BACCALAUREAT

SESSION 2022

Épreuve de l'enseignement de spécialité

NUMERIQUE et SCIENCES INFORMATIQUES

Partie pratique

Classe Terminale de la voie générale

Sujet n°34

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 `occurrence_max` prenant en paramètres une chaîne de caractères `chaine` et qui renvoie le caractère le plus fréquent de la chaîne. La chaine ne contient que des lettres en minuscules sans accent.

On pourra s'aider du tableau

```
alphabet=['a','b','c','d','e','f','g','h','i','j','k','l','m','n'
,'o,','p','q','r','s','t','u','v','w','x','y','z']
```

et du tableau `occurrence` de 26 éléments où l'on mettra dans `occurrence[i]` le nombre d'apparitions de `alphabet[i]`dans la chaine. Puis on calculera l'indice `k` d'un maximum du tableau `occurrence` et on affichera `alphabet[k]`.

Exemple :

```
>>> ch='je suis en terminale et je passe le bac et je souhaite
poursuivre des etudes pour devenir expert en informatique'
>>> occurrence_max(ch)
'e'
```

EXERCICE 2 (4 points)

On considère une image en 256 niveaux de gris que l'on représente par une grille de nombres, c'est-à-dire une liste composée de sous-listes toutes de longueurs identiques.

La largeur de l'image est donc la longueur d'une sous-liste et la hauteur de l'image est le nombre de sous-listes.

Chaque sous-liste représente une ligne de l'image et chaque élément des sous-listes est un entier compris entre 0 et 255, représentant l'intensité lumineuse du pixel.

Le négatif d'une image est l'image constituée des pixels `x_n` tels que `x_n + x_i = 255` où `x_i` est le pixel correspondant de l'image initiale.

Compléter le programme ci-dessous :

```
def nbLig(image):
    '''renvoie le nombre de lignes de l'image'''
    return ...

def nbCol(image):
    '''renvoie la largeur de l'image'''
    return ...

def negatif(image):
    '''renvoie le négatif de l'image sous la forme
       d'une liste de listes'''
```

```
    L = [[0 for k in range(nbCol(image))] for i in
range(nbLig(image))] # on créé une image de 0 aux mêmes
dimensions que le paramètre image
    for i in range(len(image)):
        for j in range(...):
            L[i][j] = ...
    return L

def binaire(image, seuil):
    '''renvoie une image binarisée de l'image sous la forme
       d'une liste de listes contenant des 0 si la valeur
       du pixel est strictement inférieure au seuil
       et 1 sinon'''
    L = [[0 for k in range(nbCol(image))] for i in
range(nbLig(image))] # on crée une image de 0 aux mêmes
dimensions que le paramètre image
    for i in range(len(image)):
        for j in range(...):
            if image[i][j] < ... :
                L[i][j] = ...
            else:
                L[i][j] = ...
    return L
```

Exemple :

```
>>> img=[[20, 34, 254, 145, 6], [23, 124, 287, 225, 69], [197,
174, 207, 25, 87], [255, 0, 24, 197, 189]]
>>> nbLig(img)
4
>>> nbCol(img)
5
>>> negatif(img)
[[235, 221, 1, 110, 249], [232, 131, -32, 30, 186], [58, 81, 48,
230, 168], [0, 255, 231, 58, 66]]
>>> binaire(img,120)
[[1, 1, 0, 0, 1], [1, 1, 0, 0, 1], [0, 0, 0, 1, 1], [0, 1, 1, 0,
0]]
```

```python
def occurrence_max(chaine):
    alphabet=['a','b','c','d','e','f','g','h','i','j','k','l','m','n','o','p','q','r
','s','t','u','v','w','x','y','z']
    occurrence = [0] * 26

    for lettre in chaine:
        if lettre not in alphabet:
            continue

        i = 0
        while alphabet[i] != lettre:
            i = i + 1
        occurrence[i] = occurrence[i] + 1

    i_max = 0
    v_max = occurrence[0]

    for i in range(26):
        if occurrence[i] > v_max:
            i_max = i
            v_max = occurrence[i]

    return alphabet[i_max]


# version avec dictionnaire
# plus classique
def occurrence_max_dic(chaine):
    occurrence = {}
    for lettre in chaine:
        if lettre == ' ': continue

        if lettre in occurrence:
            occurrence[lettre] += 1
        else:
            occurrence[lettre] = 1

    v_max = -1
    for lettre in occurrence:
        if occurrence[lettre] > v_max:
            lettre_max = lettre
            v_max = occurrence[lettre]

    return lettre_max



ch = 'je suis en terminale et je passe le bac et je souhaite poursuivre des etudes p
our devenir expert en informatique'
assert occurrence_max_dic(ch) == 'e'
from random import randint
alea = ' '.join([chr(randint(97,110)), chr(randint(97,110)), chr(randint(97,110)), c
hr(randint(97,110)), chr(randint(97,110)), chr(randint(97,110)), chr(randint(97,110)
), chr(randint(97,110)), chr(randint(97,110)), chr(randint(97,110)), chr(randint(97,
110)), chr(randint(97,110)), chr(randint(97,110)), chr(randint(97,110)), chr(randint
(97,110)), chr(randint(97,110)), chr(randint(97,110)), chr(randint(97,110)), chr(ran
dint(97,110)), chr(randint(97,110))])
print(alea)
print(occurrence_max(alea))
```

```python
def nbLig(image):
    '''renvoie le nombre de lignes de l'image'''
    return len(image)

def nbCol(image):
    '''renvoie la largeur de l'image'''
    return len(image[0])

def negatif(image):
    '''renvoie le négatif de l'image sous la forme
       d'une liste de listes'''
    L = [[0 for k in range(nbCol(image))] for i in range(nbLig(image))] # on cree un
e image de 0 aux memes dimensions que le parametre image
    for i in range(len(image)):
        for j in range(nbCol(image)):
            L[i][j] = 255 - image[i][j]
    return L

def binaire(image, seuil):
    '''renvoie une image binarisée de l'image sous la forme
       d'une liste de listes contenant des 0 si la valeur
       du pixel est strictement inférieure au seuil
       et 1 sinon'''
    L = [[0 for k in range(nbCol(image))] for i in range(nbLig(image))] # on crée un
e image de 0 aux mêmes dimensions que le paramètre image
    for i in range(len(image)):
        for j in range(nbCol(image)):
            if image[i][j] < seuil :
                L[i][j] = 0
            else:
                L[i][j] = 1
    return L


img=[[20, 34, 254, 145, 6], [23, 124, 287, 225, 69], [197, 174, 207, 25, 87], [255,
0, 24, 197, 189]]
assert nbLig(img) == 4
assert nbCol(img) == 5
assert negatif(img) == [[235, 221, 1, 110, 249], [232, 131, -32, 30, 186], [58, 81,
48, 230, 168], [0, 255, 231, 58, 66]]
assert binaire(img,120) == [[0, 0, 1, 1, 0], [0, 1, 1, 1, 0], [1, 1, 1, 0, 0], [1, 0
, 0, 1, 1]]
```