PREMIERE GENERALE

NUMERIQUE ET SCIENCES INFORMATIQUES

TRAITEMENT DES IMAGES

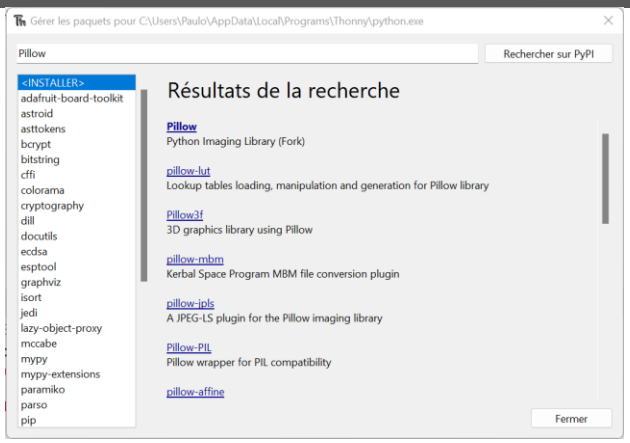
▷ Histoire de l'informatique
▷ Représentation des données : types et valeurs de base
▷ **Représentation des données : types construits**
▷ Architectures matérielles et systèmes d'exploitation
▷ Interactions entre l'homme et la machine sur le Web
▷ Architectures matérielles et systèmes d'exploitation
▷ **Langages et programmation**
▷ Algorithmique

1 – CHARGEMENT DU MODULE PIL

Lors de ce TP vous allez utiliser le module **`PIL`** (librairie **`Pillow`**) qui est une bibliothèque python permettant la **manipulation des images**.

Exercice n°1

1. **Sélectionner** « **`Gérer les paquets…`** » dans l'onglet « **`Outils`** ». Dans la barre de recherche **taper** « **`pillow`** » puis **cliquer** sur « **`Rechercher sur PyPI`** ». Dans la liste des modules **sélectionner** « **`Pillow`** » puis **cliquer** sur « **`Installer`** »



2. **Vérifier** que le module a été correctement installé en excutant l'instruction suivante.

```
>>> import PIL
```

2 – CHARGEMENT ET AFFICHAGE D'UNE IMAGE

Pour pouvoir charger et afficher une image, nous allons utiliser le sous-module **Image** de PIL.

```
from PIL import Image
```

Pour importer une image, il faut utiliser la méthode `Image.open()`. L'image doit être chargée dans une variable.

```
img = Image.open ("nom_fichier.ext")
```

Par exemple,l'image **Bordeaux.jpg** qui se trouve dans le dossier du script est importée et chargée dans la variable **img** :

```
img = Image.open("Bordeaux.jpg")
```

Pour afficher une image importée, il faut utiliser la méthode `show()` sur la variable **img** contenant l'image :

```
img.show()
```

Pour sauvegarder une image, il faut utiliser la méthode `save()` sur la variable **img** contenant l'image, en indiquant le nom du fichier et le type d'image.

```
img.save(nom_image, type)
```

Exercice n°2

1. **Editer** le script suivant permettant d'afficher l'image **Bordeaux.jpg**.

```
from PIL import Image
img = Image.open("Bordeaux.jpg")
img.save("photo.jpeg", "jpeg")
img.show()
```

2. **Exécuter** le sketch et **vérifier** son fonctionnement.

Il est possible de récupérer les principaux attributs de l'objet Image :

.format	Format de la source de l'image (si elle provient d'un fichier)
.size	Taille (en pixels) de l'image
.mode	Noms et type des différents canaux de l'image : RGB = 3 canaux ; L = 1 canal en niveaux de gris

Exercice n°3

1. **Editer** et **exécuter** le script suivant permettant d'afficher les attributs l'image **Bordeaux.jpg**.

```
from PIL import Image
img = Image.open("Bordeaux.jpeg")
print(img.format, img.size, img.size
img.show()
```

2. **Vérifier**, par rapport aux propriétés de l'image que la valeurs affichées sont correctes.

