La photo numérique avec Python

Visualiser des transformations d’images avec Python,

en utilisant la bibliothèque Pillow

Niveau : Secondes ICN, Approfondissement en AP-TS ou début Spécialité ISN

Introduction : Pillow est une bibliothèque Python qui permet de travailler sur les images.

Version Python 3.2.3

<https://www.python.org/downloads/windows/>

Version Pillow 3.1.1

<https://pypi.python.org/pypi/Pillow/3.1.1>

Ouvrir l’IDLE (environnement de développement intégré), une fenêtre s’ouvre, celle où s’exécutera le programme. (shell)

Ouvrir une « nouvelle fenêtre » pour écrire le programme (éditeur)

Dans cette activité, nous allons travailler avec l’image ci-dessous « IMAGE.jpg ». La Copier dans un répertoire « Python »



Le répertoire images&programmes\_py contient l’ensemble des programmes présentés dans cette fiche ainsi que les images résultant des transformations opérées.

Le dossier associé à cette activité images&programmes\_py.zip doit être téléchargé et décompressé.

**I Quelques rappels sur les images :**

* Une image est constituée de **pixels** (points), la **définition** d'une image L x H (par exemple 330 x 330 pour l'image considérée correspond au nombre de pixels de l'image (L : la largeur de l'image en pixel, H : la hauteur de l'image en pixel).
* Chaque pixel est constitué de 3 éléments (chacun est appelé aussi **canal**) : un rouge, un vert et un bleu. C'est la somme de ces 3 couleurs qui permet d'obtenir un grand nombre de couleurs (synthèse additive).
* Classiquement, à chaque canal, on associe un nombre binaire codé sur 8 bits (soit donc 24 bits par pixel).
* Pour chaque pixel, on aura donc une valeur pour le rouge (comprise entre 0 et 255 puisque codé sur 8 bits), une valeur pour le vert (comprise entre 0 et 255) et une valeur pour le bleu (comprise entre 0 et 255). Quelques exemples : (0,0,0) => noir ; (255,0,0) => rouge ; (255,255,255) => blanc ; (0,255,0) => vert...

N.B. Il peut exister un 4e canal en plus du canal rouge, canal vert et canal bleu, le **canal "alpha"** qui permet de gérer la transparence du pixel (par exemple, les images au format .png gèrent la transparence). Chaque pixel est donc codé sur 32 bits (4 x 8).

1) Partant du principe que l'image que vous avez copiée possède une définition de 330\*330, que chaque pixel est constitué de 3 éléments (RVB) chacun codé sur 8 bits = 1 octet, calculez la taille (ou poids) de cette image en kilo octet (ko).

Rappel : 1 ko = 1024 octets ; 1 octet = 8 bits

Si vous regardez la taille réelle de l'image en ko, vous allez sans doute trouver une valeur inférieure à celle que vous venez de calculer. Ceci est tout à fait normal, car le .jpeg est un format compressé (le taux de compression est compris entre 10 : 1 et 25 : 1).

Vous pouvez convertir cette image en .PNG avec « paint » ou « photofiltre » et comparer sa nouvelle taille à la taille précédente.

**II Fonctions de base**

1) La méthode **open** vous permet d'ouvrir un fichier image.

img=open("image.jpg")

Le paramètre correspond au nom du fichier image (attention de bien renseigner le bon chemin si votre fichier image et votre fichier "python" ne se trouvent pas dans le même dossier).

2) Saisissez, analysez et testez ce code :

#!/usr/bin/env python

# -\*- coding: utf-8 -\*-

from PIL.Image import \*

img=open("image.jpg")

img.save("resultat.jpg","JPEG")

Sauvegarder ce fichier dans le répertoire « python » où vous avez collé l’image. ( en . py).

La troisième ligne du programme proposé, permet d'importer la bibliothèque Pillow.

La ligne "img.save("resultat.jpg","JPEG")" permet de sauvegarder une image. L'image sera sauvegardée dans le dossier courant (le même dossier que "image.jpg") et aura pour nom "resultat.jpg".

N.B Dans ce cas précis, il est totalement inutile de sauvegarder l'image puisque nous ne l'avons pas modifiée. L'idée est juste de vous montrer l'utilisation de "img.save".

