Cours en grande partie issue du cours de fabrice sincère
Avant de commencer ce cours il tester si le module PIL est installé.
Ouvrez **pyzo** puis testez

```
from PIL import Image
```

S'il y a une erreur alors nous devons installer le module ***PIL***, plus exactement Pillow qui est un fork de PIL pour python3.
Dans la console pyzo taper :

```
pip install pillow
cd U:\\login
```

1 Traitement des images

1.1 Représentation d'une image

a) Représentation vectorielle

Pour représenter sur ordinateur l'image d'un disque noir on peut imaginer plusieurs procédés.
On peut dire à l'ordinateur qu'on veut tracer un cercle de centre O et de rayon R et préciser les coordonnées de O et la valeur du rayon. Dans ce cas il faut disposer d'un logiciel avec les fonctions nécessaires (tracé de cercle, remplissage).
On parle alors de représentation vectorielle d'une image.
On trouve les formats ***pdf, svg, eps*** etc...

Activité 1 : Avec pyzo créez le fichier ***imagevec.svg*** avec

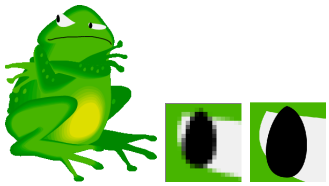
```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<svg width="800px"
     height="800px"
     xmlns="http://www.w3.org/2000/svg">
<circle cx="400px" cy="400px" r="100px" fill="red" />
<title>Disque noir en SVG </title>
 <desc>
<Creator>Meilland jean claude</Creator>
</desc>
</svg>
```

Ouvrez alors ce fichier avec ***inkscape*** pour voir la figure.

b) Représentation matricielle

Activité 2 : Télécharger l'image https://megamaths.hd.free.fr/~pi/statics/photo.jpg
Ouvrez cette image avec ***gimp*** puis faite de nombreux zoom

Une autre méthode de représentation des images consiste à superposer un quadrillage de façon à obtenir $nb_{lignes} \times nb_{colonnes}$ petits carrés les uns à côté des autres ; un de ces « carrés » est appelé ***pixel*** (pour picture element).
On associe alors une couleur à chaque pixel selon divers encodages.



On parle alors de représentation ***matricielle*** ou ***bitmap*** d'une image.
Le ***nombre de pixels*** d'une image bitmap est sa ***définition***.
La résolution d'une image bitmap s'exprime en ***ppi*** (***pixels per inch***), c'est le rapport entre sa définition et la dimension réelle de sa représentation sur support physique (papier ou écran).
dots = points = pixels.
Le pouce est une unité de mesure britannique qui vaut à peu près 2,54 cm.
On a la formule : $\text{résolution} = \dfrac{\text{définition}}{\text{dimension}}$
Pour les images matricielle on trouve les formats ***jpg, png, gif pgm*** etc...

- Le format jpg a une compression avec perte
- Le format png a une compression sans perte

Dans la suite de ce cours on utilisera des images matricielles.

1.2 Entête d'une image

a) Convertir une image au format pgm

Les images au format ***PGM*** *(portable graymap file format)* sont codées par des valeurs représentant les niveaux de gris de leurs pixels.
Ces valeurs sont comprises entre 0 (noir) et 255 (blanc) et représentent différentes teintes de gris.
C'est un format d'image matricielle, sans compression, assez peu utilisé mais qui présente l'intérêt d'être facilement manipulé.
Commencez par copier sur votre Bureau une image en couleur de format quelconque (jpeg, png, bmp, ppm ...).

Activité 3 : Convertir cette image au format PGM

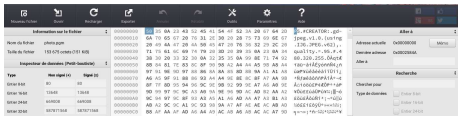
- Avec ***gimp***,
  Ouvrez l'image puis export as -> photo.pgm -> Formatage des données Brut
- Avec python3 et son module PIL *(Le plus formateur)*

```
from PIL import Image
img = Image.open('photo.jpg')
img0 = img.convert('L')  # conversion en niveau de gris
img0.save('photo.pgm')   # enregistrement de l'image
```

b) Entête d'un fichier PGM

Avec un éditeur hexadécimal, observons le contenu du fichier ***photo.pgm***

- Sous windows au lycée nous n'avons pas d'éditeur héxadécimal, nous utiliserons donc un éditeur en ligne https://hexed.it/.