3 – DECOUPAGE ET REDIMENSIONNEMENT D'UNE IMAGE

La méthode **.crop((gauche, haut, droit, bas))** permet de découper une portion de l'image. Elle prend en paramètre un tuple contenant les 4 coordonnées **(gauche, haut, droit, bas)** en pixels de la zone à découper. Dans le système de coordonnées d'une image numérique, l'origine **(0, 0)** se situe en haut à gauche.

```
from PIL import Image
img = Image.open("Bordeaux.jpeg")
img_zone = img.crop((280, 80, 360, 220))
img_zone.save("photo.jpeg", "jpeg")
img_zone.show()
```

Exercice n°4

Modifier le script précédent afin de ne garder de l'image que le tramway.

La méthode **.resize(width, height))** permet de redimensionner l'image. Elle prend en paramètre un tuple contenant la largeur et la hauteur de la nouvelle image.

```
from PIL import Image
img = Image.open("Bordeaux.jpeg")
new_img = img.resize((200, 400))
new_img.save("photo.jpeg", "jpeg")
print(new_img.size)
new_img.show()
```

```
(200, 400)
```

Exercice n°5

Modifier le script précédent afin que l'image soit 2 fois plus petite.

4 – COLLAGE D'UNE IMAGE DANS UNE AUTRE

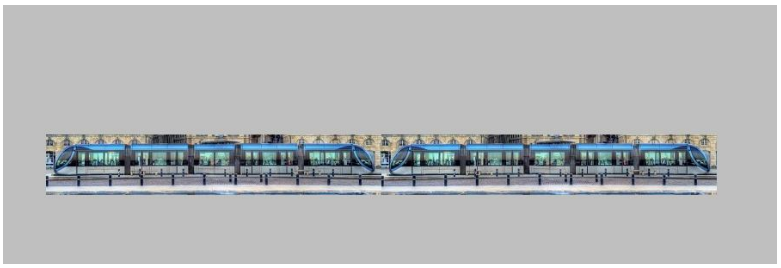
La méthode **.paste(image, (gauche, haut))** permet de coller une image sur une autre. Elle prend en paramètre l'image à coller et un tuple contenant les coordonnées du coin gauche haut à partir duquel sera collé l'image.

La methode **.new(mode, size, couleur)** permet de créer un nouvelle image en configurant le mode, les dimensions et la couleur.

```
from PIL import Image
img1 = Image.new("RGB", (900, 300), (191, 191, 191))
img2 = Image.open("Bordeaux.jpeg")
img_zone = img2.crop((120, 210, 510, 280))
img1.paste(img_zone, (50, 150))
img1.save("photo5.jpeg", "jpeg")
img1.show()
```

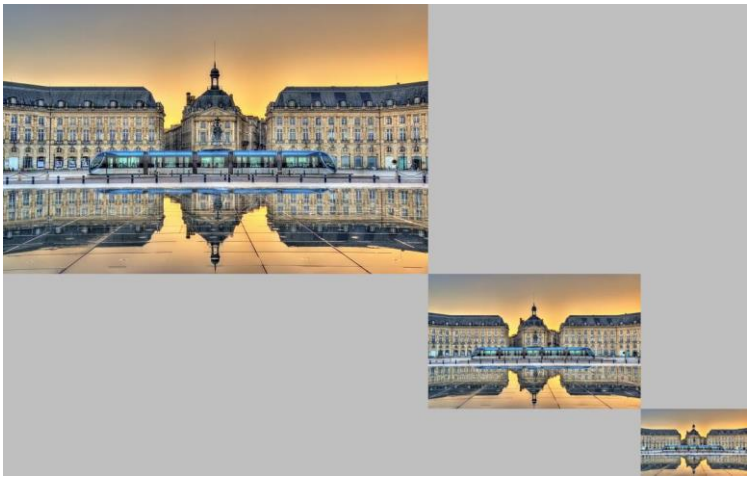
Exercice n°6

Editer un script permettant d'obtenir l'image suivante :



Exercice n°7

Editer un script permettant d'obtenir l'image suivante :

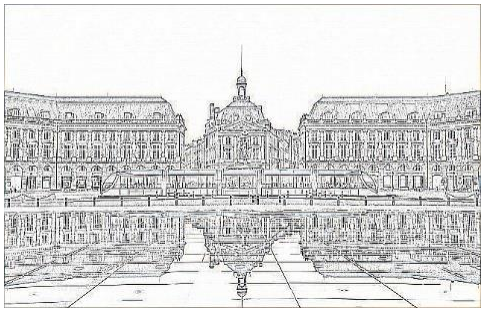


5 – FILTRES

PIL possède de nombreuses méthodes de traitement des images par filtrage. Les filtres sont contenus dans le sous module **`ImageFilter`** et applicable par la méthode **`.filter()`**. La description des différents filtres est disponible sur le site : https://pillow.readthedocs.io/en/stable/reference/ImageFilter.html.

