Concours Blanc

Épreuve d'informatique

Sans calculatrice ni documents

Exercice 1 : Traitement d'image

Une société de graphisme souhaite développer en interne une bibliothèque de fonctions python pour modifier des images aux formats standards.
Les images sont manipulées sous forme de tableau (`array`) à trois dimensions : ligne, colonne, couleur. Les couleurs étant, dans cet ordre, le rouge, le vert et le bleu.
Leur intensité est codée sur un octet comme un nombre entier compris entre 0 et 255.
Un pixel ayant la couleur `[0,0,0]` est noir, un pixel ayant la couleur `[255,255,255]` est blanc.
Certaines de ces fonctions ont déjà été écrites (*cf. annexe en dernière page*), d'autres sont à réaliser.

1. Justifier qu'un nombre entier codé sur un octet peut prendre des valeurs entières de 0 à 255.
2. Commenter la ligne 3 de l'entête.
3. Quelle est la différence entre le fichier `Images.png` et la variable `Image` ?
4. Fonction `flou` : commenter les lignes 10 et 13 de la fonction `flou`
5. fonctions `conv` et `masque` : commenter les lignes 21, 26, 33, 34 et 39.
6. **Propositions de programme**
   Choisir, en fonction du temps disponible et de ses affinités, parmi les propositions suivantes de réalisation de programme.
   - **a.** Proposer une fonction qui renvoie le symétrique de l'image par rapport à son centre de symétrie.
   - **b.** Proposer une fonction qui réalise une rotation de l'image d'un quart de tour dans le sens des aiguilles d'une montre
   - **c.** On considère que la luminosité d'un pixel est la moyenne des nombres codant chacune de ses trois couleurs. Proposer une fonction `nb` qui permet l'affichage de l'image en noir et blanc (*chaque pixel est soit tout noir, soit tout blanc*) en fonction d'un seuil de luminosité. Cette fonction prend en paramètres l'image sous forme de tableau et un seuil de luminosité.
   - **d.** Le rendu de la fonction `nb` ci-dessus est incertain, en fonction de la luminosité de l'image initiale. Afin d'obtenir automatiquement un rendu plus satisfaisant, la société souhaite développer une fonction `nb2` qui appelle la fonction `nb` ci-dessus, et à l'aide d'une boucle conditionnelle, ajuste le seuil de luminosité de sorte qu'en sortie de boucle, l'image en noir et blanc ait « autant » de pixels noirs que de pixels blancs. La fonction `nb2` renverra en sortie l'image en noir et blanc ainsi que le seuil obtenu.
   - **e.** Proposer une fonction `etire` qui « étire » l'image d'un facteur 3 dans le sens « vertical »
   - **f.** Proposer une fonction `PopArt` qui passe l'intensité de chacune des trois couleurs à 0 ou 255 selon que son nombre associé est, respectivement, inférieur ou supérieur à 128.

Exercice 2 : Manipulation de chaînes de caractères, tri

Néo est un dissident dans un pays sous dictature.
Il souhaite échanger avec des correspondants via internet et sait que non seulement ses messages sont interceptés, mais qu'en outre les algorithmes informatiques de cryptage à sa disposition sont contrôlés par la dictature qui peut obtenir le message non crypté à partir du message qu'il aurait crypté par un tel procédé.

Il décide de créer lui même des algorithmes de cryptage de ses messages.

1. **Premier algorithme**
   Néo pense d'abord au code dit *« César »*, qui consiste à décaler toutes les lettres du texte d'un même déplacement dans l'alphabet.
   Par exemple si on déplace de +2 les lettres du texte `essai` on obtient `guuck`.
   - **a.** Donner le texte crypté avec le code *César* du mot `bonjour` avec un déplacement de +3.
   - **b.** *Pour enregistrer et manipuler du texte en informatique, on associe conventionnellement à 256 symboles usuels (caractères de l'alphabet en majuscule et en minuscule, virgule, point, espace, ...) un nombre entier unique compris entre 0 et 255 (on établit une bijection entre ces 256 symboles et l'ensemble* $[\![0, 255]\!]$*).*
     La fonction Python `ord()` prend comme argument un symbole usuel de texte et renvoie le nombre qui lui est associé. La fonction `chr()` réalise la bijection réciproque de `ord()`.
     Néo a réalisé la fonction `decaler()` donnée en annexe. Décrire le fonctionnement de cet algorithme.
   - **c.** Proposer un programme qui réalise la même fonction mais en utilisant l'instruction `while` plutôt que `for`, et qui teste la longueur de la chaîne `crypt`.
   - **d.** Si la somme entre le nombre associé à un caractère et le déplacement du cryptage dépasse 255, le nombre obtenu pour le caractère codé ne correspond plus à un caractère. Dans ce cas, Néo décide d'enlever 256 au nombre obtenu. Proposer un ajout dans le code du programme initial pour traiter ce cas de figure.
     (On écrira sur sa copie la ligne du programme précédent l'ajout pour repérer l'endroit de son insertion)

