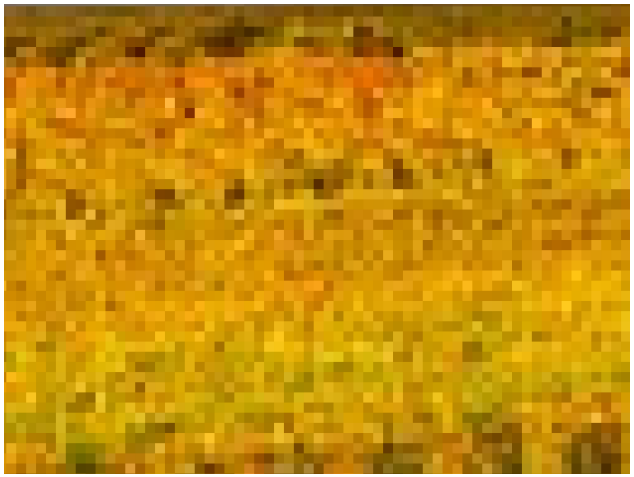
Construire une image numérique Introduction à Python

Christophe Viroulaud

Seconde - SNT

Phot 02

Une image numérique est composée de points colorés appelés **pixels**.



Pour construire une image numérique il suffit d'aligner suffisamment de points.

Comment construire une image numérique par programmation ?

Sommaire

Stocker une image en mémoire

Pour pouvoir utiliser des données dans un programme, il faut les stocker dans une variable.

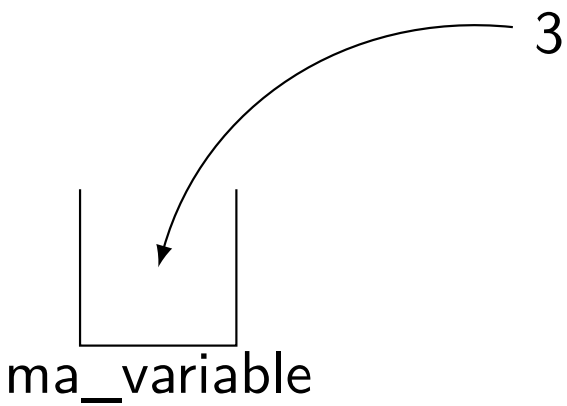


FIGURE 1 – Affectation

```
1 ma_variable = 3
```

Code 1 – Créer une variable en Python

```
1 # Créer une variable 'image'
2 image = Image.new('RGB', (800, 600), (255, 255, 255))
```

Code 2 – Stocker une image blanche

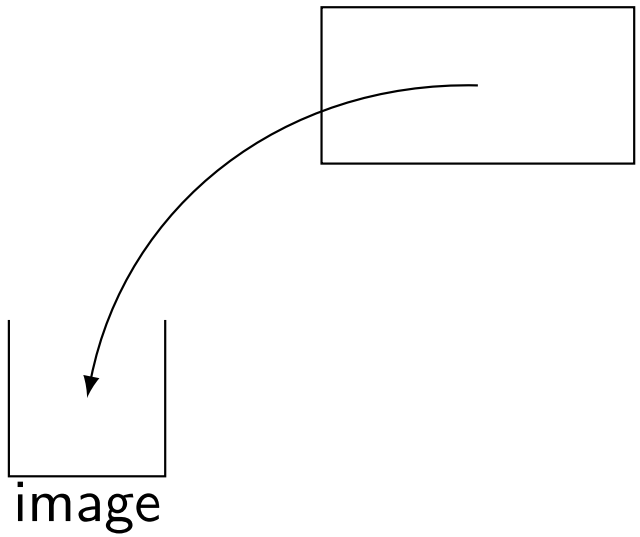


FIGURE 2 – Affecter une image vide dans `image`

Activité 1 :

1. Ouvrir le logiciel *Spyder*.
2. Écrire le code 3 dans la partie gauche.
3. Enregistrer le programme dans le dossier **SNT** sous le nom **`mon_image.py`**

```
1 # Bibliothèque de gestion des images
2 from PIL import Image
3 # Créer une variable 'image'
4 image = Image.new('RGB', (800, 600), (255, 255, 255))
5 # Afficher l'image
6 image.show()
```

Code 3 – Afficher l'image

Sommaire

Répéter une opération

Pour modifier l'image blanche, il faut poser des pixels d'une autre couleur.

