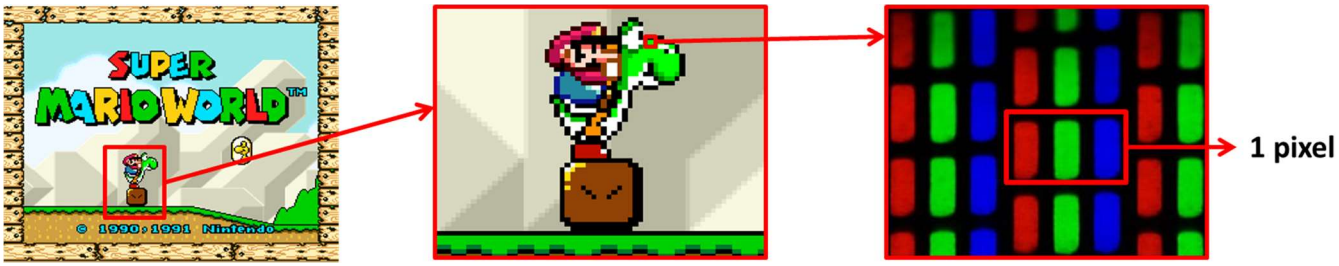
Traitement d'images avec corrigé

1. L'image en informatique

Une image est composée de petits points appelés pixel. La définition d'une image donne le nombre de pixels qui compose l'image, par exemple une image de définition 800 x 600 (800 par 600), signifie que cette image est composée de 800 pixels en largeur et de 600 pixels en hauteur, soit en tout 800 x 600 = 480 000 pixels.

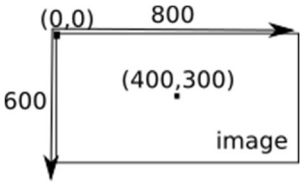
Un pixel est composé de trois parties : une partie rouge, une partie verte et une partie bleue. À chaque pixel on associe donc 3 couleurs : le rouge, le vert et le bleu. On parle de composantes rouge, vert ou bleu d'un pixel (on parle de système RVB ou RGB en anglais). La théorie physique de la synthèse additive des couleurs montre que la variation de l'intensité lumineuse de chaque composante permet d'obtenir un très grand nombre de couleurs. La valeur de l'intensité lumineuse associée à chaque composante de chaque pixel d'une image est codé sur 8 bits (un octet), chaque composante peut donc voir son intensité lumineuse variée de 0 à 255 (1 octet => 256 valeurs). On codera donc un pixel à l'aide d'un triplet de valeur (par exemple "247, 56, 98"). La première valeur donnant l'intensité de la composante rouge, la deuxième valeur donnant l'intensité de la composante verte et la troisième valeur donnant l'intensité de la composante bleue.



Quand on observe un pixel "à la loupe", on peut constater que le pixel est bien constitué de trois parties : une partie rouge, une partie verte, et une partie bleue (voir schéma ci-dessus).
Un pixel est tellement petit que l'œil humain superpose la partie rouge, la partie verte et la partie bleue du pixel, ceci explique pourquoi l'œil voit des pixels de différentes couleurs.

Le nom **pixel**, souvent abrégé **px**, provient de la locution anglaise "**pict**ure **el**ement", qui signifie "élément d'image".

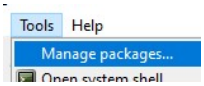
Dans une image, chaque pixel a des coordonnées x,y.
Par exemple sur le schéma ci-contre, le pixel de coordonnées (0, 0) se trouve en haut à gauche de l'image. Si l'image fait 800 pixels de large et 600 pixels de haut, le pixel ayant pour coordonnées (400, 300) sera au milieu de l'image.

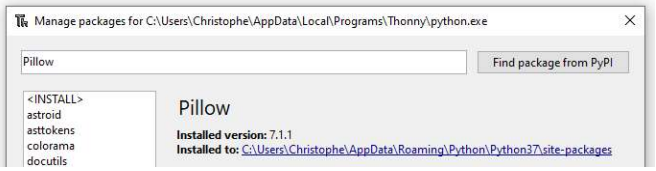


2. La bibliothèque Pillow

Le traitement d'image, avec Python, va se faire à l'aide de la bibliothèque Pillow, qui est une bibliothèque de traitement d'images. Par conséquent il est nécessaire d'effectuer son installation.

