FICHE BILAN DE L'ATELIER 4 : PYTHON ET COULEURS (2 pages)

LA BIBLIOTHEQUE ou MODULE PILLOW contenant la classe IMAGE

MODULE PILLOW
(anciennement PIL)

Il faut trouver le chemin d'accès du fichier pip.exe. Par exemple :
C:\Users\winadmin\AppData\Local\Programs\Python\Python35-32\Scripts

Il faut alors donner à l'ordinateur le chemin d'accès vers ce dossier.
Soit à la main en ouvrant la console et en partant à la main dans le répertoire (avec la commande cd).
Soit en créant une nouvelle entrée dans la variable Path (chemin) :
set Path=%Path% ; C:\Users\winadmin\AppData\Local\Programs\Python\Python35-32\Scripts
(on rajoute alors ce chemin définitivement sur l'ordinateur)

Si pip n'est pas à jour, on peut taper cela dans la console de l'ordinateur (pas de Python) :
pip install -- upgrade pip
Sinon, il suffit de taper cela dans la console : **pip install Pillow**

Importation d'un module

```
from PIL import Image as Img
```

Importe la classe Image depuis la bibliothèque Pillow/PIL.
On donne un alias à cette classe à l'aide de Img pour aller plus vite (voir exemple avec open).

Ouverture
d'un objet-image
à partir
d'un fichier

```
x = Img.open( "linux.jpg" )
```

On crée ainsi un objet-image nommé x qui contient l'image codée dans le fichier dont on donne l'adresse en chemin relatif : cela veut dire que le chemin est donné à partir du dossier dans lequel est votre fichier Python .py

Création
D'un objet-image
vide

```
x = Img.new("RGB",(largeur_voulue, hauteur_voulue),(R,G,B))
```

On crée ainsi un objet-image nommé x dont tous les pixels ont la même couleur (R,G,B).

```
x = Img.new("L",(largeur_voulue, hauteur_voulue), intensite)
```

Idem mais avec une image grisée dont l'intensité est fournie.

Quelques METHODES
(fonctions qu'on applique sur un objet)

qui donnent les valeurs des **attributs** de l'objet.

Attention : x doit être la référence d'un objet-image

```
x.width     → Renvoie la largeur en pixels
x.height    → Renvoie la hauteur de l'image
x.size      → Renvoie un 2-uplets contenant (x.width, x.height)
```

```
x.format
```

→ Renvoie le format d'**enregistrement** du fichier-image associé à x (jpg, png…)

```
x.getbands()
```

→ Renvoie le nombre de couches et leurs noms :

- 3-uplets ('R','G','B') si image couleur
- 'L' si image est grisée (**niveaux de gris**) ou s'il s'agit de l'une des couleurs de couleurs.

```
x.getpixel( (largeur,hauteur) )
```

→ Renvoie les informations sur la couleur du pixel dont les coordonnées sont définies en largeur , hauteur de pixels par rapport au côté **en haut à gauche** de l'image. Attention, si l'une possède 3 couches, l'attribut donnera les intensités de RGB sous forme d'une 3-uplets (intensité rouge, intensité verte, intensité bleu). Sinon, il renvoie juste l'intensité si l'image est grise ou qu'il s'agit d'une des couches de couleur.

Exemples :

```
larg = x.width              # Stocke la largeur dans la variable larg
haut = x.size               # Stocke la hauteur dans la variable haut
largeur,hauteur = x.size    # ou en utilisant le 2-uplets x.size
```

Quelques METHODES (fonctions qu'on applique sur un objet)
 	
qui modifient l'objet
	`x.show()` → Affiche l'objet-image à l'écran.
	
	`z = x.resize((1000,1000))`
	→ Crée un objet-image z à partir de x en modifiant la taille en pixels (largeur, hauteur).
	
	`x.save( "nom.jpg" )`
	→ Sauvegarde l'objet-image dans un fichier-image dont nom et extension sont fournis.
	
	`x.putpixel((largeur,hauteur),(r,g,b) )`
	→ Si objet RGB, modifie le 3-uplet vers (r,g,b) la couleur du pixel aux coordonnées.
	
	`x.putpixel((largueur,hauteur),i)`
	→ Si objet L, modifie l'intensité du pixel de coordonnées **(largeur,hauteur).**
	
	`xr, xg, xb = x.split()`
	→ Crée trois objets-image L contenant les couches de couleur R,G,B de l'objet-image x. xr est donc ici un objet-image contenant la couche rouge de l'objet-image x.
	
	`x.paste(z, (largeurVoulue,hauteurVoulue) )`
	→ Colle l'objet-image z sur l'objet-image x aux coordonnées données.
	
	`x.paste( z, (largeurVoulue, hauteurVoulue, largeurBoite, hauteurBoite) )` (hors activité)
	→ Colle l'objet-image z sur l'objet-image x aux coordonnées données mais en se limitant aux dimensions de la boite.
	
	`x.paste((r,g,b),(larg_voulue,hauteur_voulue,larg_boite,hauteur_boite))` (hors activité)
	→ Comme ci-dessus mais avec une couleur plutôt qu'une image.
	
	`z = x.convert( "L" )` (hors activité)
	→ Crée un objet-image z de type L en convertissant l'objet-image x en grisé.
	
	`z = x.convert( "1" )` (hors activité)
	→ Crée un objet-image z de type L en convertissant l'objet-image x en noir ou blanc.

Création d'un objet-image à partir des trois couches 'L' RGB	`z = Img.merge( "RGB" , (xr,xg,xb) )` Crée l'objet-image z en mode 3 couches RGB en utilisant les trois objets-image en tant que couches R, G et B dans l'ordre. Attention, les trois sous-couches doivent avoir les mêmes dimensions et de type « L », sinon ça ne peut pas fonctionner.

LA BIBLIOTHEQUE ImageChops : *channel operations ("chops")*
LA BIBLIOTHEQUE ImageOps : opérations

Voir sur Internet : on peut mélanger deux images …

Importation d'un module et d'une classe en particulier	`from PIL import ImageChops as ImgChp`
	`from PIL import ImageOps as ImgOp`
	Importe la classe **ImageChops** contenue dans le module Pillow/PIL.
	On appelle la classe à l'aide de **ImgChp** pour aller plus vite (voir exemple avec open).

Quelques METHODES qui peuvent s'appliquer sur un objet-image ImgChp	`x_r = ImgChp.invert(x_r)`
	→ Modifie l'intensité des pixels en les inversant : nouvelle intensité = 255 – ancienne
	`x_r = ImgOp.colorize( x_r, (0,0,0) , (255,0,0) )`
	→ Si x_r est une image grisée 'L', on colorie celle-ci à partir des deux nouvelles couleurs RBG qu'on utilise pour l'ancien 0 et l'ancien 255.