Partie 4 : La simulation numérique

Chapitre 3

Traitement de l'image ~ Corrigé

Exercice 1 :

```
from PIL import Image
import os

dossierTravail = os.getcwd()

# ouverture d'une image
source = dossierTravail+'/pythongris.jpg'

img=Image.open(source) # ouvrir l'image

imgMode = img.mode

if imgMode =='1' :
        noir = 0
        blanc = 1
elif imgMode =='L' :
        noir = 0
        blanc = 255
elif imgMode =='RGB' :
        noir = (0,0,0)
        blanc = (255,255,255)

L = list(img.getdata())
nbNoir = 0
nbBlanc = 0
for p in L:
        if p == noir :
                nbNoir += 1
        elif p == blanc:
                nbBlanc += 1

print(nbNoir, nbBlanc)
```

Exercice 2 :

```
from PIL.Image import *

img = new ('1',(100,100),"white")

larg, haut = img.size

for i in range (haut):
     for j in range(larg):
          img.putpixel((i,larg-1-i),0)
          img.putpixel((i,i),0)

img.show()
```

Exercice 3 :

```
from PIL.Image import *
import numpy as np

n=10
p=10

nouv = new ('1',(n,p))

larg,haut = nouv.size

def f(i,j):
    return (i+j)%2

t1 = np.fromfunction(f,(larg,haut))

m=t1.reshape(haut*larg)
L=list(m)
nouv.putdata(L)
nouv.show()
```

Exercice 4 :

```
from PIL import Image
import os
import numpy as np

dossierTravail = os.getcwd()

# ouverture d'une image
source = dossierTravail+'/pythongris.jpg'

img=Image.open(source) # ouvrir l'image

imgGris = img.convert("L")

larg, haut = imgGris.size


for i in range (haut):
     for j in range(larg):
          imgGris.putpixel((i,j),255-imgGris.getpixel((i,j)))

imgGris.show()
```

Exercice 5 :

```
from PIL.Image import *

source=open("tiger.jpg")

L,H = source.size

e = 20 #bordure de 20 pixels
```

```
#colorer la bordure orange
distination=new("RGB",(L+2*e,H+2*e),'orange')

for i in range (L):
      for j in range (H):
            p=source.getpixel((i,j))
            distination.putpixel ((i+e,j+e),p)

distination.save('tiger_bordure.jpg')
distination.show()
```

Exercice 6 :

```
import numpy as np
import PIL.Image as im

tiger = im.open("tiger.jpg")
largeur, hauteur  =tiger.size
tabPixels = np.array(tiger)


largeur2 = largeur *2 #largeur double
hauteur2 = hauteur #même hauteur
tiger2 = im.new('RGB',(largeur2, hauteur2))

L = 0
for ligne_tab in tabPixels:
      C = 0
      for pixel in ligne_tab:
            tiger2.putpixel((C, L),tuple(pixel))
            tiger2.putpixel((largeur2 - C - 1, L),tuple(pixel))
            C += 1
      L += 1

tiger2.show()
tiger2.save("tigreSymetrieAxeVertical.jpg")
```

Exercice 7 :

```
import numpy as np
import PIL.Image as im

tiger = im.open("tiger.jpg")
largeur, hauteur  =tiger.size
tabPixels = np.array(tiger)

tiger2 = im.new('RGB',(largeur*2, hauteur*2))
L = 0
for ligne_tab in tabPixels:
      C = 0
      for pixel in ligne_tab:
            tiger2.putpixel((C , L),tuple(pixel))
            tiger2.putpixel((C+1, L),tuple(pixel))
            tiger2.putpixel((C, L+1),tuple(pixel))
            tiger2.putpixel((C+1, L+1),tuple(pixel))
            C += 2
      L += 2
```

```
tiger2.show()
tiger2.save("tigreGrandFormat.jpg")
```

Exercice 8 :

```
import numpy as np
import PIL.Image as im

tiger = im.open("tiger.jpg")
largeur, hauteur  =tiger.size
tiger1= im.new('RGB',(hauteur, largeur))

tabPixels = np.array(tiger)

#rotation de 90 degré sans perte de données
ligne_image = 0
for ligne_tab in tabPixels:
     colonne_image = 0
     for pixel in ligne_tab:
          tiger1.putpixel((ligne_image,colonne_image),tuple(pixel))
          colonne_image += 1
     ligne_image += 1

tiger1.show()
tiger1.save("tigreRotation90degre.jpg")

#Etape 2 : Symetrie par rapport à l'axe vertical
tiger = im.open("tigreRotation90degre.jpg")
largeur, hauteur  =tiger.size
tabPixels = np.array(tiger)

largeur2 = largeur *2 #largeur double
hauteur2 = hauteur #même hauteur
tiger2 = im.new('RGB',(largeur2, hauteur2))

L = 0
for ligne_tab in tabPixels:
     C = 0
     for pixel in ligne_tab:
          tiger2.putpixel((C, L),tuple(pixel))
          tiger2.putpixel((largeur2 - C -1 , L),tuple(pixel))
          C += 1
     L += 1
tiger2.show()
tiger2.save("tigreRotation90DegreSymetrieAxeVertical.jpg")
```