- Avec python, il suffit de lire le fichier (l'image) comme un texte mais en mettant l'option ***b*** pour binaire.

```
fichier = open("photo.pgm", "rb")
texte = fichier.read()
print(texte[0:500])
```

Activité 4 : Lire le code hexadécimal de l'image avec l'éditeur en ligne, puis avec python.
On retrouve

- P5 Mot clé indiquant un fichier PGM en mode binaire
- # ... Ligne de commentaires
- 480 320 Dimensions de l'image (480 colonnes, 320 lignes)
- 255 Valeur du niveau de gris max
- Z... Niveau de gris des pixels octet par octet

Un fichier image possède ***une entête***.
L'en-tête contient en particulier la largeur et la hauteur de l'image (480 pixels x 320 pixels) et le nombre 255 (soit 256 niveaux de gris).
La couleur d'un pixel est codée sur un octet.
On va du noir ***(0x00)*** au blanc ***(0xff)*** en passant par tous les niveaux de gris.

1.3 Traitement d'images avec le module PIL

a) Découverte du module PIL

Dans le ***shell pyzo :***

```
from PIL import Image          # On importe le module
img = Image.open('photo.pgm') # on ouvre l'image
print(img.size,img.format)    # Des informations sur l'image
img.show()                    # pour voir l'image
data = list(img.getdata())    # on obtient la liste des pixels
print(data[:30])              # Pour voir les 30 premiers pixels
img = Image.open('photo.jpg') # on ouvre l'image
img.show()                    # pour voir l'image
data2 = list(img.getdata())   # on obtient la liste des pixels rgb
print(data2[:30])            # Pour voir les 50 premiers éléments
r=[i[0] for i in data]       #Pour récupérer la liste des pixels rouges,
# transformation de la premiére image
for i in range(len(data)//2): # attention c'est bien //
    data[i]=0
print(data[:30])
imNew=Image.new(img.mode ,img.size)
imNew.putdata(data)           #crétion de l'image
imNew.show()
```

b) Un exemple de traitement : l'inversion de couleurs

```
from PIL import Image
print('Inversion d\'une image PGM en mode binaire à 256 niveaux de gris\n')
def Inversion(octet):
    return 255-octet
    # cette fonction fait une inversion du niveau de gris 0 (noir)  <-> 255 (blanc)
img = Image.open('photo.pgm')# on ouvre l'image
data = list(img.getdata())   #on obtient la liste des pixels
for i in range(len(data)):
    data[i]=Inversion(data[i])

imageInverse=Image.new(img.mode ,img.size)
imageInverse.putdata(data)   #crétion de l'image
imageInverse.show()          # visualiser l'image
imageInverse.save('imageInverse.pgm')  # Sauvegarder l'image
```

Le script crée le fichier ***imageInverse.pgm*** dans le répertoire courant.

2 PIL pour aller un peu plus loin

Cette librairie fournit des outils de traitement d'images.
Le script suivant permet de convertir une photo en couleur au format JPEG en une image en niveau de gris au format PGM, puis de la transposer (retournement, miroir).
Les résultats sont affichés dans des fenêtres graphiques indépendantes et enregistrés dans des fichiers images :

```
from PIL import Image            # importation du module Image de la librairie PIL
img = Image.open('photo.jpg')    # ouverture de l'image
img.show()                       # affichage de l'image
print(img.size)                  # affichage de la taille de l'image (en pixels)
img.save('photo.ppm','PPM') # conversion au format PPM (en couleur) et enregistrement de l'image
img.show()
img0 = img.convert('L')          # conversion en niveau de gris (pour obtenir le format PGM)
img0.save('photo.pgm')           # enregistrement dans le fichier photo.pgm
img0.show()
img1 = img0.rotate(180)          # retournement de l'image
img1.show()                      # affichage et enregistrement de l'image retournée
img1.save('image_retourne.pgm')
img2 = img0.transpose(Image.FLIP_LEFT_RIGHT)     # miroir horizontal
img2.show()
img2.save('image_miroir_horizontal.pgm')
img3 = img0.transpose(Image.FLIP_TOP_BOTTOM)      # miroir vertical
img3.show()
img3.save('image_miroir_vertical.pgm')

#création d'une image
from PIL import Image,ImageDraw, ImageFont
img = Image.new("RGB",(500,300),"#006A4E")  # création d'une image 400x 400 (fond blanc)
dessin = ImageDraw.Draw(img)                # création d'un objet Draw
dessin.ellipse((125,50,325,250), fill="red")
dessin.text((10,20),"Drapeau Bangladesh",fill="black",font=ImageFont.truetype("arial.ttf", 25))
img.show()
```

2.1 Dessiner avec PIL

On peut dessiner des cercles, tracer des segments etc....

```
help(ImageDraw)
```

