Boucles Imbriquées : ASCII Art

Pour faire ce TP on utilisera Spyder (Documentation Spyder). On utilisera PythonTutor pour tester les fonctions et observer la mémoire.

On téléchargera l'archive ressources_activite_6.zip et on l'extraira dans le dossier de votre choix sur l'ordinateur (Extraire un fichier sous Windows 10). Vous ouvrirez dans Spyder le fichier `activite_6.py` contenu dans l'archive.

Les exercices de cette activité permettent de manipuler des images en niveau de gris, voilà quelques sorties des exercices :

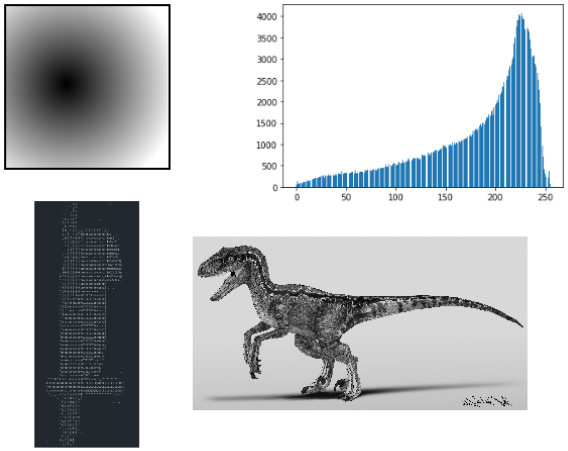

FIGURE 1 – Rendus des exercices

1 Images et Matrices

Lire la section sur les matrices dans le cours .
Lire la section sur les modules dans le cours .

Une image en niveau de gris de dimension $l \times h$, où l est la longueur de l'image et h la hauteur, est représentée par une matrice $h \times l$ à valeurs entières entre 0 (noir) et 255 (blanc). La valeur à l'indice `i,j` de la matrice correspond au niveau de gris du pixel de coordonnées `(j,i)`.

Les coordonnées des pixels sont obtenus suivant le schéma ci-dessous :

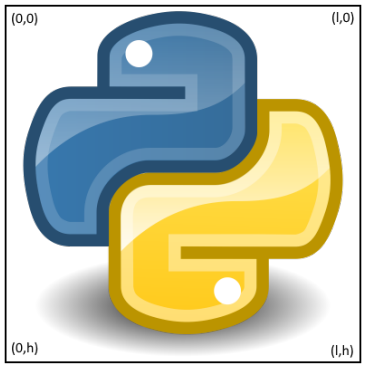

FIGURE 2 – Coordonnées des pixels dans l'image

Les images sont manipulées grâce à l'objet `Image` du module `PIL` et au module `numpy`, nommé `np` dans les exemples ci-dessous. Voici quelques instructions utiles pour manipuler les images.

- `img = Image.open(r"chemin/vers/image.jpg")` renvoie un objet `Image` contenant toutes les informations de l'image `image.jpg` dont on a précisé le chemin en argument, puis l'affecte à la variable `img`.
- `img_gris = img.convert("L")` renvoie un nouvel objet `Image` contenant l'objet image `img` convertie en niveau de gris, puis l'affecte à la variable `img_gris`.
- `np.asarray(img).tolist()` renvoie une *matrice* contenant les niveaux de gris de chaque pixel de l'image `img`.
- `Image.fromarray(np.array(matrice, dtype="uint8"),mode="L")` renvoie un objet image à partir d'une matrice de valeurs entière entre `0` et `255`.
- Pour afficher une image plusieurs solution. Si `img` contient un objet image, alors
  - taper `img` dans la console Spyder devrait afficher l'image dans la console. Attention à ne pas le faire sur des images trop grosses.
  - taper `display(img)` dans la console Spyder devrait afficher l'image dans le panneau réservé à l'affichage graphique de Spyder. Attention à ne pas le faire sur des images trop grosses.
  - taper `img.show()` dans la console Spyder devrait afficher l'image dans le logiciel de visualisation d'image par défaut de votre système. A privilégier pour les grosses images.
- `img.save(r"chemin/vers/image.jpg")` enregistre l'image `img` dans le fichier `image.jpg`.
- `img.close()` permet de fermer le fichier utilisé pour créer l'objet image associé à `img`.

Exercice 1. *Importation de Module*

En début de fichier :

1. Importez le module `numpy` en le renommant `np`.
2. Importez le module `matplotlib.pyplot` en le renommant `plt`
3. Importez l'objet `Image` du module `PIL` sans importer tout le module `PIL`.

Exercice 2. *Image vers Matrice*

Implémentez une fonction `image2matrice` qui prend en entrée un chemin vers une image sous la forme de chaîne de caractère et qui renvoie la matrice associée à l'image convertie en niveau de gris.

Exercice 3. *Symétrie Verticale*

Implémentez une fonction `symetrie_verticale` qui prend en entrée une matrice représentant une image en niveau de gris et qui renvoie la matrice représentant l'image obtenue en faisant une symétrie axiale par rapport à l'axe vertical coupant l'image en son milieu.

FIGURE 3 – Symétrie verticale

En utilisant `image2matrice`, `symetrie_verticale` et `Image.fromarray`, affichez l'image obtenue en prenant la symétrie verticale de `velociraptor.jpg`.

Exercice 4. *Histogramme*

L'histogramme d'une image en niveau de gris représente la distribution des pixels de l'image selon leur intensité. Pour notre cas précis, l'histogramme d'une image sera une liste de 256 éléments, où, à l'indice i, on trouvera le nombre de pixels dont le niveau de gris est i.