3) La méthode getpixel((x,y)) renvoie un **tuple** (un tuple ressemble beaucoup à une liste, à l'exception près, qu'un tuple, une fois créé, n'est pas modifiable.) contenant les trois composantes : rouge, verte et bleue du pixel situé aux coordonnées x, y (sachant que le pixel de coordonnées 0, 0 se situe en haut à gauche de l'image).

Saisissez, analysez et testez ce code :

#!/usr/bin/env python

# -\*- coding: utf-8 -\*-

from PIL.Image import \*

img=open("image.jpg")

(r,v,b)=img.getpixel((100,100))

print(r,v,b)

Si tout se passe correctement, vous devriez avoir à l'écran (dans la console) : (251, 139, 1). Nous pouvons donc affirmer que le pixel de l'image "image.jpg", ayant pour coordonnées (100, 100), a pour composantes : Rouge=>251 ; Vert=>139 ; Bleu=>1.

Encore une chose importante sur les tuples : si vous voulez récupérer une valeur du tuple, cela se passe exactement comme avec les listes. Si vous reprenez l'exemple ci-dessus, tu [0] est égal à 251 et tu [2] est égal à 1.

Vous remarquerez que la méthode getpixel prend en paramètre un tuple : (100,100).

4) Autre fonction très utile : **putpixel**. Cette méthode permet de modifier les canaux Rouge, Vert et Bleu d'un pixel.

La méthode putpixel prend deux paramètres (2 tuples) : le premier paramètre correspond aux coordonnées du pixel que nous voulons modifier (tuple avec 2 éléments : (x, y)). Le second paramètre correspond aux valeurs à attribuer aux canaux R, V, B (tuple avec 3 éléments : (255,255,255), ici un pixel blanc).

Saisissez, analysez et testez ce code :

#!/usr/bin/env python

# -\*- coding: utf-8 -\*-

from PIL.Image import \*

img=open("image.jpg")

img.putpixel((100,100),(0,0,255))

img.save("resultat.jpg","JPEG")

Cherchez maintenant le pixel bleu sur l’image.

5) Récupérer la définition d’une photo

(L,H)= img.size

6) Redimensionner une photo

img=img.resize((L,H))

L et H étant les nouvelles dimensions de « img »

#### III Modification d’une photo

1) Écrire et commenter le programme permettant de **"colorier" le blanc en noir** (tous les pixels blancs devront être remplacés par des pixels noirs).

N.B. Le résultat ne sera pas parfait, car de nombreux pixels paraissant blancs ne sont, en fait, pas vraiment blancs.

2) Écrire et commenter un programme en Python qui permettra de **transformer une image couleur en son négatif**.

3) Écrire et commenter un programme en Python qui permettra de **transformer une image couleur en une image en niveau de gris** (souvent improprement appelée "noir et blanc", car une image "noir et blanc" est uniquement composée de pixel noir et de pixel blanc).

 Une des façons d’obtenir du gris est de prendre les fractions suivantes pour les composantes RVB : (rouge=0.299 ; verte=0.587 ; bleue=0.144), la somme faisant 1. Remarque : en Python, int(x) permet de prendre la valeur entière de x.

#### IV Propositions de mini-projets

#### Filtres : Récupérer chacune des 3 composantes et fabriquer 3 images différentes (chacune représentant le niveau d’intensité d’une des trois couleurs)

#### Augmenter le contraste

#### Faire pivoter une image de 180°

#### Faire pivoter une image de 90°

#### Transformer l’image avec un miroir horizontal ou vertical

#### Remplacer l’image par une mosaïque dont la dimension des carreaux est paramétrable

#### Corriger un pixel

* **Faire l’histogramme simplifié d'une photo en niveau de gris**.

​Avec le logiciel Adobe Photoshop Lightroom par exemple, les tons sont découpés selon les valeurs suivantes :

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Noir** : de 0 à 25 | **Tons foncés** : de 26 à 63 | **Exposition** : de 64 à 178 | **Tons clairs** : de 179 à 229 | **Blanc** : de 230 à 255 |

Vous pouvez faire un programme qui donne pour une image, le nombre de pixels dans chacune de ces 5 zones puis le pourcentage correspondant.

#### Codage d’un texte dans une image et décodage

#### Brouillage d’une image par rotation

Voici deux liens si vous désirez poursuivre votre découverte de Pillow :

* La documentation [en anglais] : <http://pillow.readthedocs.org/>
* Une série de vidéos [en anglais] : <https://www.youtube.com/watch?v=_H9uPRJWMNk>