Regarder aussi la documentation sur internet : http://effbot.org/imagingbook/imagedraw.htm
Voici un exemple

```
from PIL import Image, ImageDraw
# création d'une image 200x200 avec un fond de couleur blanche
img = Image.new("RGB",(400,400), "#FFFFFF")
# création d'un objet Draw
dessin = ImageDraw.Draw(img)

dessin.text((100,100),"Voici un exemple",fill="red")
dessin.polygon([(200,40),(300,40),(300,60),(220,70),(200,40)], fill="red", outline="green")
dessin.line((0,200,img.size[0],img.size[1]), fill="brown")
# un arc partiel et le remplit
dessin.pieslice((-80,-80,80,80),0,90,  fill = (0, 255, 0, 128))
img.show()
```

2.2 À savoir et à retenir

- importer pil : `from PIL import Image`
- Ouvrir une image `img = Image.open('image.jpg')`
- Pour créer une nouvelle image de mode ***RGB***. `img = Image.new("RGB",(400,400))`
  Les principaux modes sont

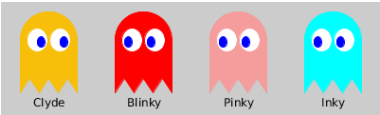
Mode	Bands	Description
"1"	1	Blanc et noire (monochrome), un bit par pixel.
"L"	1	Échelle de gris, un octet de 8 bits par pixel.
"RGB"	3	rouge-vert-bleu, trois octets par pixel.
"RGBA"	4	couleur avec une bande de transparence, de quatre octets par pixel, avec le canal alpha variant de 0 (transparent) à 255 (opaque).
"CMYK"	4	Cyan-magenta-jaune-noir, couleur de quatre octets par pixel.

- Obtenir la liste de tous les pixels : `list(img.getdata())`
- Obtenir la valeur du pixel (x,y)=(10,0) `img.getpixel((10, 0))`
- Mettre la valeur (255,0,0) au pixel (10, 0) : `img.putpixel((10, 0),(255,0,0))`
- Ajouter un un canal alpha avec la valeur 0 (transparent) à tous les pixels : `img.putalpha(0)`
- Voir l'image. `img.show()`
- Sauvegarder une image *img*. `img.save('ma_belle_image.png')`

3 Exercices

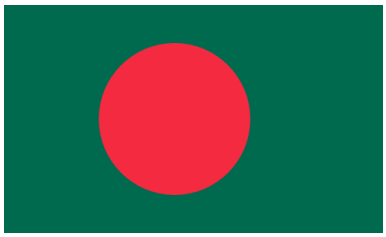
➔ ***Exercice 1***
Modifier le programme d'inversion de couleurs (3b.) pour inverser de même les pixels RGB d'une image JPEG ou PNG.

➔ ***Exercice 2***
Voici image une image http://megamaths.hd.free.fr/~pi/statics/ghost.png crée avec processing et python.



Lire les pixels de cette image puis copier les pixels sur une nouvelle image pour rendre le fond transparent.

➔ ***Exercice 3*** (Calcul du diamètre d'un disque)



Voici l'image du drapeau du Bangladesh.
http://megamaths.hd.free.fr/~pi/statics/Bangladesh.png

- Estimer le diamètre (en pixel) du disque rouge de l'image.
- Déterminer le centre du cercle
  Aide : On pourra utiliser la méthode ***getpixel()*** *du module Image de la librairie PIL,*

➔ ***Exercice 4***

Créer une image de 400 pixels sur 400 pixels, avec un carré rouge formé des points dont l'abscisse est comprise entre 100 et 250 et l'ordonnée comprise entre 50 et 200.

1. À l'aide du module ***PIL.ImageDraw*** (comme ci-dessus partie 6))
2. À l'aide de la fonction ***putpixelx***.
   Je rappelle que la syntaxe est ***putpixel((x,y),valeur)*** par exemple pour avoir un pixel x=0 et y=10 de couleur (255,0,0) on utilisera ***img.putpixel((10, 0), (255,0,0))***

4 Des idées de projet avec pil

4.1 La stéganographie

La stéganographie est l'art de cacher des messages dans une image.
Un bel exemple http://calque.pagesperso-orange.fr/js/stegano.html où on cache une image dans une image.

4.2 Code barre

On peut lire un barre code ou générer des barres codes.

http://dsaurel.free.fr/Code_barre/cb.htm

4.3 Daltonisme

Rechercher sur le Web ce qu'est le daltonisme et quelles en sont les différentes formes.
Écrire un programme qui lit une image dans un fichier au format PPM et l'affiche à l'écran comme la verrait une personne atteinte de chacune des formes de daltonisme.