A. Introduction au pixel art

E.1290 *(Ceci n'est pas un exercice)*

E.1138

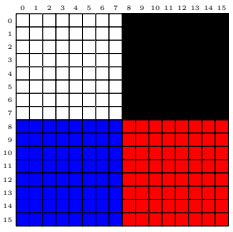
La bibliothèque `PIL` *(Portable Image Library)* permet de créer et de gérer des images sous Python.

1. Téléchargez le code ci-dessous et interprétez les double-boucle et la fonctionnalité de `putpixel()`

```
from PIL import Image
image = Image.new("RGB",(16,16));
for i in range(0,5):
    for j in range(0,5):
        image.putpixel((10+i,10+j),(255,0,0))
for i in range(0,10):
    for j in range(0,2):
        image.putpixel((2+i,2+j),(255,255,0))
for i in range(0,4):
    for j in range(0,8):
        image.putpixel((2+i,8+j),(255,255,255))
image.show()
```

1138-code.py

2. Modifiez ce code pour obtenir l'image ci-dessous :



E.478

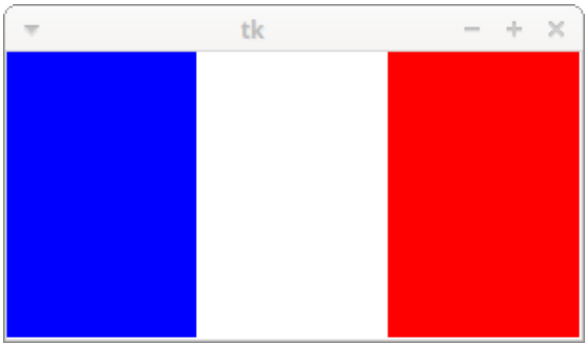
Nous allons utiliser le code suivant :

```
1  from PIL import Image
2  def pixel(x,y,r,g,b):
3    for i in range(0,s):
4      for j in range(0,s):
5        image.putpixel((x*s+i,y*s+j),(r,g,b))
6  w = 5;
7  h = 5;
8  s = 50;
9  image = Image.new("RGB",(w*s,h*s))
10 pixel(0,0,255,0,0)
11 pixel(3,0,0,255,0)
12 image.show()
```

478-code.py

1. Exécutez le code et modifiez les valeurs des variables `w`, `h`, `s`. Quelles effets ont ces variables sur l'affichage de l'image.

2. Modifiez le code pour obtenir l'image suivante qui a une dimension de 300×150 :



E.752

On souhaite obtenir le drapeau espagnol représenté ci-dessous et dont la dimension en pixel est 400×300

Pour cela, on complétera les pointillés du code ci-dessous

```
1  from PIL import Image
2  def pixel(x,y,r,g,b):
3    for i in range(s):
4      for j in range(s):
5        image.putpixel((x*s+i,y*s+j),(r,g,b))
6  w = ...;
7  h = ...;
8  s = ...;
9  image = Image.new("RGB",(w*s,h*s))
10 ...
11 image.show()
```

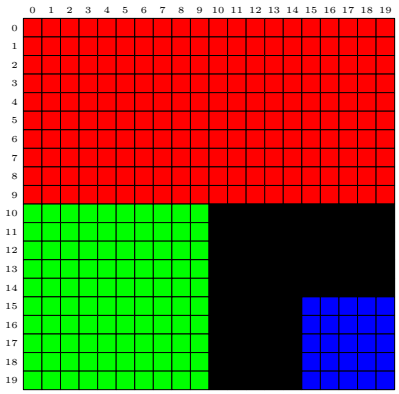
Indication : pour insérer un pixel noir en haut à gauche de l'image, on utilise l'instruction :

```
pixel(0,0,0,0,0)
```

B. Introduction au pixel art / évaluation

E.1380

On souhaite obtenir l'image ci-dessous :



Compléter le code ci-dessous afin d'obtenir cette image :

