LES FACULTÉS
DE L'UNIVERSITÉ
CATHOLIQUE DE LILLE

TP N°8 :

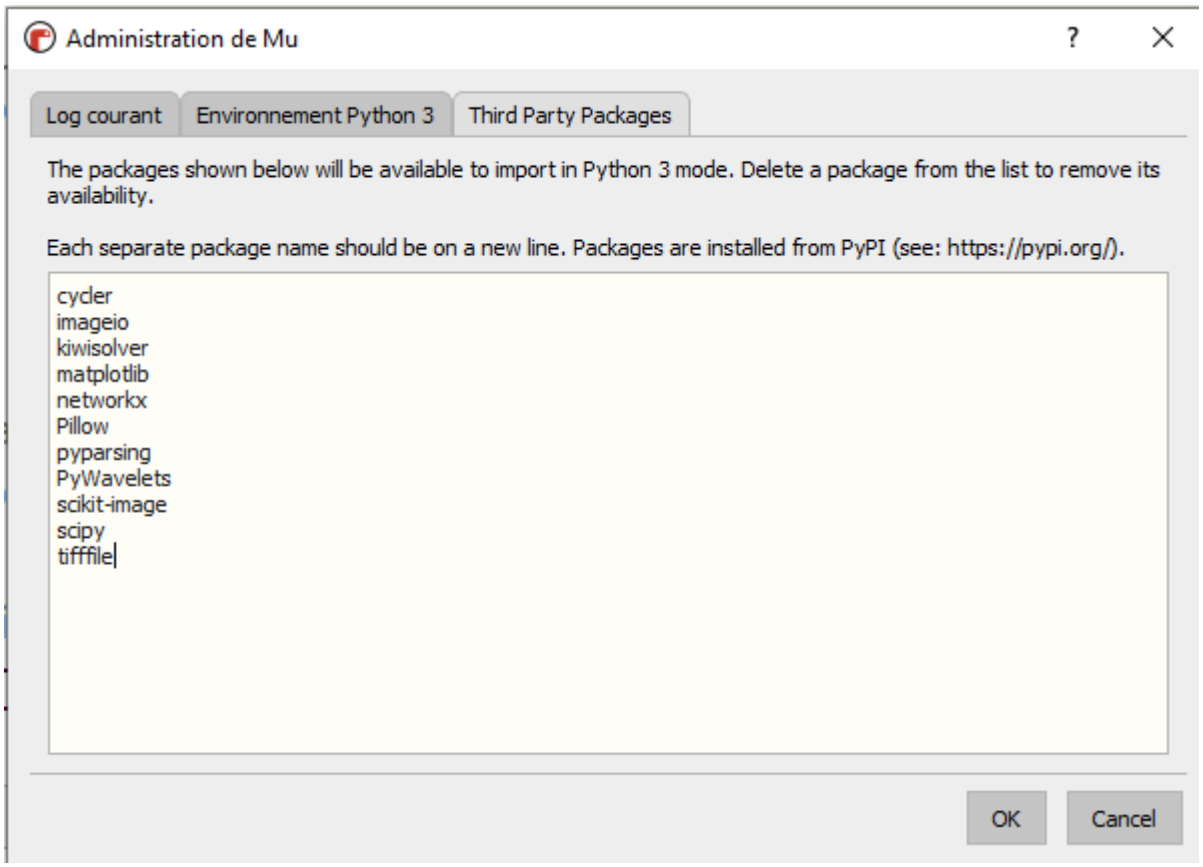
Programmation en Python

Traitements d'images

SOMMAIRE

REMARQUES TECHNIQUES

https://codewith.mu/en/tutorials/1.1/pypi :



installer PIL et numpy sur VS :

```
pip install pillow
pip install numpy
```

I. OUVERTURE ET CARACTÉRISTIQUES D'UNE IMAGE

1. Ouverture d'une image

Tâche N°1. Complétez les commentaires (_____) et faites fonctionner ce code :

```
from PIL import Image #importation du sous-module Image du module PIL
import numpy as np

im=Image.open("mina.png") #ouverture d'une image au format png dans Python.

#Mise dans un tableau
tab=np.array(im)
print(tab)

#im.size renvoie ____________________________________________
print('La taille est ',im.size)

#nb de sous-tableaux de tab, c'est-à-dire nombre de ____________________
print('Nb lignes =',len(tab))

#nb de sous-sous-tableaux de tab, c'est-à-dire nombre de _________
print('Nb de col=',len(tab[0]))

#nb de couleurs additives utilisées, ici _____ : R, _______
print('Nb de coul=',len(tab[0][0]))

print(tab.shape) #renvoie un tuple contenant les éléments précédents (h, l, 3)
nouvelle_image=Image.fromarray(tab)

nouvelle_image.show() # pour _______ l'image
nouvelle_image.save("nom_de_la_nouvelle_image.png")
# pour _______ au format voulu
```

Tâche N°2. Expliquez l'affichage et notamment cette matrice :

```
  [193  97  96]
  [194  94  99]
  [192  97  99]]
```

Tâche N°3. Qu'est-ce qu'un pixel ?

