

A. Parcours des pixels d'une image

E.479   (Tracer une droite sur une image)

1 a Télécharger l'image suivante:  479-chat.jpg

Puis, saisissez le code ci-dessous dans votre IDE/Python :

```
1 from PIL import Image
2 img = Image.open('chat.jpg')
3 WIDTH = img.size[0]
4 HEIGHT = img.size[1]
5 pixels = img.load()
6 for i in range(WIDTH):
7     pixels[i,50] = (255,0,0)
8 img.save("chat2.jpg")
9 img.show()
```

 479-code.py

Indication: le fichier de votre programme Python et l'image doivent être enregistré dans le même dossier.

b Compilez votre programme et observez l'apparition d'une image "chat2.jpg" dans le dossier.
Quelle différence est apportée à votre image?

.....

.....

2 a Modifiez le code pour que le trait rouge soit vertical.

b Modifiez le code pour le trait rouge soit diagonal.

E.480   (Symétrique d'une image)

1 a Télécharger l'image suivante:  480-chat.jpg

Puis, saisissez le code ci-dessous dans votre IDE/Python :

```
1 from PIL import Image
2 img = Image.open('chat.jpg')
3 WIDTH = img.size[0]
4 HEIGHT = img.size[1]
5 pixels = img.load()
6 imgNew = Image.new('RGB', (WIDTH,HEIGHT), 'black')
7 pixelsNew = imgNew.load()
8 for i in range(WIDTH):
9     for j in range(HEIGHT):
10         pixelsNew[i,j] = pixels[WIDTH-i-1,j]
11 imgNew.save("chat2.jpg")
12 imgNew.show()
```

 480-code.py

Indication: le fichier de votre programme Python et l'image doivent être enregistré dans le même dossier.

b Compilez votre programme et observez l'apparition d'une image "chat2.jpg" dans le dossier.
Quelle différence est apportée à votre image?

.....

.....

2 a Modifiez le code pour que l'image subisse une symétrie axiale dont l'axe est horizontal passant par le "milieu" de l'image.

b Modifiez le code pour que l'image subisse une symétrie centrale.

Question subsidiaire: sauriez-vous faire un demi-tour à l'image?

E.481   (Extrait des calques de couleurs)

1 a Télécharger l'image suivante:  481-chat.jpg

Puis, saisissez le code ci-dessous dans votre IDE/Python :

```
1 from PIL import Image
2 img = Image.open('chat.jpg')
3 WIDTH = img.size[0]
4 HEIGHT = img.size[1]
5 pixels = img.load()
6 imgNew = Image.new('RGB', (WIDTH,HEIGHT), 'black')
7 pixelsNew = imgNew.load()
8 for i in range(WIDTH):
9     for j in range(HEIGHT):
10         r,g,b = pixels[i,j]
11         pixelsNew[i,j] = (r,0,0)
12 imgNew.save("chat2.jpg")
13 imgNew.show()
```

 481-code.py

Indication: le fichier de votre programme Python et l'image doivent être enregistré dans le même dossier.

b Compilez votre programme et observez l'apparition d'une image "chat2.jpg" dans le dossier.

Comment justifiez que le calque rouge de l'image originale apparaisse?

.....

.....

2 a Faites apparaître le calque vert

b Faites apparaître le calque bleu

3 Pour passer l'image en niveau de gris, pour chaque pixel, on fait la moyenne des 3 composantes: rouge, vert, bleu.
Réaliser le filtre "niveau de gris"

E.753   (calques et symétries)

Le code ci-dessous prend l'image *imgOriginale.jpg*, parcourt l'ensemble des pixels de cette image pour créer la nouvelle image *imgDesintation.jpg*

```
from PIL import Image
img = Image.open('imgOriginale.jpg')
WIDTH = img.size[0]
HEIGHT = img.size[1]
pixels = img.load()
imgNew = Image.new('RGB', (WIDTH,HEIGHT), 'black')
pixelsNew = imgNew.load()
for i in range(WIDTH):
    for j in range(HEIGHT):
        ...
imgNew.save("imgDestination.jpg")
```

```
imgNew.show()
```

La ligne 13 a été effacée et vous devez donner les instructions nécessaires pour répondre à chacune des questions :

- 1 Les instructions ci-dessous permettent de lire les trois composantes (*rouge*, *vert*, *bleu*) de chaque pixel et de n'écrire que la composante rouge dans la nouvelle image :

```
1 r,g,b = pixels[i][j];  
2 pixelsNew[i,j] = (r,0,0)
```

- a Quelles instructions doit-on écrire pour que seul le calque bleu apparaisse?
- b Quelles instructions doit-on écrire pour afficher l'image

en niveau de gris?