Exercice 9 :

```
from PIL.Image import *
import numpy as np
from random import *
#Q1
def compression_RLE(img):
        L = [img.mode, img.size]
        data = img.getdata()
        pixels = list(data)
```

```
        i =0 #indice dans la liste de pixels
        while i < len(pixels):
                n = 1
                j = i+1
                while j<len(pixels) and pixels[j]==pixels[i]:
                        j+=1
                        n+=1
                L.append((n,pixels[i]))
                i=j
        return L
#Q2
def decompression_RLE(L):
        pixels = []
        for e in L[2:]:
                for i in range(e[0]):
                        pixels.append(e[1])

        mode = L[0]
        larg, haut = L[1]
        img = new (mode,(larg,haut))
        img.putdata(pixels)
        img.show()
#test
#Générer une image Noir et Blanc aléatoire
larg = 10
haut = 10
tabPixel = np.ones((larg,haut))

for i in range(haut):
        for j in range(larg):
                #générere des 1 et des 0 aléatoirement
                tabPixel[i,j]=(int(random()*10))%2

tabPixel=tabPixel.reshape((larg*haut))
print(tabPixel)
L = list(tabPixel)
print(L)
img = new ('1',(larg,haut))
img.putdata(L)
img.show()
#Compression
L = compression_RLE(img)
print(L)
#Decompression
decompression_RLE(L)
```

Exercice 10 :

```
import PIL.Image as im
import numpy as np
import os
from math import *
```

```
def ajouter_bordure(img,k=1):
    ''' Ajouter une bordure de k pixels autour de l'image '''
    larg, haut = img.size #dimensions de l'image
    #les pixels de la bordure sont tous à 0
    img1 = im.new('L',(larg+2*k,haut+2*k),0)
    for i in range(larg):
        for j in range(haut):
            pixel = img.getpixel((i,j))
            img1.putpixel((i+k,j+k),pixel)

    return img1

def supprimer_bordure(img,k=1):
    ''' supprimer la bordure de l'image d'épaisseur k pixel(s) '''
    larg, haut = img.size #dimensions de l'image
    t=np.array(img)
    # Suuprimer k premiere(s) ligne(s)
    for i in range(k):
        t=np.delete(t,0,0) #tab, pos, axis
    # Suuprimer k deniere(s) ligne(s)
    for i in range(k):
        t=np.delete(t,-1,0) #tab, pos, axis
    # Suuprimer k premiere(s) colonne(s)
    for i in range(k):
        t=np.delete(t,0,1) #tab, pos, axis
    # Suuprimer k deniere(s) colonne(s)
    for i in range(k):
        t=np.delete(t,-1,1)

    img = im.fromarray(t)
    return img

def filtre_moyenneur(img,k):
    ''' Appliquer le filre moyenneur d'épaisseur 2*k+1 pixel(s) '''
    img1 = ajouter_bordure(img,k)

    imgMode = img.mode

    taille = 2*k+1   # taille du filter

    # Convertir l'image en tableau numpy
    t = np.array(img1)

    nbColonnes, nbLignes = img1.size

    #calculer la moyenne des sous matrices
    for i in range(nbLignes-taille-1):
        i1,i2=i,i+taille
        for j in range(nbColonnes-taille-1):
            j1,j2=j,j+taille
            ind = ceil(taille/2)-1
            t[i1+ind,j1+ind]= ceil(t[i1:i2,j1:j2].mean())
    img1=im.fromarray(t)

    img2 = supprimer_bordure(img1,k) #resultat

    return img2
```

```
if __name__ == '__main__':

    #Variables globales
    k = 5 # taille du filre : 2k+1

    # Dossier de travail : dossier courant
    folder = os.getcwd()
    file = "python1.jpg"
    source = folder +"/" + file

    # Charger l'imager à filtrer
    img = im.open(source)

    #Convertir l'image à filtrer en niveau de gris
    img = img.convert('L') #valeurs de 0 à 255

    # Afficher l'image avant filtrage
    img.show()

    #tester le filtre moyenneur
    imgF = filtre_moyenneur(img,k)

    # Afficher l'image après filtrage
    imgF.show()
    img.save("filtre_moyenneur_RGB_k10.jpg")
```