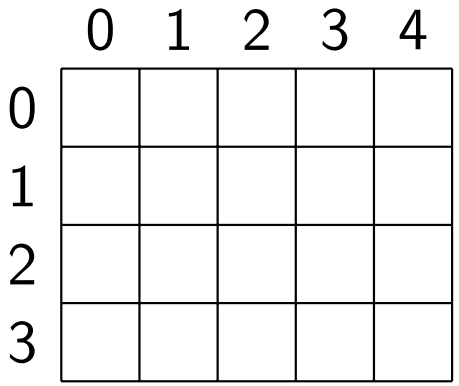


FIGURE 3 – Coordonnées d'un pixel

```
1 from PIL import Image
2 image = Image.new('RGB', (800, 600), (255, 255, 255))
3 # Poser un pixel noir en (10,10)
4 image.putpixel((10,10),(0,0,0))
5 image.show()
```

Code 4 – Poser un pixel

Activité 2 : Poser plusieurs pixels noirs à côté du premier jusqu'à voir une forme sur l'image.

Correction

```
1  image.putpixel((10,10),(0,0,0))
2  image.putpixel((11,10),(0,0,0))
3  image.putpixel((12,10),(0,0,0))
4  image.putpixel((13,10),(0,0,0))
5  image.putpixel((11,10),(0,0,0))
6  image.putpixel((11,11),(0,0,0))
7  image.putpixel((11,12),(0,0,0))
8  image.putpixel((11,13),(0,0,0))
9  image.putpixel((12,10),(0,0,0))
10 image.putpixel((12,11),(0,0,0))
11 image.putpixel((12,12),(0,0,0))
12 image.putpixel((12,13),(0,0,0))
```

Code 5

```
1 for x in range(100):
2     image.putpixel((x,10),(0,0,0))
```

Code 6 – Répéter une opération

Activité 3 :

1. Remplacer les ajouts manuels de pixels par le code 7.

```
1 for x in range(100):
2     image.putpixel((x,10),(0,0,0))
```

Code 7 – Répéter une opération

2. Modifier le code pour tracer un trait sur toute la largeur de l'image.
3. Créer une nouvelle boucle pour tracer un trait vertical.
4. Tracer un trait oblique.

Correction

```
1  # horizontal
2  for x in range(800):
3      image.putpixel((x,10),(0,0,0))
4
5  # vertical
6  for y in range(600):
7      image.putpixel((400,y),(0,0,0))
8
9  # oblique
10 for y in range(600):
11     image.putpixel((y,y),(0,0,0))
```

Code 8 – Tracé de trois traits

Sommaire

Fonction

```
1  for x in range(800):
2      image.putpixel((x,10),(0,0,0))
3
4  for x in range(800):
5      image.putpixel((x,20),(0,0,0))
6
7  for x in range(800):
8      image.putpixel((x,30),(0,0,0))
9
10 for x in range(800):
11     image.putpixel((x,40),(0,0,0))
```

Code 9 – Répétition de code

Pour éviter de répéter du code on peut utiliser une **fonction**.

marteau(petit)

marteau(moyen)

marteau(gros)

À retenir

Une fonction est un outil qui possède des **paramètres** et que l'on peut réutiliser dans le programme.

```
1 def trait_horizontal(position):
2     for x in range(800):
3         image.putpixel((x,position),(0,0,0))
```

Code 10 – Construction de la fonction

```
1 trait_horizontal(10)
```

Code 11 – Appel de la fonction

```
1  from PIL import Image
2
3  def trait_horizontal(position):
4      for x in range(800):
5          image.putpixel((x,position),(0,0,0))
6
7  image = Image.new('RGB', (800, 600), (255, 255, 255))
8
9  trait_horizontal(10)
0
1  image.show()
```

Code 12 – Afficher l'image

À retenir

On crée la fonction en début de programme puis on l'utilise quand on le souhaite.

Activité 4 :

1. Écrire la fonction **`trait_vertical(position)`** qui trace un trait vertical.
2. Dans le programme principal, écrire le code 13.

```
1 for y in range(0,600,10):
2     trait_horizontal(y)
```

Code 13

3. En s'appuyant sur le code 13, tracer des lignes verticales tous les 10 pixels.

Correction

```
1 def trait_vertical(position):
2     for y in range(600):
3         image.putpixel((position,y),(0,0,0))
```

Code 14 – Fonction pour un trait vertical

```
1 for x in range(0,800,10):
2     trait_vertical(x)
```

Code 15 – Traits verticaux

Activité 5 :

1. Modifier la fonction `trait_horizontal(position, couleur)` pour qu'elle trace un trait de la couleur désirée.
2. Écrire la fonction `carre(o_x, o_y, longueur)` qui trace un carré dont le sommet haut gauche est en `(o_x, o_y)` et de côté `longueur`.

Correction

```
1 def trait_horizontal(position, couleur):
2     for x in range(800):
3         image.putpixel((x, position), couleur)
```

Code 16 – Trait de couleur

Correction

```
1 def carre(o_x, o_y, longueur):
2     for x in range(o_x, o_x+longueur):
3         for y in range(o_y, o_y+longueur):
4             image.putpixel((x, y), (0, 0, 0))
```

Code 17 – Carré