- c Quelles instructions doit-on écrire pour qu'à partir de l'image en niveau de gris s'affichent en blanc seulement les pixels ayant une intensité de gris supérieur ou égal à 120, et en noir les autres?

- 2 Les instructions ci-dessous permettent de réaliser une symétrie axiale d'axe vertical :

```
1 pixelsNew[i,j] = pixels[WIDTH-i-1,j]
```

- a Quelles instructions doit-on écrire pour obtenir une symétrie axiale d'axe horizontal?
- b Quelles instructions doit-on écrire pour obtenir une rotation de 180°?

B. Agrandissement et réduction d'images

E.1351  

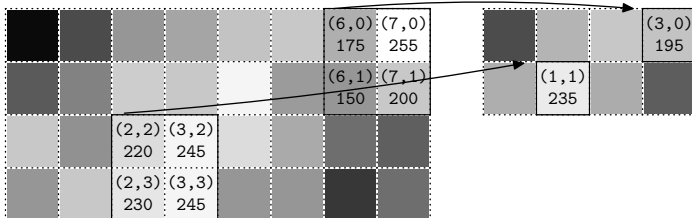
- 1 Télécharger l'image ci-dessous et rechercher sa dimension en pixel :

 1351-image.jpg

- 2 Le but de cet exercice est d'obtenir une réduction de cette image par 2 de ses dimensions : obtenir une image dont les dimensions sont 600×300.

La première technique que nous allons utiliser est simplement de récupérer 1 pixel sur 2 à l'horizontal et 1 pixel sur 2 à la verticale.

L'image ci-dessous illustre cette transformation :



Pour réaliser cet algorithme, complétez cet algorithme :

```
from PIL import Image  
img = Image.open('image.jpg')  
w = img.size[0]  
h = img.size[1]  
pixels = img.load()  
newW = ...  
newH = ...  
imgNew = Image.new('RGB', (newW,newH), 'black')  
pixelsNew = imgNew.load()  
for i in range(newW):  
    for j in range(newH):  
        pixelsNew[i,j] = pixels[...,...]  
imgNew.save("image2.jpg")  
imgNew.show()
```

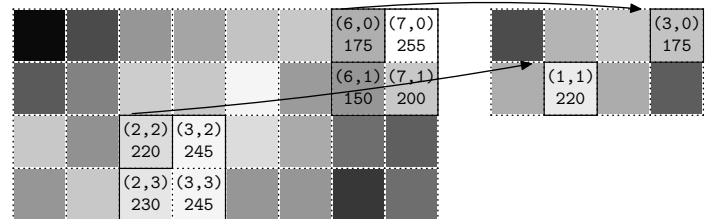
 1351-code.py

Remarque : on supposera que les dimensions de l'image sont des multiples de 2.

E.1352  

Nous allons reprendre l'exercice précédent, mais cette fois-ci, pour réduire la dimension d'une image par 2, nous allons faire la moyenne des quatre pixels présents à droite et en dessous du pixel considéré.

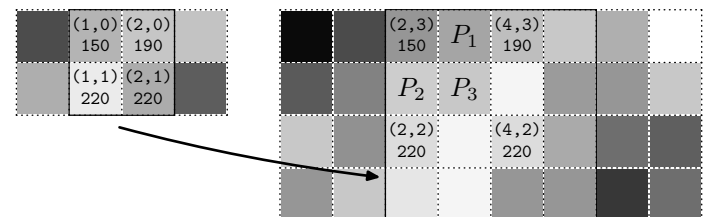
Cette méthode est illustrée ci-dessous :



- 1 Réalisez cet algorithme en reprenant votre code précédent.
- 2 Comparez les 2 images obtenues au cours de ces exercices

E.1353  

Pour agrandir les dimensions d'une image par 2, on utilise le principe représenté ci-dessous sur une image en niveau de gris :



- chaque bloc de 2×2 pixels est recopié dans la nouvelle image en les écartant de 1 pixel chacun.
- Les pixels ajoutés seront obtenus par la moyenne de leurs 2 pixels mitoyens : soit verticalement, soit horizontalement, soit en diagonales.

