

# Manipulation d'images avec Python

## A propos des images :

### Les pixels dans une image

Une image informatisée est discrétisée en **pixels** (« picture elements »), c'est-à-dire en (petites) zones carrées. Chaque pixel possède une couleur (ou une nuance de gris). L'ensemble des pixels est organisé sous forme d'un tableau bidimensionnel dans lequel chacun d'eux peut être repéré par un numéro de lignes  $x$  et un numéro de colonnes  $y$ . Le couple  $(x,y)$  forme les coordonnées du pixel dans l'image.

Ce nombre de pixels est souvent mis en avant comme un critère de qualité d'un appareil photo numérique.

Le nombre de lignes et de colonnes donnent la dimension de l'image, et le produit de ces deux nombres est égal au nombre de pixels contenus dans l'image.

### Le codage RVB






La couleur d'un pixel peut être décrite de plusieurs façons par un ou plusieurs nombres. Un codage courant est le **codage RVB**. Une couleur peut être désignée par un triplet RVB (rouge, vert, bleu ; on utilise RGB en anglais). On précise une intensité, de 0 % à 100 % pour chacune des composantes. La couleur résultante est obtenue par synthèse additive des trois composantes.

Pour une intensité nulle des trois composantes, on obtient du noir. Pour une intensité maximale des trois composantes, on obtient du blanc. Lorsque les trois intensités sont égales on a une nuance de gris.

L'intensité de chacune des composantes peut être exprimée sur une échelle qui va de 0 à 255. Huit bits, soit un octet, sont nécessaires pour représenter une telle valeur :  $256 = 2^8$ . On utilise donc commodément une notation hexadécimale de deux chiffres entre  $00_{16}$  et  $FF_{16}$ .

Une couleur est désignée par la concaténation de trois valeurs hexadécimales de deux chiffres, une pour chacune des trois composantes, dans l'ordre rouge, vert, bleu. On pourra donc écrire  $000000_{16}$  pour noir,  $FFFFFF_{16}$  pour blanc, ou  $FF0000_{16}$  pour rouge.

### Quelques formats de fichiers

| Format                          |   | Nombre de couleurs          | Compression                     | Transparence       |
|---------------------------------|---|-----------------------------|---------------------------------|--------------------|
| Bitmap                          |  | 16 millions                 | Non                             | Non                |
| Joint Photographic Expert Group |  | 16 millions et +            | Oui, réglable avec pertes       | Oui                |
| Graphics Interchange Format     |  | 256                         | Oui, sans perte                 | Oui                |
| Portable Network Graphics       |  | 16 millions                 | Oui, sans perte                 | Oui (couche alpha) |
| Tagged Image File Format        |  | De monochrome à 16 millions | Oui ou non, avec ou sans pertes | Oui (couche alpha) |

Dans ce projet, il est recommandé de travailler avec des fichiers bitmap bmp pour plus de clarté.

## Le décryptage d'une image en python

Voici un exemple de code python permettant de lire les pixels d'une image, de les modifier, et de les enregistrer dans une nouvelle image.

```
from PIL import Image

image1 = Image.open(".\christmas.png")
L, H = image1.size
image2 = Image.new("RGB", (L,H))

for y in range(H):
    for x in range(L):
        pixel = image1.getpixel((x,y))
        r = pixel[0]
        v = pixel[1]
        b = pixel[2]
        g = int( (r+v+b) /3 )
        image2.putpixel((x,y), (g,g,g))

image2.save(".\christmas_gris.png")
image2.show()
```

### 1. Comprendre le code

Examinez le code ci-dessus. Recopiez-le dans Edupython, et mettez des commentaires pour chacune des lignes de ce programme. Ajouter des commentaires au début du programme pour expliquer le fonctionnement et le but de ce programme. Sauvegardez le programme dans votre répertoire personnel.

### 2. Exécuter le code

Récupérer le fichier image « christmas.png » situé dans le répertoire de classe et le copier dans votre répertoire personnel (au même endroit que le fichier python précédent). Exécuter le code et vérifier son fonctionnement. Tester le programme avec un fichier différent que vous récupèrerez sur internet (utiliser des images libres de droit).

### 3. Comprendre l'algorithme

Compléter l'algorithme présenté en Annexe, correspondant au code python ci-dessus.

## Objectif :

Créer un programme Python permettant d'effectuer une ou plusieurs transformations sur une image numérique. Le programme, disposant d'une interface graphique, permet de reproduire une fonctionnalité existante d'un logiciel de manipulation d'image (GIMP, PhotoShop, Paint.NET, etc.) ou d'en inventer une nouvelle (mieux).

