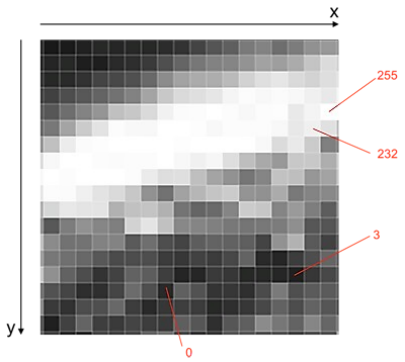
Activité 11a : Traitement d'images noir et blanc

L'objectif de cette activité est de comprendre le codage d'une image numérique et de manipuler des images en niveaux de gris (appelée abusivement « noir et blanc ») afin d'effectuer des traitements automatiques.

1. Les images "en noir et blanc"

✗ **Ouvrir** le fichier nb.png et **regarder** l'image obtenue, en zoomant sur l'image. **Observer** qu'elle est constituée d'une multitude de pixels organisés en lignes (y) et colonnes (x).

Pour une image au format PGM, le niveau de gris de chaque pixel est codée sur un octet (c'est-à-dire 8 bits), et représente donc une nuance de gris parmi 255 valeurs possibles. On peut voir cette valeur comme la luminosité du pixel: la valeur 0 code la couleur noire, la valeur 255 code la couleur blanche, et les valeurs intermédiaires codent des nuances de gris de plus en plus claires à mesure que la valeur augmente.



Zoom d'image en niveau de gris

2. Traitement d'une image sous Python

Dans l'environnement Python, on peut charger en mémoire une image à partir d'un fichier. Elle sera alors considérée comme un objet sur lequel on pourra agir en lui appliquant des fonctions et méthodes.

Code de base pour afficher une image

Le code de base suivant permet de charger dans la variable img, la nuance de gris de chaque pixel de l'image imageSimple.png. Cette image ne comporte volontairement que très peu de pixels afin de faciliter la compréhension du traitement automatique d'images.

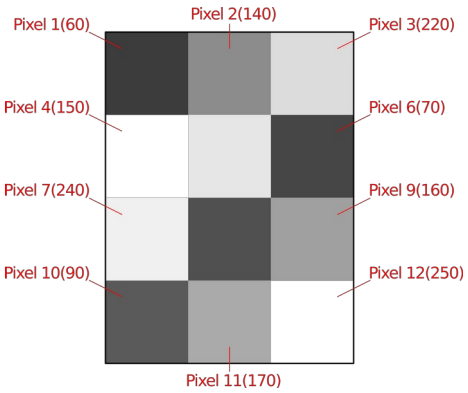


Figure 1: Contenu de l'image imageSimple.png

```
from manipulationImage import *#Importer la bibliothèque traitement
img=ouvrirImage("imageSimple.png") #Charger l'image dans img img.
        #img->liste contenant la nuance de gris de chaque pixel
afficherImage(img)                          #Afficher l'objet
```

✗ **Saisir** le programme précédent dans l'IDE Python. **Exécuter** le programme puis **manipuler** dans la console la variable img afin de compléter le tableau suivant.

Commande exécutée	len(img)	img[1][0]	img[1][2]	img[2][0]	img[1]	img[0]	len(img[0])
Résultat obtenu	4	150	70	240	[150, 230, 70]	[60, 140, 0]	3
Action réalisée	Nombre de lignes de l'image	Lire la nuance de gris du pixel 4	Lire la nuance de gris du pixel 6	Lire la nuance de gris du pixel 7	Extraire la deuxième ligne	Extraire la première ligne	Largeur de l'image en nombre de pixels

✗ **Ecrire** les instructions qui réalisent les actions suivantes puis **vérifier** le résultat:
- Colorier en blanc le pixel 0 → img[0][0] = 255
- Colorier en noir le pixel 8 → img[2][1] = 0
- Eclaircir de 10 nuances le pixel 4 → img[1][0] = img[1][0] +10

Balayage d'une image

Balayer une image consiste à pointer successivement chaque pixel afin d'effectuer si besoin un traitement. Ce balayage peut être implémenté en utilisant une double boucle pour.

Le code suivant efface une image en coloriant chaque pixel en blanc :

```
1  from manipulationImage import *
2  img=ouvrirImage("imageSimple.png")
3  hauteur=len(img)
4  largeur=len(img[0])
5  for ligne in range(hauteur) :
6          for colonne in range(largeur) :
7              img[ligne][colonne]=255
```

✗ A la lecture de ce programme, **lister** dans l'ordre de modification, les numéros des pixels modifiés

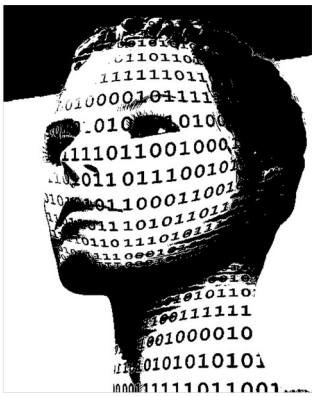
3.Traitements automatiques d'une images réelle

Cette partie a pour objectif de programmer différents traitements d'images numériques. Les différents programmes seront testés sur l'image nb.png disponible sur le réseau.

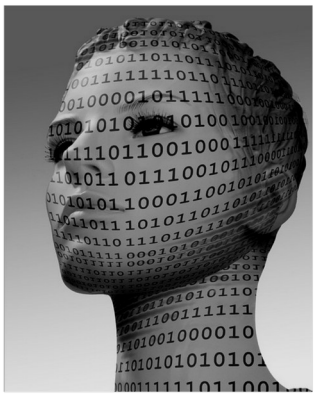
- ✗ **Proposer** puis **tester** un programme Python qui affiche le négatif d'une image (Traitement 1).
- ✗ **Proposer** puis **tester** un programme Python qui affiche une image en noir et blanc à partir d'une image en niveau de gris. (Traitement 2).
- ✗ **Proposer** puis **tester** un programme Python qui assombrit une image (Traitement 3).
- ✗ **Proposer** puis **tester** un programme Python qui trace une ligne blanche horizontale au milieu de l'image (Traitement 4).

Image d'origine

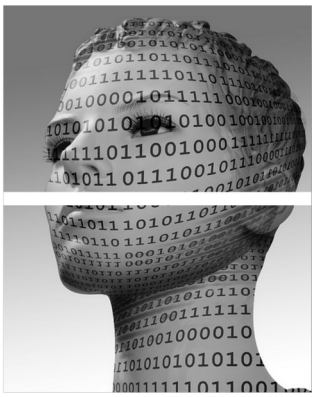
Traitement 1



Traitement 2



Traitement 3



Traitement 4