Dans Thonny, depuis le menu *Tools*, sélectionner *Manages packages…*, puis rechercher la bibliothèque Pillow et l'installer.





3. Créer une image

✍ Tester le programme ci-dessous qui permet d'obtenir l'image ci-dessous.

```
# import du module Image de la bibliothèque Pillow
from PIL import Image
# création d'une nouvelle image (new_im)
# codée avec le code RGB de taille 100,200
# par défaut la couleur de fond (background) est noire
new_im = Image.new('RGB', (100,200))
# enregistrement de l'image au format PNG
# dans le répertoire où se situe le programme
new_im.save("MonImage.png", "PNG")
```

Pour avoir une image avec un fond de couleur donné il faut donner en argument le code RGB de la couleur.

```
from PIL import Image
new_im = Image.new('RGB', (200,200), (255,0,0))
new_im.save("MonImage.png", "PNG")
```

✍ Créer une image avec un fond vert de dimension 50 par 20.

4. Agir sur une image

✍ Placer l'image de la pomme (pomme.jpg) dans le même répertoire que les programmes précédents.

4.1. Lire un pixel

✍ Tester le programme ci-dessous :

```
from PIL import Image
img = Image.open("pomme.jpg")
r,v,b = img.getpixel((100,250))
print(f'composante rouge : {r}, composante verte : {v}, composante bleue : {b}')
```

- L'instruction **`getpixel()`** permet de récupérer les composantes rouge, verte et bleue du pixel de coordonnées (100, 250) de l'image "pomme.jpg".

✍ Quelles sont les composantes de couleur du pixel (200, 200) ?

4.2. Modifier un pixel

✍ Tester le programme suivant :

```
from PIL import Image
img = Image.open("pomme.jpg")
img.putpixel((250,250),(255,0,0))
img.show()
```

Au centre de l'image apparait un pixel rouge (il faut zoomer si besoin).

- L'instruction **`putpixel()`** permet de colorier le pixel (250, 250) en rouge (255, 0, 0).
- **`show()`** permet d'afficher l'image **`img`**.

4.3. Parcourir une image

✍ Tester le programme suivant :

```
from PIL import Image
img = Image.open("pomme.jpg")
largeur, hauteur = img.size
for y in range(hauteur):
    for x in range(largeur):
        r, v, b = img.getpixel((x, y))
        img.putpixel((x, y), (v, b, r))
img.show()
```

Le résultat attendu est ci-contre.
L'instruction **`size`** donne un tuple correspondant aux dimensions de l'image associé (ici `img`).

Les deux boucles `for` permettent de parcourir l'ensemble des pixels de l'image.

```
for y in range(hauteur):
    for x in range(largeur):
```

Les variables x et y permettent de parcourir l'ensemble des pixels de l'image :

- Le programme démarre avec le pixel de coordonnées (0, 0) ;
- puis le pixel de coordonnées (1, 0) ;
- puis le pixel de coordonnées (2, 0) ;
- jusqu'au pixel de coordonnées (largeur -1, 0).
- Ensuite, cela change de ligne avec le pixel de coordonnées (0, 1) ;
- puis le pixel de coordonnées (1, 1) ;
- et ainsi de suite jusqu'au pixel de coordonnées (largeur – 1, hauteur – 1).

Pour chaque pixel lu, les composantes r, v et b sont changées (le rouge est remplacé par le vert, le vert par le bleu et le bleu par le rouge).

✍ Modifier le programme ci-dessus afin d'inverser les composantes rouge et verte en conservant la composante bleue. Le résultat à obtenir est ci-contre.

5. Corrigé

✍ Créer une image avec un fond vert de dimension 50 par 20.

```
from PIL import Image
new_im = Image.new('RGB', (50,20), (0,255,0))
new_im.save("MonImage.png", "PNG")
```

✍ Quelles sont les composantes de couleur du pixel (200, 200) ? (160,200,75)

✍ Modifier le programme ci-dessus afin d'inverser les composantes rouge et verte en conservant la composante bleue. Le résultat à obtenir est ci-contre.

```
img.putpixel((x, y), (v, r, b))
```