## Consignes

Les élèves s'associent en trinômes pour réaliser le projet. Le cahier des charges de base est présenté ci-dessous :

- l'ensemble des programmes sont écrits en langage Python
- les programmes doivent utiliser une ou plusieurs fonctions
- les programmes doivent être proprement commentés
- au lancement du programme, un menu graphique apparaît. Ce menu est utilisable soit avec la souris, soit avec le clavier et comporte au moins les éléments suivants :
  - o bouton de sortie du programme : lors de la sélection de ce bouton, le programme se ferme
  - o bouton de sélection de l'image : lors de la sélection de ce bouton, une fenêtre contextuelle s'ouvre et permet de sélectionner une image
  - o bouton(s) de modification(s) : ce sont les boutons permettant d'effectuer la ou les modifications programmées ; à la suite de l'appui sur ce bouton, une fenêtre s'ouvrira affichant l'image modifiée.
- Les images à modifier sont choisies sur le thème de l'astronomie, et on utilisera des images libres de droit pour faire la démonstration du logiciel.
- L'algorithme du programme de manipulation de l'image devra être créé.

## Mise en œuvre :

On travaillera sur ordinateur avec des logiciels interpréteurs de python. Pour le menu graphique, on pourra utiliser le module tkinter (à installer dans Edupython). Parmi les tâches à se répartir : imaginer l'interface graphique, concevoir l'algorithme, programmer les fonctions, réunir les codes, préparer le document, récupérer les images de test, etc...

## Travail à rendre :

Le travail est rendu sous forme d'archive au format zip. L'archive devra contenir :

- Un **document** au format Word (.docx) décrivant le programme et ses fonctionnalités, la répartition des tâches entre les membres de l'équipe, l'algorithme sous forme de diagramme, des exemples de capture d'écran de transformations d'images
- Les différents **programmes python** et leurs commentaires
- La ou les **image(s)** nécessaires au fonctionnement du programme

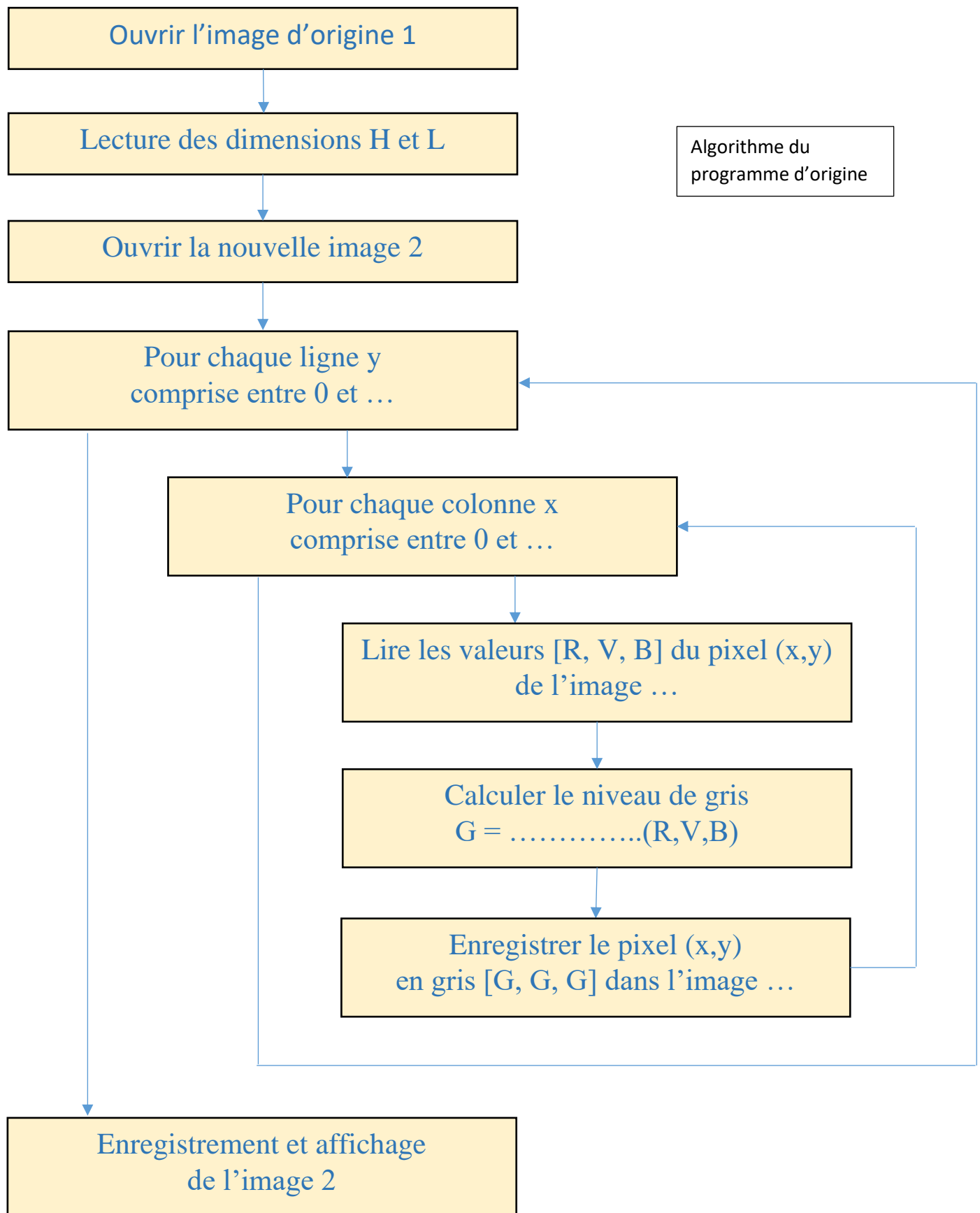
## Durée envisagée :

Le travail est à rendre pour le 5 mars 2023 dernier délai, la capacité d'organisation et la gestion du calendrier seront pris en compte dans l'évaluation.

