

Pixel Art avec Python

1. Se préparer pour l'activité

1. Ouvrez le logiciel **Thonny** depuis le menu **Windows**.
2. Dans le menu **Tools>Manage packages**, tapez dans la barre de saisie le mot **pillow** puis cliquez sur **Find package from PyPi**.



Puis cliquez sur **Install** (si vous voyez à la place les boutons **Upgrade** et **Uninstall**, ce n'est pas nécessaire). Laissez le logiciel installer jusqu'au bout, cela prend quelques secondes.

3. Ouvrez l'explorateur de fichier **windows**, puis allez dans le dossier **Images**. Créez un sous-dossier avec votre nom et prénom. Ce dossier sera celui dans lequel vous devrez copier **tous les fichiers de cette activité**.
4. Rendez-vous sur le site www.zonensi.fr, et cliquez sur **Activite Pixel Art**. Enregistrez tous les fichiers suivants dans votre dossier personnel.
 - [PixelArt.py](#)
 - [makeMatrix.py](#)
 - [bigMatrix.txt](#)

2. Du Pixel Art en Noir et Blanc

Pixels et tableaux de nombres

Une image numérique est composée de pixels. Ces pixels sont les plus petites « lampes » allumables par l'ordinateur. Pour demander à l'ordinateur d'allumer un pixel, on lui donne une valeur numérique. Une image numérique est donc représentée dans un ordinateur par un tableau de nombres, appelé **matrice** en français (*matrix* en anglais).

1. Voici une image en noir et blanc de 8 pixels de long, et de 8 pixels de haut, ainsi que sa représentation sous la forme d'une matrice dans le langage Python :



```
matrix=[
    [255, 0, 0, 255, 255, 0, 0, 255],
    [0, 0, 0, 0, 0, 0, 0, 0],
    [0, 0, 0, 0, 0, 0, 0, 0],
    [0, 0, 0, 0, 0, 0, 0, 0],
    [255, 0, 0, 0, 0, 0, 0, 255],
    [255, 255, 0, 0, 0, 0, 255, 255],
    [255, 255, 255, 0, 0, 255, 255, 255],
    [255, 255, 255, 255, 255, 255, 255, 255],
]
```

A quoi correspondent les valeurs $\backslash(0\backslash)$ et $\backslash(255\backslash)$ dans la matrice ?

- Ouvrir le logiciel Thonny, et depuis ce logiciel ouvrez le fichier `pixelArt.py` (avec le menu `File>open`).
- Cliquez sur la petite flèche verte pour exécuter le script Python.



Qu'obtient-on ?

- La ligne créant l'image et l'affichent à partir de la matrice est la ligne 35 :

```
makeImage(matrix, 8, 8)
```

Dans cette ligne, la fonction `makeImage` utilise la variable `matrix`, pour créer une image de dimension $\backslash(8 \times 8\backslash)$. Remplacer cette ligne par la ligne suivante :

```
makeBiggerImage(matrix, 8, 8, scale=5)
```

Qu'obtient-on ?

- Sauvegarder cette nouvelle image dans le même dossier que le fichier `pixelArt.py` sous le nom `greatHeart.png`. Combien de pixels composent cette image ?
- Comment faire pour obtenir une image de $\backslash(80 \times 80\backslash)$ pixels ?
- A l'aide des grilles suivantes, créez deux nouvelles images de dimension $\backslash(8 \times 8\backslash)$ (vous pouvez vous inspirer du site www.pixelart.name).



- Modifier le programme Python afin d'afficher vous deux images en noir et blanc, en taille $\backslash(80 \times 80\backslash)$, et sauvegardez-les.
- Combien de nombres sont nécessaires pour une image de $\backslash(16 \times 16\backslash)$ pixels ? de $\backslash(32 \times 32\backslash)$ pixels ? de $\backslash(64 \times 64\backslash)$ pixels ?

3. Une image en niveau de gris

- Ouvrez avec le bloc-note le fichier `bigMatrix.txt` et copiez-collez la totalité de ce texte dans le fichier `pixelArt.py`, à la place de la variable `matrix`.
- Cette nouvelle variable `bigMatrix` est une matrice de dimension $\backslash(16 \times 16\backslash)$. Mis à part sa taille et son nom, qu'a-t-elle de nouveau par rapport à la variable précédente ?
- Modifiez la ligne `makeBiggerImage...` de la manière suivante :

```
makeBiggerImage(bigMatrix, 16, 16, scale=5)
```

Puis exécutez le code. Qu'obtient-on ?

Les niveaux de gris

Nous avons vu qu'un pixel peut être « allumé » quand on lui donne la valeur $\backslash(255\backslash)$, et qu'il est « éteint » quand on lui donne la valeur $\backslash(0\backslash)$. Mais en réalité on peut lui donner n'importe quelle valeur entre $\backslash(0\backslash)$ et $\backslash(255\backslash)$, ce qui donnera un **niveau de gris**, sachant que plus la valeur est grande, plus la nuance se est éclaircie, et plus elle est proche de $\backslash(0\backslash)$ plus elle est assombrie.

4. Votre propre matrice

1. Allez chercher sur le web une image - pas trop grande en dimension, et sauvegardez-là dans votre dossier.
2. Ouvrez dans Thonny le fichier `makeMatrix.py`.
3. Modifiez la ligne $\backslash(11\backslash)$ en remplaçant le nom `Mario.png` par le nom de votre image (ou par le nom que vous choisissez). Attention à bien conserver les guillemets !
4. Exécutez alors le code. Qu'obtenez-vous ?
5. Copiez-collez la matrice obtenue à la place de la précédente dans le fichier `pixelArt.py`, puis modifiez le fichier afin de pouvoir l'exécuter. Attention, il vous faudra connaître le nombre de pixels de cette image pour pouvoir exécuter correctement le code !