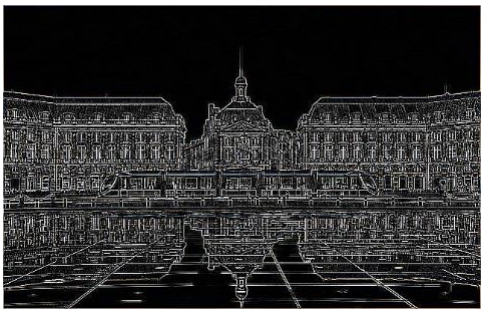
Par exemple les lignes de codes ci-dessous permettent de mettre en valeur les contours de l'image à l'aide du filtre **`CONTOUR`**.

```
from PIL import Image, ImageFilter
img = Image.open("Bordeaux.jpeg")
img = img.filter(ImageFilter.CONTOUR)
img.save("photo.jpeg", "jpeg")
img.show()
```



Exercice n°8

1. **Editer** un script permettant d'afficher uniquement les contours de l'image :



2. **Editer** un script permettant de flouter l'image :

6 – MODIFICATIONS DE GEOMETRIE SIMPLES

La methode **`.transpose()`** permet de **réaliser des rotations** (90°, 180° ou 270°) ou des **miroirs** (horizontal ou vertical) de l'image : https://pillow.readthedocs.io/en/stable/reference/Image.html.

Par exemple, les lignes de codes ci-dessous permettent de mettre de réaliser une rotation de **90°** de l'image.

```
from PIL import Image
img = Image.open("Bordeaux.jpeg")
img = img.transpose(Image.ROTATE_90)
img.save("photo.jpeg", "jpeg")
img.show()
```

Exercice n°9

Editer un script permettant de réaliser un miroir vertical de l'image :

7 – ACCES AUX PIXELS

7.1 – Récupérer les composantes R,G et B d'un pixel

La methode **`.getpixel((x, y))`** renvoie un tuple contenant les valeurs des composantes **R**, **G** et **B** du pixel défini par les coordonnées **x** et **y**.

Par exemple pour accéder aux composantes **R**, **G** et **B** du 1er pixel de l'image **`Bordeaux.jpeg`** :

```
from PIL import Image
img = Image.open("Bordeaux.jpeg")
R,G, B = img.getpixel((0, 0))
print(R, G, B)
```

```
122 135 152
```

Exercice n°10

Editer un script permettant d'afficher les composantes **R**, **G** et **B** du dernier pixel de l'image **`Bordeaux.jpeg`** :

7.2 – Modifier les composantes R,G et B d'un pixel

La methode **`.putpixel((x, y), (R, G, B))`** permet de modifier les valeurs des composantes **R**, **G** et **B** du pixel défini par les coordonnées **x** et **y**.

Par exemple pour colorier le 1er pixel de l'image **`Bordeaux.jpeg`**, en vert :

```
from PIL import Image
img = Image.open("Bordeaux.jpeg")
img.putpixel((0, 0), (0, 255, 0))
R,G, B = img.getpixel((0, 0))
print(R, G, B)
```

```
0 255 0
```

Exercice n°11

Editer un script permettant le dernier pixel de l'image **`Bordeaux.jpeg`**, en rouge.

8 – MODIFIER LES COULEURS D'UNE IMAGE

8.1 – Images en niveaux de rouge, vert ou bleu

Il possible de modifier les couleurs d'une image en modifiant la couleur de chacun des pixels, pour cela il faut parcourir l'ensemble des pixels au moyen de boucles **for**.

Pour garder uniquement le canal Vert par exemple, il suffit de colorier les différents pixels avec la couleur **(0, G, 0)** avec **G**, la composante verte initiale de chaque pixel.

Exercice n°12

1. **Compléter** le script permettant de ne conserver que les niveaux de vert.

