sujet 16 des épreuves pratiques NSI 2022
"""


from doctest import testmod


def maxi(tab: list) -> tuple[int, int]:
    """Renvoie le plus grand élément du tableau 'tab'
    et son indice.

    Args:
        tab (list): tableau à parcourir

    Returns:
        tuple[int, int] : couple de nombres entiers :
            * valeur maximale du tableau
            * indice de cette valeur

    Exemples et tests:
    >>> maxi([1,5,6,9,1,2,3,7,9,8])
    (9, 3)
    """

    # invariant de boucle
    #   * v_max est la valeur maximale de la portion tab[0..i]
    #   * i_max est l'indice de v_max

    # initialisation (avec la première valeur du tableau):
    #   * i ¿ 0
    #   * v_max ¿ tab[0]
    #   * i_max ¿ 0
    i_max = 0
    v_max = tab[0]

    # condition d'arrêt (tout le tableau est parcouru):
    #   * i == len(tab)
    for i in range(1, len(tab)):

        # mise à jour de v_max et i_max si la
        # valeur courante du tableau est plus grande
        # que le maximum trouvé jusqu'à présent
        if tab[i] > v_max:
            i_max = i
            v_max = tab[i]

    # sortie de boucle : tout le tableau est parcouru
    # et grâce aux invariants, nous avons la valeur
    # maximale et son indice.
    return (v_max, i_max)


# Tests de l'énoncé: méthode classique
assert maxi([1,5,6,9,1,2,3,7,9,8]) == (9,3)

# Tests de l'énoncé: méthode avec doctest
testmod()
```

```
"""
Author: Pascal Padilla
Source: correction de l'exercice 2 du sujet 16 des épreuves pratiques NSI 2022

Remarques:
    * quand vous créez vos propres programmes, utiliser des noms de
    variables en minuscules (sauf pour les CONSTANTES)
"""


def positif(T):
    # l'idée de l'algo (pour laisser T inchangée) est:
    #   1. copier T dans une autre pile (T2)
    #   2. dépiler T2 dans T3 en ignorant les nombres négatifs
    # La pile T3 contient les bons éléments, mais dans l'ordre inverse
    #   3. dépiler T3 dans T2 (qui est vide au début)  pour rétablir l'ordre
    #      des éléments

    # BOUCLE n°1
    # invariant:
    #   * T3 contient tous les éléments dépilés de T2, sauf les négatifs
    #   * T3 et T2 contiennent tous les éléments positifs de T

    # initialisation:
    #   * T2 ¿ copie de T
    #   * T3 ¿ pile vide
    T2 = list(T)
    T3 = []

    # condition d'arrêt:
    #   * T2 n'as plus aucun élément
    while T2 != []:
        # dépiler un élément de T2
        x = T2.pop()

        # et l'empiler dans T3 seulement s'il est positif
        if x >= 0:
            T3.append(x)

    # BOUCLE n°2
    # invariant: T3 et T2 contiennent tous les éléments positifs de T

    # initialisation: T2 est vide (c'est déjà le cas, mais on le formalise)
    T2 = []

    # condition d'arrêt: T3 est vide
    while T3 != []:
        x = T3.pop()
        T2.append(x)

    # on vérifie que 'T' est inchangée
    print('T = ',T)

    # on renvoie T2
    return T2


# Test de l'énoncé ajouté pour tester la fonction
assert positif([-1, 0, 5, -3, 4, -6, 10, 9, -8]) == [0, 5, 4, 10, 9]

# Autre façon de tester avec un affichage
print() # pour sauter une ligne
print("Autre test possible avec un affichage:")
print( positif([-1, 0, 5, -3, 4, -6, 10, 9, -8]) )
```