2. **Deuxième algorithme**
   Néo se dit que l'algorithme `decaler` est aisément déchiffrable. Il décide d'appliquer d'autres transformations à un texte à coder et crée la fonction `coupure()` donnée en annexe.
   - **a.** Donner le résultat renvoyé par l'instruction : `coupure('essai',3)`.
   - **b.** Que réalise cet algorithme ?

3. **Troisième algorithme**
   Proposer un algorithme `combi` sous forme d'une fonction qui admet comme paramètre la chaîne `texte` à crypter et qui appelle les fonctions `decaler` et `coupure`.
   On souhaite que cet algorithme `combi` réalise un cryptage difficile à déchiffrer. Cette fonction `combi` pourra prendre d'autres paramètres.

4. **Quatrième et cinquième algorithme**
   Néo souhaite pouvoir déterminer si un caractère donné est présent dans un texte.
   Pour cela il réalise deux fonctions Python.
   La première s'appelle `tri()` et est donnée en annexe.

a. Décrire ce que fait la fonction `tri()`.

b. La seconde s'appelle `present()`, prend comme paramètres la chaîne `texte` renvoyée après la fonction `tri()` ainsi que le caractère à chercher. A l'aide d'un algorithme de dichotomie elle teste si le caractère est présent dans le texte soumis.
Proposer un code pour la fonction `present`.

Annexe exercice 2

```
def decaler(texte,decalage):
    longueur=len(texte)
    crypt=''
    for n in range(longueur):
        caractere=texte[n]
        code=ord(caractere)
        code=code+decalage
        crypt=crypt+chr(code)
    return(crypt)

def coupure(texte,endroit):
    premier_bout=texte[:endroit]
    deuxieme_bout=texte[endroit:]
    crypt=deuxieme_bout+premier_bout
    return(crypt)

def tri(texte):
    longueur=len(texte)
    fini=False
    while fini==False:
        fini=True
        for n in range(longueur-1):
            caractere1=texte[n]
            caractere2=texte[n+1]
            if ord(caractere1)>ord(caractere2):
                fini=False
                texte1=texte[:n]
                texte2=texte[n+2:]
                texte=texte1+caractere2+caractere1+texte2
                print(texte)
    return(texte)
```

Annexe exercice 1

```python
import numpy as np
import matplotlib
import matplotlib.pyplot as plt
Image=matplotlib.image.imread("Image.png")

def flou(im):
    n, p, s = im.shape
    flou= np.zeros_like(im)
    for i in range(1,n-1):
        for j in range(1,p-1):
            r,v,b=0,0,0
            for n in range(-1,2):
                for m in range(-1,2) :
                    r=r+im[i+n,j+m,0]
                    v=v+im[i+n,j+m,1]
                    b=b+im[i+n,j+m,2]
            flou[i, j] =[r/9,v/9,b/9]
    return plt.imshow(flou)

def masque(x):
    taille = 2*x+1
    masque=np.zeros((taille,taille))
    for n in range(taille):
        for m in range(taille):
            print('ligne',n,'colonne',m,' : ')
            masque[n,m]=input()
    return masque

def conv(im,x):
    n, p, s = im.shape
    ma=masque(x)
    print('Traitement en cours...')
    conv= np.zeros_like(im)
    for i in range(x,n-x):
        for j in range(x,p-x):
            r,v,b=0,0,0
            for n in range(-x,x+1):
                for m in range(-x,x+1) :
                    r=r+im[i+n,j+m,0]*ma[n,m]
                    v=v+im[i+n,j+m,1]*ma[n,m]
                    b=b+im[i+n,j+m,2]*ma[n,m]
            conv[i, j] =[r,v,b]
    return plt.imshow(conv)
```