

	 <p style="text-align: center;">THEME 2 : LA PHOTOGRAPHIE NUMERIQUE</p>	<p style="text-align: center;">Document Technique Pillow</p>	
Fonctionnalités du module Image de la bibliothèque PIL (Pillow)			

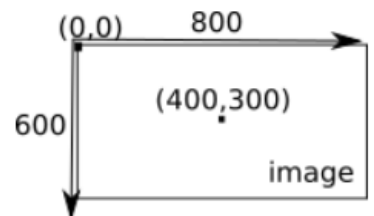
I. Définition des coordonnées & importer le module

Nous allons utiliser le langage de programmation Python avec le module Image de la bibliothèque PIL (Pillow).

Pour vous ce sera transparent, la bibliothèque est déjà installée sous Jupyter notebook.

Avant de commencer à écrire un programme qui nous permettra de travailler sur les pixels d'une image, il est nécessaire de préciser que chaque pixel a des coordonnées **x** et **y**.

Comme vous pouvez le constater sur le schéma, le **pixel de coordonnées (0,0)** se trouve en **haut à gauche** de l'image.



Si l'image fait 800 pixels de large (**coordonnée x**) et 600 pixels de haut (**coordonnée y**), le pixel ayant pour coordonnées (400,300) sera au milieu de l'image.

- **En programmation Python**

Le début de votre programme devra commencer par l'importation d'un module spécial, cela se fera via la commande :

```
from PIL import Image
```

II. Modes d'une image et représentation des couleurs dans PIL.Image

Le mode d'une image est une chaîne de caractères qui indique quel est le codage couleur utilisé.

Les deux principaux **modes** que l'on va utiliser sont :

- **"L"** pour une seule bande en **nuance de gris** : une couleur est donnée par un entier compris entre **0 et 255**.
Par exemple **67** correspond à un gris clair.
- **"RGB"** pour le codage **couleur RGB** : une couleur est représentée par un tuple de trois entiers entre 0 et 255 (un pour chaque bande).
Par exemple **(255, 0, 0)** correspond à du rouge.

III. Ouvrir ou créer une nouvelle image à partir de la classe Image

Dans les cas les plus courants, deux méthodes s'offrent à vous :

- **Ouvrir un fichier image** situé dans le même dossier que le fichier python utilisé et l'appeler « image1 ».

```
image1 = Image.open("nom_fichier")
exemple: image1 = Image.open("./images/brest_1960.jpg")
```

L'image qui s'ouvrira sera « brest_1960 » située dans le dossier « images »



- **Créer une image** dans un mode couleur spécifié avec la couleur indiquée (la couleur par défaut est le noir).

```
image2 = Image.new(mode, taille, couleur)
exemple : image2 = Image.new("RGB", (500, 400), (0, 255, 128))
```

L'image aura comme définition 500x400 et sera vert clair



IV. Attributs et méthodes des images instances de la classe Image

Les images (instances de la classe Image) disposent des principaux attributs suivants (sans parenthèses lors de l'appel) :

- **.format** : indique le format du fichier image utilisé ("GIF", "JPG" ...). Renvoie None si l'image a été créée dans PIL.
- **.mode** : indique le mode couleur de l'image utilisé ("RGB", "L" etc.).
- **.size** : indique la taille de l'image sous forme d'un tuple, par exemple (1280, 960).

Et les images disposent aussi des principales méthodes suivantes (avec parenthèses lors de l'appel et, le cas échéant, les valeurs des arguments).

On fera attention car la règle générale pour ces méthodes est de retourner une image modifiée, ce qui préserve ainsi l'image de départ.

- **.save("nom_fichier", format)** : pour enregistrer l'image au format spécifié.
- **.convert("L")** : pour convertir une image en mode "L" (en nuances de gris).
- **.crop((x1, y1, x2, y2))** : pour retourner une image contenant les pixels situés dans le rectangle déterminé par les deux couples de coordonnées. Attention, les lignes et colonnes de pixels spécifiées par x2 et y2 ne seront pas extraites.
- **.resize((larg, haut))** : pour redimensionner une image aux dimensions spécifiées.
- **.rotate(theta)** : pour tourner une image d'un angle theta.
- **.putpixel(x, y, couleur)** : pour afficher un pixel de la couleur souhaitée à l'emplacement souhaitée.
- **.getpixel(x, y)** : pour obtenir la couleur du pixel de l'emplacement demandé.
- **print(nom)** : affiche ce que représente le nom

Pour afficher les résultats de votre programme :

- pour afficher votre résultat dans une fenêtre (celle qui ouvre les photos par défaut) : **.show()**
- pour afficher votre résultat dans la fenêtre Jupyter Notebook : **display(img)**

Exemple :

```
from IPython.display import Image
from PIL import Image

img = Image.new("RGB", (3, 3))
img.putpixel((0,0), (6,250,7))
img.putpixel((1,0), (241,252,23))
img.putpixel((2,0), (6,250,7))
img.putpixel((0,1), (126,14,203))
img.putpixel((1,1), (241,252,23))
img.putpixel((2,1), (6,250,7))
img.putpixel((0,2), (241,252,23))
img.putpixel((1,2), (6,250,7))
img.putpixel((2,2), (126,14,203))

r,v,b=img.getpixel((1,1))
print(" rouge:",r," vert:",v," bleu:",b)

code_couleur=img.getpixel((1,1))
print(code_couleur)

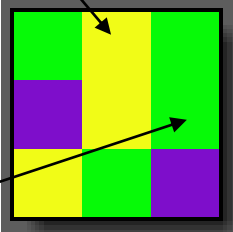
img.save(".\exemple_1.jpg")
display(img)
img.show()
```

Pixel (1, 0)
Red (241), Green (252), Blue (23)

Pixel (0, 1)
Red (126), Green (14), Blue (203)

Pixel (2, 1)
Red (6), Green (250), Blue (7)

Affiche le code RGB du pixel (1,1)
de 2 manières différentes



```
rouge: 241 vert: 252 bleu: 23
(241, 252, 23)
```