Nous commencerons par utiliser le code ci-dessous :

```
from PIL import Image  
img = Image.open('image.jpg')  
w = img.size[0]  
h = img.size[1]  
pixels = img.load()  
imgNew = Image.new('RGB', (2*w,2*h), 'black')  
pixelsNew = imgNew.load()
```

```
for i in range(w):
    for j in range(h):
        pixelsNew[2*i,2*j] = pixels[i,j]
imgNew.save("image2.jpg")
imgNew.show()
```

 1353-code.py

Nous pourrions utiliser l'image ci-dessous et on observera, en zoomant dessus, les détails de la nouvelle image obtenue:

 1353-image.jpg

E.1354



Pour pouvoir augmenter les dimensions d'une image quel que soit le facteur k d'agrandissement ($k \in \mathbb{N}$), on utilise l'interpolation linéaire. En voici le principe:

Soit f une fonction définie sur $\mathbb{R}^2$ et on souhaite construire la fonction g telle que:

Pour tout $(u,b) \in \mathbb{N}^2$, on a:

$$g(k \cdot u, k \cdot v) = f(u, v)$$

Dans le cas, où x et y sont deux nombres entiers non-multiples de k , on définit a et b les deux entiers tels que:

$$a < \frac{x}{k} < a+1 \quad ; \quad b < \frac{y}{k} < b+1$$

Notons: $\alpha = \frac{x}{k} - a \quad ; \quad \beta = \frac{y}{k} - b$

On définit alors l'image de (x,y) par la fonction g :

$$g(x,y) = (1-\alpha)(1-\beta) \cdot f(a,b) + (1-\alpha)\beta \cdot f(a,b+1) \\ + \alpha(1-\beta) \cdot f(a+1,b) + \alpha\beta \cdot f(a+1,b+1)$$

Remarque: cet algorithme s'appelle l'interpolation linéaire.

C. PIL et Tkinter

E.491



Télécharger l'image suivante:  491-chien.jpg

Recopier le code ci-dessous et faites en sorte de compléter les fonctionnalités des boutons en bas de la page:

```
1 import tkinter as tk
2 from tkinter import *
3 from PIL import Image, ImageTk
4 fichierImage = "chien.jpg"
5 def flipH():
6     global tmp
7     print("flipH")
8     w1 = image1.size[0]
9     h1 = image1.size[1]
10    pixelsA = image1.load()
11    pixelsB = image2.load()
12    for i in range(w1):
13        for j in range(h1):
14            pixelsB[w1-i-1,j] = pixelsA[i,j]
15    tmp = ImageTk.PhotoImage(image2)
16    label2.configure(image = tmp)
17    label2.image = image2
18 def flipV():
19    print("flipV")
```

```
20 root = Tk()
21 frame = tk.Frame(root, width=800, height=600, background="white")
22 frame.pack_propagate(0)
23 frame.pack()
24 btnA = Button(root, text = 'flip H', bd = '5',
25               command = flipH)
26 btnA.pack(side = tk.LEFT)
27 btnB = Button(root, text = 'flip V', bd = '5',
28               command = flipV)
29 btnB.pack(side = tk.LEFT)
30 btnZ = Button(root, text = 'Fermer', bd = '5',
31               command = root.destroy)
32 btnZ.pack(side = tk.RIGHT)
33 # Place l'image originale
34 image1 = Image.open(fichierImage)
35 test1 = ImageTk.PhotoImage(image1)
36 label1 = tk.Label(image=test1)
37 label1.place(x=0, y=0)
38 image2 = Image.open(fichierImage)
39 test2 = ImageTk.PhotoImage(image2)
40 label2 = tk.Label(image=test2)
41 label2.place(x=400, y=0)
42 root.mainloop()
```

 491-code.py