I- lectures et affichage des images en python :

a) lecture :

PIL permet de lire une image enregistrée localement dans de nombreux formats (il n'est pas forcément disponible sur vos machines). `matplotlib.image` ne permet que de charger des `.png`. Mais vous avez directement un tableau numpy (pas besoin de transformation)

avec matplotlib.image :

```
Entrée [2]: import matplotlib.image as mpimg
            import numpy as np
            img = mpimg.imread("img.jpg")
```

avec PIL image :

```
from PIL import Image
import numpy as np
imgpil = Image.open("img.jpg")
#img = np.array(imgpil) # Transformation de l'image en tableau numpy
```

b) affichage :

Matplotlib permet d'afficher une image (si c'est un tableau numpy).

```
import matplotlib.pyplot as plt
plt.imshow(img)
plt.show()
```

c) transformation d'une image en gris :

Il est donc intéressant de pouvoir convertir une image couleur en niveau de gris tout en agissant sur les poids relatifs sur les couleurs RGB comme dans cet exemple:

```python
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
#cette fonction permet de jouer sur les couleurs rgb et les transformer
def rgb2gray(rgb):
    return np.dot(rgb[...,:3], [0.299, 0.587, 0.144])

img = mpimg.imread('img.jpg')

gray = rgb2gray(img)

plt.imshow(gray, cmap = plt.get_cmap('gray'))

plt.savefig('imgg.png')
plt.show()
```

II- les distances :

Les distances sont généralement calculées pour déterminer la similarité entre 2 points de vecteur. Or on sait que les images en numpy donc il faut convertir un tableau de numpy d'image en un vecteur en utilisant flatten().

a) flatten :

Parfois, vous devez aplatir un tableau. L'ancienne façon serait de le faire en utilisant quelques boucles l'une dans l'autre. Bien que cela fonctionne, vous pouvez vous en passer. Cette astuce montre comment prendre un tableau de tableaux et l'aplatir sur une seule ligne à l'aide de la compréhension de tableau

avant l'utilisation de flatten :

```
print(img)
```

```
[[[ 13  12   8]
  [  1   0   0]
  [ 17  16  11]
  ...
  [ 11   7   0]
  [ 17  13   0]
  [  7   3   0]]

 [[  5   4   0]
  [  6   5   1]
  [  2   1   0]
  ...
  [ 11   6   0]
  [  6   2   0]
  [ 24  21  21]
```

avec flatten :

```
: print(img.flatten())
```

```
[13 12  8 ... 32 38 28]
```

b) distance simple entre 2 images :

Définition:

$$D=\sum(Xi-Yi)^2$$

Or X(X1,.....,Xn) et Y(Y1,.....,Yn) 2 vecteurs de même longueur

Application :

```
Entrée [15]: def Distance(i1, i2):
                 return np.sum((i1-i2)**2)

Entrée [16]: Distance(img1,img2)

Out[16]: 1802.0
```

c) Euclidienne distance :

Définition :

$$\sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$

Application :

```
Entrée [18]: def EuclidienneDistance(i1, i2):
                 return np.sqrt(np.sum((i1-i2)**2))

Entrée [19]: EuclidienneDistance(img1,img2)

Out[19]: 42.44997055358225
```

d) Cosine similarity :

Définition :

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|_2 \|\mathbf{B}\|_2} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}$$, where A_i and B_i are components of vector A and B

Application :

En python la cosine similarité entre 2 images :

```
Entrée [20]: def cosSimilarity(i1, i2):
                 return np.sum(i1*i2)/(np.sqrt(np.sum(i1**2))*np.sqrt(np.sum(i2**2)))

Entrée [21]: cosSimilarity(img1,img2)

Out[21]: 0.7734140182844601
```

e) Similarité de jaccard entre 2 images :

Définition :

Jaccard Similarity

Consider two sets $A = \{0, 1, 2, 5, 6\}$ and $B = \{0, 2, 3, 5, 7, 9\}$. How similar are A and B?

The *Jaccard similarity* is defined

$$\begin{aligned} \text{JS}(A, B) &= \frac{|A \cap B|}{|A \cup B|} \\ &= \frac{|\{0, 2, 5\}|}{|\{0, 1, 2, 3, 5, 6, 7, 9\}|} = \frac{3}{8} = 0.375 \end{aligned}$$

application :

Nous allons maintenant calculer la similarité de jaccard entre 2 images en python, En utilisant la formule :

```
Entrée [22]: def Similarity_jaccard(i1, i2):
                 return len(np.intersect1d(i1,i2))/(len(i1)+len(i2)-len(np.intersect1d(i1,i2)))

Entrée [23]: Similarity_jaccard(img1,img2)

Out[23]: 0.10344827586206896
```