SNT – ELLA

LA PHOTO NUMÉRIQUE

SÉANCE 4 - CRYPTER DES IMAGES MATRICIELLES AVEC JUPYTER NOTEBOOK

PARTIE 1 : DÉCOUVERTE DE JUPYTER NOTEBOOK

1. INTRODUCTION

Jupyter Notebook est un concepteur de cahier électronique qui, dans le même document, peut rassembler du texte, des images, des formules mathématiques et du code python exécutable.
Il est manipulable interactivement dans un navigateur web.

2. EXEMPLE : CRÉATION D'UNE FONCTION MOYENNE

1. Dérouler le menu démarrer et exécuter **Jupyter Notebook**.
2. Un navigateur se lance. A droite de la page, dérouler le menu **Nouveau** → **Python**.
3. Saisir chacune des trois **entrées** suivantes en la validant à chaque fois en pressant **MAJ** + **ENTREE**.

Entrée [1]:
```
def moyenne(a,b,c):
  m=(a+b+c)/3
  return(m)
```

Entrée [2]:
```
moyenne(10 , 12 , 17)
```

Entrée [3]:
```
moyenne(14 , 10.5 , 18.75)
```

3. EXERCICE : CRÉATION D'UNE IMAGE ROUGE ET D'UNE IMAGE BLEUE

1. Saisir chacune des trois **entrées** suivantes.

Entrée [4]:
```
from PIL import Image
```

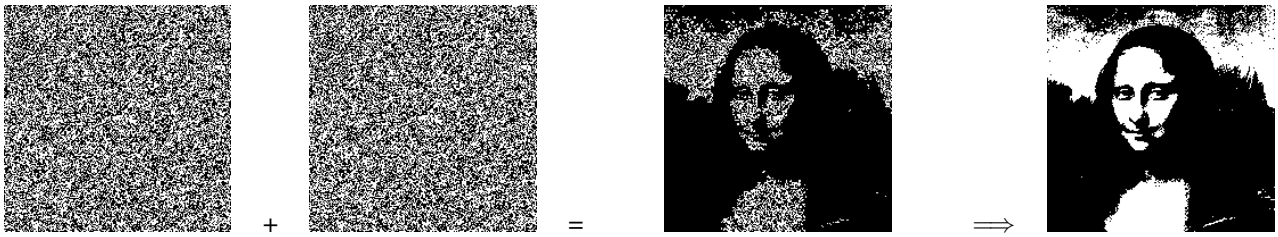
Entrée [5]:
```
def creerImage():

  largeur = 200
  hauteur = 100
  image=Image.new('RGB', (largeur,hauteur))
  for x in range(largeur):
      for y in range(hauteur):
          image.putpixel((x,y),(255,0,0))

  return (image)
```

Entrée [6]:
```
creerImage()
```

2. Modifier la fonction définie à l'**entrée [5]** pour qu'elle fabrique une image bleue de définition 320×200 pixels.
3. Revalider l'**entrée [6]** afin de vérifier la modification.

1. PRINCIPE

On appelera **bruit aléatoire**, une image aléatoire constituée de pixels ayant chacun la couleur soit noire soit blanche.

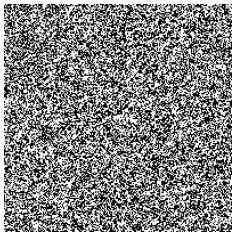


bruit aléatoire + image codée = superposition des deux images ⟹ image réelle

1. Compléter la fonction **creerBruit()** de l'entrée suivante dont le rôle est de créer une image "bruit".

Entrée [7]:

```
from random import randint      # entier aléatoire
from PIL import Image

def creerBruit(taille):
    image=Image.new('RGB', (taille,taille))
    for x in range(taille):
        for y in range(taille):
            hasard = randint(0,1)
            if (hasard ==0) :
                c=( ... , ... , ... )
            else:
                c=( ... , ... , ... )
            image.putpixel((x,y),c)

    return (image)
```



Entrée [8]:

```
bruit = creerBruit(200)

bruit
```

2. On se propose de créer une fonction **transforme()** dont le rôle est de transformer une image donnée en image en 2 couleurs.