## Annexe : le module tkinter

Le module tkinter est largement présenté sur cette [page](#).

## Annexe : algorithme



## Grille d'évaluation :

| Item  | Compétence évaluée  | Niveau   | Indicateur de réussite   |
|---|---|----------|--|
| <b>Collaboration :</b><br>- tâches réparties équitablement, mais en respectant les compétences de chacun<br>- communication pertinente ayant permis de résoudre les difficultés                           | <b>Coopérer</b> au sein d'une équipe dans le cadre d'un projet  | <b>A</b> | Les tâches identifiées ont été réparties sur l'ensemble des participants ; les participants sont restés en communication permanente ce qui a permis de résoudre les difficultés rencontrées par certains.                        |
|   |   | <b>B</b> | La répartition des tâches semble déséquilibrée, un ou deux élèves se sont chargés de l'essentiel du travail / certains élèves n'ont pu résoudre certains problèmes et n'ont bénéficié d'aucune aide                              |
|   |   | <b>C</b> | Un membre de l'équipe a eu du mal à s'intégrer, et a été chargé des tâches les plus ingrates, comme la rédaction des documents de présentation / le manque de communication n'a pas permis de résoudre de nombreux problèmes.    |
|   |   | <b>D</b> | Un participant n'a quasiment pas effectué de tâches, les participants ne sont pas parvenus à s'entendre.   |
| <b>Interface graphique :</b><br>- Utilisation correcte du module tkinter<br>- interface fonctionnelle   | Traduire un algorithme dans un <b>langage de programmation</b> , en spécifier les interfaces et les interactions  | <b>A</b> | L'interface graphique correspond aux spécifications demandées et permet parfaitement de faire fonctionner le programme – le module est bien utilisé et les codes proprement commentés  |
|   |   | <b>B</b> | L'interface a de légers dysfonctionnements ou défauts d'ajustement, mais permet de faire fonctionner le programme – la majorité du code est commentée  |
|   |   | <b>C</b> | L'interface graphique n'est pas aboutie – des défauts importants de fonctionnement ou d'ergonomie sont constatés – les codes sont peu commentés  |
|   |   | <b>D</b> | L'interface graphique est très superficielle voire inexistante   |
| <b>Aboutissement du projet : le fonctionnement</b><br>- conformité aux spécifications<br>- gestion des débordements<br>- fonctionnement final<br>- résolution des dysfonctionnements<br>- codes commentés | Traduire un algorithme dans un <b>langage de programmation</b> , en spécifier les interfaces et les interactions, comprendre et réutiliser des codes sources existants, développer des processus de mise au point et de validation des programmes | <b>A</b> | Le projet est totalement abouti, les fonctionnalités sont conformes aux spécifications, aucun dysfonctionnement n'est visible – les codes sont proprement commentés  |
|   |   | <b>B</b> | Le projet est globalement conforme aux spécifications initiales, mais une ou deux fonctionnalités n'ont pas pu être finalisées / des dysfonctionnements n'ont pas pu être résolus  |
|   |   | <b>C</b> | Le projet est moyennement fonctionnel, plusieurs fonctionnalités sont absentes ou dysfonctionnent, l'ensemble s'écroule au bout de quelques utilisations   |
|   |   | <b>D</b> | Le projet n'est pas abouti, il ne fonctionne pas.  |
| <b>Algorithme de manipulation d'image :</b><br>- l'algorithme fourni est cohérent et vérifié<br>- le programme Python correspond à l'algorithme présenté<br>- complexité de la fonctionnalité présentée   | Concevoir des solutions <b>algorithmiques</b>   | <b>A</b> | La fonctionnalité présentée est complexe – l'algorithme et le programme python sont en parfaite cohérence  |
|   |   | <b>B</b> | La fonctionnalité présentée est de complexité raisonnable – l'algorithme et le programme sont globalement cohérents  |
|   |   | <b>C</b> | La fonctionnalité présentée est plutôt simple – l'algorithme a été étudié mais ne correspond pas bien au programme réellement proposé / OU / La fonctionnalité est simple mais l'algorithme et le programme collent parfaitement |
|   |   | <b>D</b> | La fonctionnalité est basique – le programme ne correspond pas à l'algorithme / pas d'algorithme   |