# BACCALAUREAT

## SESSION 2022

---

## Épreuve de l'enseignement de spécialité

# NUMERIQUE et SCIENCES INFORMATIQUES

# Partie pratique

## Classe Terminale de la voie générale

---

**Sujet n°26**

---

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 `RechercheMin` qui prend en paramètre un tableau de nombres non trié `tab`, et qui renvoie l'indice de la première occurrence du minimum de ce tableau. Les tableaux seront représentés sous forme de liste Python.

Exemples :

```
>>> RechercheMin([5])
0
>>> RechercheMin([2, 4, 1])
2
>>> RechercheMin([5, 3, 2, 2, 4])
2
```

## EXERCICE 2 (4 points)

On considère la fonction `separe` ci-dessous qui prend en argument un tableau `tab` dont les éléments sont des `0` et des `1` et qui sépare les `0` des `1` en plaçant les `0` en début de tableau et les `1` à la suite.

```
def separe(tab):
    i = 0
    j = ...
    while i < j :
        if tab[i] == 0 :
            i = ...
        else :
            tab[i], tab[j] = ...
            j = ...
    return tab
```

Compléter la fonction `separe` ci-dessus.

Exemples :

```
>>> separe([1, 0, 1, 0, 1, 0, 1, 0])
[0, 0, 0, 0, 1, 1, 1, 1]

>>> separe([1, 0, 0, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 0])
[0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1]
```

```
"""
Author: Pascal Padilla
Source: correction de l'exercice 1 du sujet 26 des épreuves pratiques NSI 2022

Remarques:
  * pas de noms de fonctions avec des majuscules
  * les noms doivent respecter le format snake_case :
  RechercheMin -> recherche_min
"""


def RechercheMin(tab: list) -> int:
    """Renvoie l'indice de la première occurence du minimum

    Args:
        tab (list): tableau d'entiers relatifs

    Returns:
        int: indice du min du tableau

    Tests et exemples:
    >>> RechercheMin([5])
    0
    >>> RechercheMin([2, 4, 1])
    2
    >>> RechercheMin([5, 3, 2, 2, 4])
    2
    """
    # clause de garde (non demandée)
    n = len(tab)
    if n == 0:
        return None

    # BOUCLE
    # invariant
    #  -> i_min est l'indice du min de la zone tab[0 .. i-1]
    #  -> v_min est la valeur minimale de la zone tab[0 .. i-1]
    #  -> i est l'indice en cours d'analyse
    #
    # initialisation
    #  -> avec la première valeur du tableau
    #  (qui existe grâce à la clause de garde)
    #  -> i_min <- 0 et v_min <- tab[0]
    #  -> commencer à partir de la 2ème case: i <- 1
    i_min = 0
    v_min = tab[0]  # pas utile mais rend le code plus lisible !

    # condition d'arrêt:
    #  -> i sort du tableau: i >= n
    for i in range(1, n):
        # conservation de l'invariant
        # lorsqu'un élément plus petit est trouvé
        if tab[i] < v_min:
            i_min = i
            v_min = tab[i]

    # renvoie de l'indice de la valeur minimale
    return i_min


# tests avec des affichages
print(RechercheMin([5]))
print(RechercheMin([2, 4, 1]))
print(RechercheMin([5, 3, 2, 2, 4]))


# tests avec des assertions
assert RechercheMin([5]) == 0
assert RechercheMin([2, 4, 1]) == 2
assert RechercheMin([5, 3, 2, 2, 4]) == 2


# tests avec doctest
```

```
from doctest import testmod
testmod()
```

```
"""
Author: Pascal Padilla
Source: correction de l'exercice 2 du sujet 26 des épreuves pratiques NSI 2022

Remarque:
    l'algorithme fait penser à la dichotomie car il partage le tableau en plusieurs
zones

    l'algorithme partage le tableau en 3 zones grâce aux indices i et j:
      * celle qui ne contient que des '0' (du début jusqu'à i-1)
      * celle qui ne contient que des '1' (de j+1 à la fin)
      * celle qui n'est pas encore triée  (entre i et j)
"""


def separe(tab):
    # BOUCLE
    # invariants
    #  -> tab[0.. i-1]  contient que des '0'
    #  -> tab[j+1..fin] contient que des '1'
    #  -> tab[i..j] contient un mélange de '0' et de '1' non triés
    # initialisation
    #  -> le tableau entier n'est pas encore trié:
    #     i <- 0 et j <- dernier indice
    i = 0
    j = len(tab) - 1

    # condition d'arrêt
    #  -> la zone à trier est vide : i et j se 'croisent'
    while i < j :
        # le premier élément non trié est un '0'
        if tab[i] == 0 :
            # ne rien faire et diminuer le début de la zone non triée
            # tab [0..i] ne contient que des '0'
            i = i + 1

        # le premier élément non trié est un '1'
        else :
            #  permuter le premier élément avec le dernier non trié
            tab[i], tab[j] = tab[j], tab[i]
            # puis diminuer la fin de la zone non triée
            # tab[j..fin] contient que des '1'
            j = j - 1

    # le tableau tab est trié
    return tab


# tests avec des assertions
assert separe([1, 0, 1, 0, 1, 0, 1, 0]) == [0, 0, 0, 0, 1, 1, 1, 1]
assert separe([1, 0, 0, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 0]) == [0, 0, 0, 0, 0, 0
, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1]


# tests avec des affichages
print(separe([1, 0, 1, 0, 1, 0, 1, 0]))
print(separe([1, 0, 0, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 0]))
```