Tâche N°4. Expliquez cette ligne :

```
Hauteur, Longueur, NbCol = tab.shape
```

2. A vous de jouer

Tâche N°5. En utilisant la ligne précédente, remplacer les lignes de codes où apparait la fonction len()

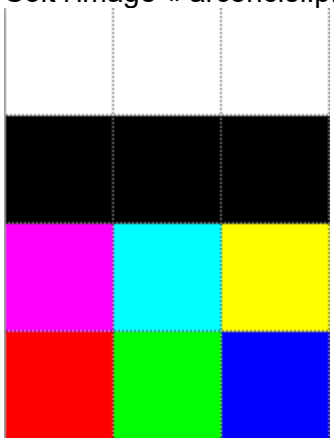
3. Les composantes RGB de chaque pixel

Chaque pixel commençant par celui en bas à gauche défini par un groupe de 3 octets représentant respectivement ses teintes Bleue, Verte et Rouge.

Tâche N°6. A l'aide d'Internet ou du logiciel pain, remplissez ce tableau en donnant les trois valeurs RGB pour chaque couleur

	Valeur de la composante entre 0 et 255		
	Rouge	Vert	Bleu
Rouge			
Vert			
Bleu			
Jaune			
Noir			
Blanc			

Soit l'image « arcenciel.png » suivante :



Tâche N°7. Corrigez l'affichage du code suivant :

```
from PIL import Image #importation du sous-module Image du module PIL
import numpy as np

im=Image.open("arcenciel.png") #ouverture d'une image au format png dans Python.

#Mise dans un tableau
tab=np.array(im)
NbLig, NbCol,NbCoul = tab.shape

print("La dimension de l'image en pixels est de ",NbLig," lignes sur ",NbCol," colonnes")
print('Et le nombre de composantes par pixel est ',NbCoul)
print('Le premier pixel vaut ',tab[0][0],' et correspondant à celui en bas à gauche')
print('Le dernier pixel vaut ',tab[NbLig-1][NbCol-
1],' et correspondant à celui en haut à droite')

im.close()
```

Soit le programme suivant :

```
from PIL import Image #importation du sous-module Image du module PIL
import numpy as np

im=Image.open("arcenciel.png") #ouverture d'une image au format png dans Python.

#Mise dans un tableau
tab=np.array(im)
NbLig, NbCol,NbCoul = tab.shape

print('Le premier pixel vaut R=',tab[0][0][0],' G=',tab[0][0][1],' B=',tab[0][0][2],)
for ligne in range(0,NbLig):
```

```
    for colonne in range(0,NbCol):
        print('Le pixel N°',ligne*NbCol+colonne+1,'est',tab[ligne] [colonne]),
im.close()
```

Tâche N°8. Modifiez le code afin d'afficher pour tous les pixels, ses trois composantes RGB. Pour rappel, le pixel est représenté par un 'tableau' à trois valeurs.

tab[0][0][0] est un tableau à trois dimensions. Comme il est du type numpy, il est possible de l'écrire tab[0,0,0]

Tâche N°9. Remplacez tab[0][0][0] et tab[ligne][colonne] dans le code précédent

II. QUELQUES TRANSFORMATIONS DE BASE

Soit le code suivant qui une image :

```
from PIL import Image #importation du sous-module Image du module PIL
import numpy as np

im=Image.open("mina.png") #ouverture d'une image au format png dans Python.

#Mise dans un tableau
tab=np.array(im)
nouvelle_image=Image.fromarray(tab)
nouvelle_image.show() # pour afficher l'image
nouvelle_image.save("nom_de_la_nouvelle_image.png")
# pour l'enregistrer au format voulu
```

Vous remarquez que c'est le tableau qui est copié. Donc pour modifier une image il suffit de modifier le tableau de pixels

1. Niveau de gris - version moyenne -

Tâche N°10. Dans cette première version de passage en niveau de gris, se fait par le calcul de la moyenne des composantes RGB.
Testez votre code sur le chiot et n'oubliez pas d'afficher la nouvelle image en niveau de gris.
Utilisez le code de la tâche précédente ainsi que le tout premier.

Remarque :

Tâche N°11. Si vous avez une « overflow encountered in ubyte_scalars » en sachant que R,G,B sont codés sur 8 bis, essayez de l'expliquer.
L'image est-elle réellement en niveaux de gris ?

Deux solutions

- forcer la valeur en valeur entière codée sur 64 bits avec la fonction int()
- diviser la valeur R, G ou B directement par 3 et ajouter ces trois divisions

Tâche N°12. Essayez une des deux solutions

1. Niveau de gris version norme 709

Dans sa norme 709, elle dit que pour les images naturelles les poids respectifs doivent être 0.2125 * R + 0.7154 * G + 0.0721 * B

Explication : la forte pondération sur le vert est due au fait que l'œil humain est plus sensible à cette couleur, et donc que les détails (polypes potentiels) seront plus visibles avec cette forte pondération

Tâche N°13. En utilisant la norme 709, créez un programme qui met en niveau de gris une image.
Comparez cette dernière image avec le résultat du niveau de gris par moyenne.

2. Seuillage en noir et blanc

Ce seuillage permet d'obtenir une image en noir et blanc. Pour cela, il faut prendre une valeur de seuil, par exemple 127, puis si une des composantes dépasse ce seuil c'est tout le pixel qui est blanc [255,255,255]. Dans le cas contraire il est noir [0,0,0]

Tâche N°14. Codez ce seuillage noir et blanc et testez plusieurs valeurs de seuil

3. Image rouge : garder la composante rouge uniquement

Tâche N°15. Tout est dans le titre. Il faut mettre les autres composants à zéro.
Tâche N°16. Créez trois fonctions : une qui garde le rouge, une autre le vert et la dernière le bleu

4. En négatif – inversion des couleurs

Tâche N°17. Cette inversion se fait en remplaçant les composantes du pixel (R,G,B) par (255-R,255-G,255-B)