On peut tracer cette histogramme grâce à la fonction `bar` du module `plt` chargé en début de fichier. La fonction `bar` prend deux entrées :

— Une liste de valeurs en abscisses (quelconques)
— Une liste de même taille donnant la hauteur de la barre associée à chaque valeur.

On trace ensuite le diagramme à l'aide de la fonction `show`. Par exemple :

```
plt.bar(["Nadine", "Cuisine", "Tajine"], [21,12,18])
plt.show()
```

En utilisant `image2matrice`, `histogramme` et le module `plt`, tracez l'histogramme de l'image `"fox.jpg"` , implémentez une fonction `histogramme` qui prend en entrée une matrice représentant une image en niveau de gris et qui renvoi l'histogramme de l'image.

Exercice 5. *Dégradé*

Le but de cet exercice est de générer un degradé circulaire en niveau de gris.

Les paramètres du dégradé sont
— l et h, la largeur et la hauteur de l'image.

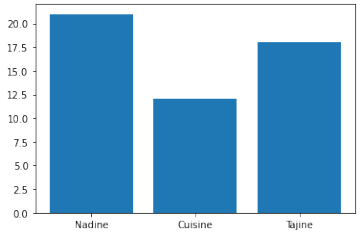

FIGURE 4 – Exemple de diagramme en barres.

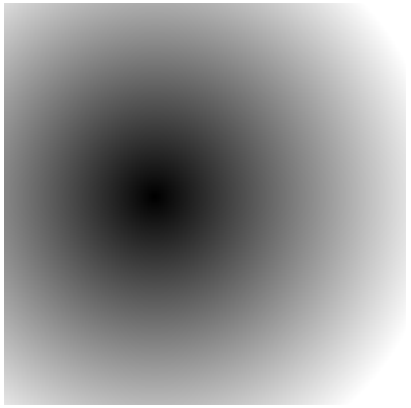

FIGURE 5 – Dégradé de centre 144,112, de rayon 212, pour une image de taille 300×300.

— x et y, les coordonnées du pixel au centre du degradé.
— r le rayon du dégradé.

Pour réaliser un dégradé on regarde l'éloignement d de chaque pixel (i, j) au centre (x, y) et on construit une échelle d'intensité telle que :

— si $d = 0$ l'intensité du pixel est 0
— si $d \geq r$ l'intensité du pixel est 255
— si $0 < d < r$ l'intensité du pixel évolue de manière uniforme entre 0 et 255.

Proposez une formule mathématique permettant le calcul de la valeur de l'intensité d'un pixel (i, j) en fonction du centre (x, y) et du rayon r.

Implémentez une fonction `degrade` qui prend en paramètre l, h, x, y et r décris ci-dessus et qui renvoie un objet `Image` représentant l'image de dimensions $l \times h$, dessinant un degradé circulaire uniforme de centre (x, y) et de rayon r.

Exercice 6. *ASCII Art*

Un ASCII Art est un dessin réalisé uniquement à partir de caractères ASCII.

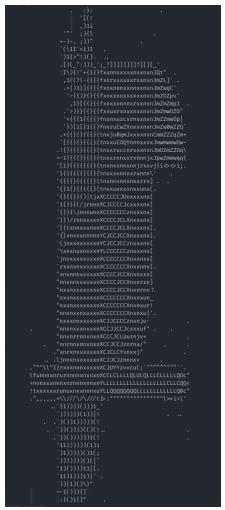

FIGURE 6 – ASCII Art

On veut construire un ASCII art à partir d'une image en niveau de gris.

Un pixel est associé à un caractère et chaque caractère est disposé en grille comme sur l'image. Chaque ligne de pixel est représenté par une ligne de caractères se terminant par un retour à la ligne, `"\n"`.

Le caractère utilisé pour chaque pixel est choisi selon l'intensité du pixel en niveau de gris. Plus le pixel est foncé (proche de 0), plus le caractère aura de pixels noir dans sa représentation. Par exemple `'$'` représentera le 0 et `'.'` le 254. Il est clair que le caractère '$' contient plus de pixels noirs que le caractère `'.'`.

Voici l'échelle de 70 caractères utilisés pour réaliser des intensités différentes.

```
echelle_ascii = "$@B%8&WM#*oahkbdpqwmZO0QLCJUYXzcvunxrjft/\|()1{}[]?-_+~<>i!lI;:,\"^`'.
```

Implémentez une fonction `image2ascii` qui prend en entrée le chemin vers une image et renvoie une chaîne de caractère représentant l'image en ASCII Art.

Affichez le perroquet en ASCII Art en tapant dans la console Spyder :

```
print(image2ascii("img/perroquet.jpg"))
```

L'image est parfois trop grosse. Il est plus facile de regarder le résultat en l'ouvrant dans un bloc note et en réduisant la police.

Transformez l'image `"nadine.jpg"` en ASCII art et écrivez la chaîne obtenue en résultat dans le fichier `"img/nadine_ascii.txt"`. Vous pouvez ouvrir le fichier dans un bloc note et réduire la police (molette sous Windows par exemple) pour visualiser le résultat.

Faites de même avec le logo python `"python.jpg"` . Attention à ne pas afficher le résultat dans la console Spyder. L'image fait `1200x1200` pixels, vous risquez d'attendre très longtemps avant que le texte s'affiche...