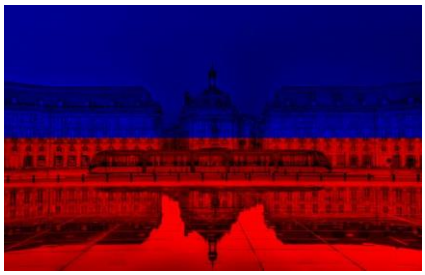
```
from PIL import Image
img = Image.open("Bordeaux.jpeg")
img.putpixel((0, 0), (255, 0, 0))
R, G, B = img.getpixel((0, 0))
print(R, G, B)
img = Image.open("Bordeaux.jpeg")
col, lig = img.size                          # dimensions de l'image RGB
for x in range(col):
    for y in range (lig):

                                             # On récupère le triplet (R, G, B)
                                             # On ne garde que la composante verte
img.save("photo.jpeg", "jpeg")
img.show()
```

2. **Modifier** le script permettant de ne conserver que les niveaux de rouge puis de bleu.

Exercice n°12

3. Modifier le script afin d'obtenir les images suivantes :



8.2 – Images en niveaux de gris

Pour passer d'une image couleur RGB à une image en niveau de gris, on utilise la formule :

$$NG = 0,299 \times R + 0,587 \times G + 0,114 \times B$$

Une image numérique en niveau de gris est représentée par **1 tableau à 2 dimensions.** Chaque pixel contenu dans l'image avec la valeur de la couleur **NG** étant un **nombre entier** compris entre 0 et 255 donc codé sur un octet.

Il n'est pas possible de convertir directement l'image RGB en image en niveau de gris. Il faut au préalable créer une nouvelle image en niveau de gris (`mode = 'L'`). Pour chacun des pixels de l'image RGB, on récupère les composantes **R**, **G** et **B** puis on calcul la composante **NG** et on colorie le pixel correspondant de l'image en niveau de gris avec cette composante **NG**.

Exercice n°13

Compléter le script permettant de convertir l'image en niveaux de gris.

```
from PIL import Image
img = Image.open("Bordeaux.jpeg")
col, lig = img.size                          # dimensions de l'image RGB
img_NVG = Image.new('L', (col, lig))          # Création d'une image en niveaux de gris
for x in range(col):
    for y in range (lig):

                                             # On récupère le triplet (R, G, B) de img
                                             # On calcul la composante NG
                                             # On colorie le pixel avec NG
img_NVG.save("photo.jpeg", "jpeg")
img_NVG.show()
```

8.3 – Images en noir et blanc

Dans une image numérique en noir et blanc ou **monochrome**, chaque pixel ne possède que **deux couleurs** : noir et blanc. Chaque pixel est alors codé sur **1 bit**.

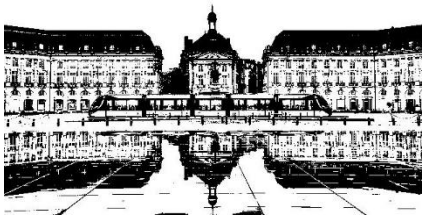
Pour obtenir une image en noir et blanc à partir d'une image RGB, il faut effectuer un **seuillage**. Pour ce faire, nous choisissons un seuil (par exemple 127).

- Si la moyenne des trois composantes dépasse la valeur du seuil, le pixel prendra la couleur blanche c'est-à-dire la valeur **1**.
- Sinon le pixel prendra la couleur noire c'est-à-dire la valeur **0**.

Il n'est pas possible de convertir directement l'image RGB en image en noir et blanc. Il faut au préalable créer une nouvelle image en noir et blanc (`mode = '1'`). Pour chacun des pixels de l'image RGB, on récupère les composantes **R**, **G** et **B** puis on calcule la moyenne de ces trois composante, on la compare au seuil et on colorie le pixel correspondant de l'image noir et blanc en fonction du résultat de la comparaison.

Exercice n°14

Compléter le script permettant de convertir l'image en noir et blanc.



```
from PIL import Image
img = Image.open("Bordeaux.jpeg")
col, lig = img.size                          # dimensions de l'image RGB
img_NB = Image.new('1', (col, lig))          # Création d'une image en noir et blanc
for x in range(col):
    for y in range (lig):
                                        # On récupère le triplet (R, G, B) de img
                                        # On calcule la moyenne des trois composantes

                                        # On colorie le pixel en blanc

                                        # On colorie le pixel en noir
img_NB.save("photo.jpeg", "jpeg")
img_NB.show()
```

Exercice n°15

A partir du script précédent, **éditer** un script permettant d'obtenir l'image ci-dessous. Attention la nouvelle image est une image RGB (`mode = 'RGB'`).