Exercice 11 :

```
import PIL.Image as im
import numpy as np
import os
from math import *

#Q1
def convolution (p1, p2, p3, p4, p5, p6, p7, p8, p9, A):
        listePixels = [p1, p2, p3, p4, p5, p6, p7, p8, p9]
        listePixels = np.array(listePixels)
        listePixels = listePixels.reshape(3,3)
        #print(listePixels)
        m = np.array(A)
        return np.sum(listePixels * m)

#Q2
def appFiltreConv (T, nc, nl, A, img2):
    for c in range(1,nc-1):
       for l in range(1,nl-1):
          p1 = T[c-1,l-1]
          p2 = T[c,l-1]
          p3 = T[c+1,l-1]
          p4 = T[c-1,l]
          p5 = T[c,l]
          p6 = T[c+1,l]
          p7 = T[c-1,l+1]
          p8 = T[c,l+1]
          p9 = T[c+1,l+1]
```

```
            pixel = convolution (p1, p2, p3, p4, p5, p6, p7, p8, p9, A)
            pixel = abs(ceil(pixel/9))
            img2.putpixel((c,l),pixel)
    return img2

#test
p1, p2, p3, p4, p5, p6, p7, p8, p9 = 1,2,1,2,12,12,2,1,2
A=[[-1,-1,-1],[0,0,0],[-1,-1,-1]]
print(convolution (p1, p2, p3, p4, p5, p6, p7, p8, p9, A))

# Dossier de travail : dossier courant
folder = os.getcwd()
file = "pythongris.jpg"
source = folder +"/" + file

# Charger l'imager à filtrer
img = im.open(source)

nc, nl = img.size

#Convertir l'image à filtrer en niveau de gris
img1 = img.convert('L') #valeurs de 0 à 255

img2 = im.new (img1.mode, img1.size)

T = img1.load()

img2 = appFiltreConv(T, nc, nl, A, img2)
img2.show()
```

Exercice 12 :

```
import PIL.Image as im
import numpy as np
import os
from math import *
import random

#Q1
def ouvrir_image(img):
        # Charger l'imager à filtrer
        img = im.open(source)

        #Convertir l'image à filtrer en niveau de gris
        img1 = img.convert('L') #valeurs de 0 à 255
        #
        return np.array(img1,int)

#Q2
def construire_image(m,nom):
        img = im.fromarray(m)
        img = img.convert('L')
        #donner un nouveau nom aléatoire
        img.save(os.getcwd()+"/"+nom+".jpg")
```

```
#Q3
def ajouter_bordure(m, e=1):
        lig, col = m.shape
        lig += 2*e
        col += 2*e
        m1 = np.zeros((lig, col))
        for i in range(lig-2*e):
                for j in range(col-2*e):
                        m1[i+e,j+e]= m[i,j]
        return m1

#Q3 (autre version: fonctions du module numpy)
def ajouter_bordure_1(m):
        t=np.copy(m)
        # insérer une ligne au début
        t=np.insert(t,0,0,0)#tab,pos,val,axis

        # insérer une colonne au début
        t=np.insert(t,0,0,1)#tab,pos,val,axis

        # ajouter un ligne la fin
        c=t.shape[1]
        ligne=[[0]*c]
        t=np.append(t,ligne,0)#tab, ligne|colonne, axis

        # ajouter un colonne à la fin
        n=t.shape[0]
        col=[[0]]*n
        t=np.append(t,col,1)#tab, ligne|colonne, axis

        return t
#Q4
def dilatation(m):
        nbLignes, nbColonnes = m.shape
        for i in range(nbLignes-3+1):
                for j in range(nbColonnes-3+1):
                        m[i+1,j+1]=np.max(m[i:i+3,j:j+3])
        return m

#Q5
def supprimer_bordure(m,k=1):
     for i in range(k):
         m = np.delete(m,0,0) #ligne 0
         m = np.delete(m,-1,0) #ligne -1
         m = np.delete(m,0,1) #colonne 0
         m = np.delete(m,-1,1) #colonne -1
     return m

if __name__=="__main__":
        #ouvrir une image
        folder = os.getcwd()
        file = "camrea.png"
        source = folder +"/" + file
        m = ouvrir_image(source)

        #Ajouter une bordure
```

```
m = ajouter_bordure(m,1)
print(m)

#dilater
m = dilatation(m)
print(m)
#supprimer la bordure
m = supprimer_bordure(m,1)
print(m)
#construire et enregistrer la nouvelle image
construire_image(m,str(random.randint(10,1000)))
```