Etape 1 : Chargement de l'image en mémoire

Entrée [9]:
```
def charger_image(fichier, resize=None):
  img = Image.open(fichier)

  if resize is not None:
      img = img.resize(resize, Image.ANTIALIAS)

  return img
```

Entrée [10]:
```
photo = charger_image('photo.png', resize=(200,200) )
```

Etape 2 : Transformation de l'image en deux couleurs

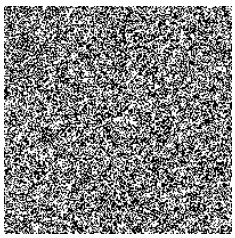
Entrée [11]:
```
def transforme(img):
  largeur=img.width
  hauteur=img.height
  image=Image.new('RGB', (largeur,hauteur))
  for x in range(largeur):
      for y in range(hauteur):
          (r,v,b) = img.getpixel((x,y))
          m = (r+v+b)//3    # moyenne des 3 composantes
          if (m> ... ) :          # le pixel est  clair
              image.putpixel((x,y),( ... , ... ,... ))
          else :
              image.putpixel((x,y),( ... , ... ,... ))

  return (image)
```

Entrée [12]:
```
photoT = transforme(photo)

photoT
```

Etape 3 : Création de l'image cryptée par le "bruit"

Entrée [13]:
```
def inversepixel (c):
  (r,v,b) = c
  inv = (255-r,255-v,255-b)
  return  inv

def crypte(img,bruit):
  largeur=img.width
  hauteur=img.height
  image = Image.new('RGB', (largeur,hauteur))
  for i in range(largeur//2):
      for j in range(hauteur//2):
          (r,v,b) = img.getpixel((i*2,j*2))
          if (r,v,b)==(0,0,0):
              image.putpixel((i*2,j*2),inversepixel( bruit.getpixel((i*2,j*2)) ))
              image.putpixel((i*2+1,j*2),inversepixel( bruit.getpixel((i*2+1,j*2)) ))
              image.putpixel((i*2,j*2+1),inversepixel( bruit.getpixel((i*2,j*2+1)) ))
              image.putpixel((i*2+1,j*2+1),inversepixel( bruit.getpixel((i*2+1,j*2+1)) ))
          else :
              image.putpixel((i*2,j*2), bruit.getpixel((i*2,j*2)) )
              image.putpixel((i*2+1,j*2), bruit.getpixel((i*2+1,j*2)) )
              image.putpixel((i*2,j*2+1), bruit.getpixel((i*2,j*2+1)) )
              image.putpixel((i*2+1,j*2+1), bruit.getpixel((i*2+1,j*2+1)) )

  return (image)
```

Entrée [14]:
```
photoCryptee = crypte(photoT,bruit)
```



Etape 4 : Superposition de l'image cryptée et du bruit

Entrée [15]:

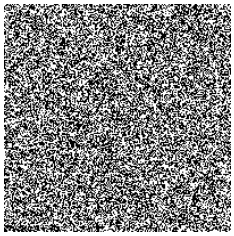
```
def superpose(img1,img2):
  largeur=img1.width
  hauteur=img1.height
  image=Image.new('RGB', (largeur,hauteur))
  for x in range(largeur):
      for y in range(hauteur):
          (r1,v1,b1) = img1.getpixel((x,y))
          (r2,v2,b2) = img2.getpixel((x,y))
          if (r1,v1,v1)==(0,0,0) or (r2,v2,b2)==(0,0,0) :
              image.putpixel((x,y), ( ... , ... , ... ))
          else:
              image.putpixel((x,y), ( ... , ... , ... ))
  return (image)
```

Entrée [16]:

```
superpose(photoCryptee, bruit)
```

2. REMARQUE

Le bruit et l'image codée sont aléatoires. Il est impossible de retrouver l'image réelle si on ne possède pas les deux images bruitées.
Ce genre de code est donc incassable.


bruit aléatoire

+

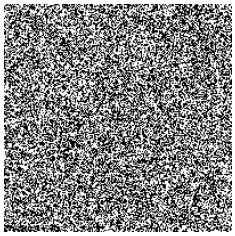
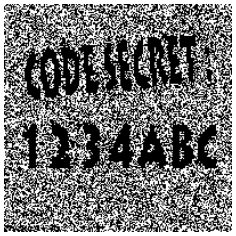


image codée

=



superposition des deux images

$\Longrightarrow$

image réelle