BACCALAUREAT

SESSION 2022

Épreuve de l'enseignement de spécialité

NUMERIQUE et SCIENCES INFORMATIQUES

Partie pratique

Classe Terminale de la voie générale

Sujet n°38

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)

Écrire une fonction `tri_selection` qui prend en paramètre une liste `tab` de nombres entiers et qui renvoie le tableau trié par ordre croissant.

On utilisera l'algorithme suivant :

- on recherche le plus petit élément du tableau, et on l'échange avec l'élément d'indice 0 ;
- on recherche le second plus petit élément du tableau, et on l'échange avec l'élément d'indice 1 ;
- on continue de cette façon jusqu'à ce que le tableau soit entièrement trié.

Exemple :

```
>>> tri_selection([1,52,6,-9,12])
[-9, 1, 6, 12, 52]
```

EXERCICE 2 (4 points)

Le jeu du « plus ou moins » consiste à deviner un nombre entier choisi entre 1 et 99.

Un élève de NSI décide de le coder en langage Python de la manière suivante :

- le programme génère un nombre entier aléatoire compris entre 1 et 99 ;
- si la proposition de l'utilisateur est plus petite que le nombre cherché, l'utilisateur en est averti. Il peut alors en tester un autre ;
- si la proposition de l'utilisateur est plus grande que le nombre cherché, l'utilisateur en est averti. Il peut alors en tester un autre ;
- si l'utilisateur trouve le bon nombre en 10 essais ou moins, il gagne ;
- si l'utilisateur a fait plus de 10 essais sans trouver le bon nombre, il perd.

La fonction `randint` est utilisée. Si a et b sont des entiers, `randint(a,b)` renvoie un nombre entier compris entre a et b.

Compléter le code ci-dessous et le tester :

```
from random import randint

def plus_ou_moins():
    nb_mystere = randint(1,...)
    nb_test = int(input("Proposez un nombre entre 1 et 99 : "))
    compteur = ...

    while nb_mystere != ... and compteur < ... :
        compteur = compteur + ...
        if nb_mystere ... nb_test:
            nb_test = int(input("Trop petit ! Testez encore : "))
        else:
            nb_test = int(input("Trop grand ! Testez encore : "))

    if nb_mystere == nb_test:
        print ("Bravo ! Le nombre était ",...)
        print("Nombre d'essais: ",...)
    else:
        print ("Perdu ! Le nombre était ",...)
```

```
"""
Author: Pascal Padilla
Source: correction de l'exercice 1 du sujet 38 des épreuves pratiques NSI 2022

Remarques:
    * toujours délicat car il y a deux boucles !
"""


def tri_selection(tab: list) -> list:
    """ Tri par sélection du tableau tab

    Exemples et tests:
    >>> tri_selection([1,52,6,-9,12])
    [-9, 1, 6, 12, 52]
    """
    nb_elements = len(tab)
    # BOUCLE 1: parcourir tout le tableau (sauf la dernière case, ce n'est pas la pe
ine)
    # pour mettre au fur et à mesure les plus petits éléments devant, triés
    # -> invariant: tab[0 .. i-1] est trié avec les plus petits éléments de tab
    for i in range(nb_elements - 1):
        # BOUCLE 2: parcourir la zone tab[i .. dernière case] à la recherche
        # de l'indice du plus petit élément
        # -> invariant: i_min est l'indice du plus petit élément de la zone tab[i ..
 j-1]
        i_min = i
        # -> initialisation: j <- i+1 (et donc i_min est correct)
        # -> condition d'arrêt: après la boucle j <- indice de la dernière case (nb_
element-1)
        for j in range(i+1, nb_elements):
            # maintien de l'invariant si un élément plus petit est trouvé
            if tab[j] < tab[i_min]:
                # mise à jour de l'indice du plus petit élément
                i_min = j

        # fin BOUCLE 2: i_min est l'indice du plus petit élément de tab[i .. dernier
]

        # permutation entre élément d'indice `i` et `i_min`
        tab[i], tab[i_min] = tab[i_min], tab[i]

    # fin BOUCLE 1: tout le tableau est parcouru sauf la dernière case
    # toutes les cases avant la dernière sont donc triées et inférieures à la derniè
re case
    # le tableau est donc trié en entier (c'est pour cela que ce n'est pas la peine
de
    # faire un tour de boucle supplémentaire)

    # tableau trié correctement: renvoie
    return tab



# vérification avec des assertions
assert tri_selection([1,52,6,-9,12]) == [-9, 1, 6, 12, 52]

# vérification avec des affichages
print(tri_selection([1,52,6,-9,12])) # [-9, 1, 6, 12, 52]

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

```
"""
Author: Pascal Padilla
Source: correction de l'exercice 2 du sujet 38 des épreuves pratiques NSI 2022

Remarques:
    * ERREUR dans l'énoncé: attention dans les affichages à la fin.
    Dans l'énoncé on peut lire `était` et dans le fichier .py on lit `etait`.
    Pour les tests, j'ai fait comme le fichier : sans accent
    * dans `randint(a, b)`, b est INCLUS ! ce n'est pas comme `range(a, b)` !!!
    * pour pouvoir tester, ajouter un affichage pour tricher qui montre
    le nombre mystère... /!\ :supprimer cet affichage pour les TESTS automatisés ;)
"""


from random import randint

def plus_ou_moins():
    # nb aléatoire entre 1 et 99 inclus
    nb_mystere = randint(1, 99)

    nb_test = int(input("Proposez un nombre entre 1 et 99 : "))
    # -> invariant: compteur est le nombre d'essais effectués
    # -> initialisation: premier essai
    compteur = 1

    # -> condition d'arrêt:
    #   * le nombre est trouvé
    #   * le compteur arrive à 10 essais effectués
    while nb_mystere != nb_test and compteur < 10 :

        compteur = compteur + 1
        # cas d'un essai inférieur au nb mystère
        if nb_mystere > nb_test:
            nb_test = int(input("Trop petit ! Testez encore : "))
        else:
            nb_test = int(input("Trop grand ! Testez encore : "))

    if nb_mystere == nb_test:
        # affichage du nombre mystère
        print ("Bravo ! Le nombre etait ", nb_mystere)
        # et du nombre de tentatives
        print("Nombre d'essais: ", compteur)
    else:
        # affichage du nombre mystère
        print ("Perdu ! Le nombre etait ", nb_mystere)


# vérification avec des essais dans le terminal
# plus_ou_moins()
```