

BACCALAUREAT

SESSION 2022

Épreuve de l'enseignement de spécialité

NUMERIQUE et SCIENCES INFORMATIQUES

Partie pratique

Classe Terminale de la voie générale

Sujet n°37

DUREE DE L'ÉPREUVE : 1 heure

**Le sujet comporte 2 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)

Programmer la fonction `verifie` qui prend en paramètre un tableau de valeurs numériques non vide et qui renvoie `True` si ce tableau est trié dans l'ordre croissant, `False` sinon.

Exemples :

```
>>> verifie([0, 5, 8, 8, 9])
True
>>> verifie([8, 12, 4])
False
>>> verifie([-1, 4])
True
>>> verifie([5])
True
```

EXERCICE 2 (4 points)

Les résultats d'un vote ayant trois issues possibles 'A', 'B' et 'C' sont stockés dans un tableau.

Exemple :

```
Urne = ['A', 'A', 'A', 'B', 'C', 'B', 'C', 'B', 'C', 'B']
```

La fonction `depouille` doit permettre de compter le nombre de votes exprimés pour chacune des issues. Elle prend en paramètre un tableau et renvoie le résultat dans un dictionnaire dont les clés sont les noms des issues et les valeurs le nombre de votes en leur faveur.

La fonction `vainqueur` doit désigner le nom du ou des gagnants. Elle prend en paramètre un dictionnaire dont la structure est celle du dictionnaire renvoyé par la fonction `depouille` et renvoie un tableau. Ce tableau peut donc contenir plusieurs éléments s'il y a des ex-aequo.

Compléter les fonctions `depouille` et `vainqueur` ci-après pour qu'elles renvoient les résultats attendus.

```
def depouille(urne):
    resultat = ...
    for bulletin in urne:
        if ...:
            resultat[bulletin] = resultat[bulletin] + 1
        else:
            ...
    return resultat

def vainqueur(election):
    vainqueur = ''
    nmax = 0
```

```
for candidat in election:
    if ... > ... :
        nmax = ...
        vainqueur = candidat
liste_finale = [nom for nom in election if election[nom] == ...]
return ...
```

Exemples d'utilisation :

```
>>> election = depouille(urne)
>>> election
{'B': 4, 'A': 3, 'A': 3}

>>> vainqueur(election)
['B']
```

```
"""
```

```
Author: Pascal Padilla
```

```
Source: correction de l'exercice 1 du sujet 37 des épreuves pratiques NSI 2022
```

```
Remarques:
```

```
    * je propose deux version, boucle FOR et boucle WHILE  
"""
```

```
def verifie(tab: list) -> bool:  
    """ Renvoie vrai <=> est trié dans l'ordre croissant  
  
    Tests et exemples:  
>>> verifie([0, 5, 8, 8, 9])  
True  
>>> verifie([8, 12, 4])  
False  
>>> verifie([-1, 4])  
True  
>>> verifie([5])  
True  
    """  
    # Cas des tableaux vide et de taille 1: renvoie vrai car triés  
    if len(tab) <= 1:  
        return True  
  
    # BOUCLE:  
    # -> invariant: la zone tab[0 .. i-1] du tableau est triée  
    # c'est-à-dire que pour tout i de [1 .. i-1], tab[i-1] <= tab[i]  
    # -> initialisation: i commence à 1 (et pas à 0 car on appelle tab[i-1])  
    for i in range(1, len(tab)):  
        # l'invariant n'est pas maintenu: le tableau entier n'est pas trié  
        if tab[i] < tab[i-1]:  
            return False  
  
    # fin BOUCLE: le tableau entier est trié  
    return True  
  
def verifie_while(tab: list) -> bool:  
    """ Renvoie vrai <=> est trié dans l'ordre croissant  
  
    Tests et exemples:  
>>> verifie_while([0, 5, 8, 8, 9])  
True  
>>> verifie_while([8, 12, 4])  
False  
>>> verifie_while([-1, 4])  
True  
>>> verifie_while([5])  
True  
    """  
    # Cas des tableaux vide et de taille 1: renvoie vrai car triés  
    if len(tab) <= 1:  
        return True  
  
    # BOUCLE:  
    # -> invariant: la zone tab[0 .. i-1] du tableau est triée  
    # c'est-à-dire que pour tout i de [1 .. i-1], tab[i-1] <= tab[i]  
    # -> initialisation: i commence à 1 (et pas à 0 car on appelle tab[i-1])  
    i = 1  
    # -> condition d'arrêt: tout le tableau a été parcouru et i > dernier indice  
    # -> condition d'arrêt: le successeur est inférieur au prédécesseur  
    while not(i == len(tab) or tab[i] < tab[i-1]):  
        i = i+1  
  
    # cas de tout le tableau parcouru  
    if i == len(tab):  
        return True  
    # cas où une partie du tableau n'est pas bien triée  
    else:  
        return False
```

```
# Vérification avec des assertions
assert verifie([0, 5, 8, 8, 9]) == True
assert verifie([8, 12, 4])      == False
assert verifie([-1, 4])         == True
assert verifie([5])             == True

# Vérification avec des affichages
print(verifie([0, 5, 8, 8, 9]))
print(verifie([8, 12, 4]))
print(verifie([-1, 4]))
print(verifie([5]))

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

```

"""
Author: Pascal Padilla
Source: correction de l'exercice 2 du sujet 37 des épreuves pratiques NSI 2022

Remarques:
    * erreur dans l'énoncé:
      page 3, Exemple d'utilisation : `election` doit renvoyer
      {'A': 3, 'B': 4, 'C': 3} ou {'B': 4, 'A': 3, 'C': 3}
"""

urne = ['A', 'A', 'A', 'B', 'C', 'B', 'C', 'B', 'C', 'B']

def depouille(urne):
    # initialisation: un dictionnaire vide
    resultat = {}
    # parcours de chaque élément (appelé `bulletin`) de l'urne
    for bulletin in urne:
        # cas où la clé `bulletin` existe déjà dans le dictionnaire `resultat`
        if bulletin in resultat:
            resultat[bulletin] = resultat[bulletin] + 1
        # cas de la première apparition de la clé `bulletin`
        # initialisation de la clé avec un premier vote
        else:
            resultat[bulletin] = 1

    # fin BOUCLE: toute l'urne est parcourue
    # resultat contient l'ensemble de tous les résultats
    return resultat

def vainqueur(election):
    # BOUCLE:
    # -> invariant: nmax est le nb maxi de bulletins d'un (ou plusieurs) candidats
    #                du tableau `election` parcouru jusqu'à présent

    # initialisation: pas de vainqueur et nmax vaut 0
    vainqueur = ''
    nmax = 0
    # parcours du dictionnaire dépouillé: clé = candidat et valeur = effectifs
    for candidat in election:
        # cas d'un candidat a plus de bulletins que le meilleur jusqu'à présent
        # `candidat` est la clé du dictionnaire et sa valeur est le nb de bulletins
        if election[candidat] > nmax:
            nmax = election[candidat]
            vainqueur = candidat
    liste_finale = [nom for nom in election if election[nom] == nmax]
    return liste_finale

# Vérification par des affichages
election = depouille(urne)
print(election)          # {'B': 4, 'A': 3, 'C': 3}
print(vainqueur(election)) # ['B']

# Vérification avec des assertions
election = depouille(urne)
assert election == {'B': 4, 'A': 3, 'C': 3}
assert vainqueur(election) == ['B']

```