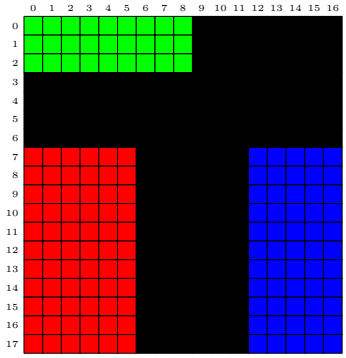
```
from PIL import Image

image = Image.new("RGB",(...,...));

for i in range(0,...):

 for j in range(0,...):

   image.putpixel((.....,.....),(...,...,...))

for i in range(0,10):

 for j in range(0,2):

   image.putpixel((.....,.....),(...,...,...))

for i in range(0,...):

 for j in range(0,...):

   image.putpixel((.....,.....),(...,...,...))

image.show()
```

Indication : on pourra partir du code ci-dessous pour obtenir le résultat escompté :

1380-code.py

E.1381

On souhaite obtenir l'image ci-dessous :



Compléter le code ci-dessous afin d'obtenir cette image :

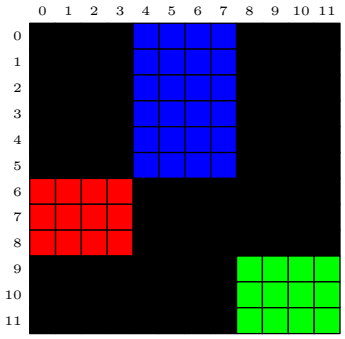
```
from PIL import Image

image = Image.new("RGB",(...,...));

for i in range(0,...):

 for j in range(0,...):

   image.putpixel((.....,.....),(...,...,...))

for i in range(0,10):

 for j in range(0,2):

   image.putpixel((.....,.....),(...,...,...))

for i in range(0,...):

 for j in range(0,...):

   image.putpixel((.....,.....),(...,...,...))

image.show()
```

Indication : on pourra partir du code ci-dessous pour obtenir le résultat escompté :

1381-code.py

E.1382

On souhaite obtenir l'image ci-dessous :



Compléter le code ci-dessous afin d'obtenir cette image :

```
from PIL import Image

image = Image.new("RGB",(...,...));

for i in range(0,...):

 for j in range(0,...):

   image.putpixel((.....,.....),(...,...,...))

for i in range(0,10):

 for j in range(0,2):

   image.putpixel((.....,.....),(...,...,...))

for i in range(0,...):
```

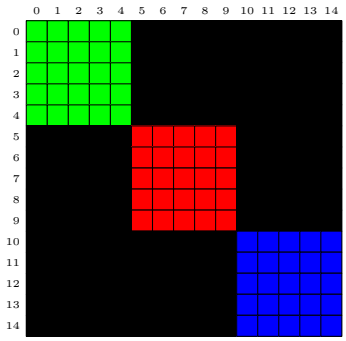
```
 for j in range(0,...):

   image.putpixel((.....,.....),(...,...,...))

image.show()
```

Indication : on pourra partir du code ci-dessous pour obtenir le résultat escompté :

1382-code.py

E.1383

On souhaite obtenir l'image ci-dessous :



Compléter le code ci-dessous afin d'obtenir cette image :

```
from PIL import Image

```

```
image = Image.new("RGB",(...,...));

for i in range(0,...):

 for j in range(0,...):

   image.putpixel((.....,.....),(...,...,...))

for i in range(0,10):

 for j in range(0,2):

   image.putpixel((.....,.....),(...,...,...))

for i in range(0,...):

 for j in range(0,...):

   image.putpixel((.....,.....),(...,...,...))

image.show()
```

Indication : on pourra partir du code ci-dessous pour obtenir le résultat escompté :

1383-code.py

C. *Pixel art*

E.506

Ci-contre est représenté un dessin en pixel-art où les couleurs ont été effacées et seuls sont restés les index de ces couleurs.

0	0	0	0	0	1	1	1	1	0	0	0	0	0
0	0	0	1	1	2	2	2	2	1	1	0	0	0
0	0	1	2	2	2	2	2	2	2	2	1	0	0
0	1	2	2	2	2	2	2	2	2	2	2	1	0
0	1	2	0	1	2	2	2	2	0	1	2	1	0
1	2	4	1	1	2	2	2	2	1	1	4	2	1
1	2	4	4	2	2	1	1	2	2	4	4	2	1
1	2	2	2	2	2	2	2	2	2	2	2	2	1
1	2	2	2	2	2	2	2	2	2	2	2	2	1
0	1	1	2	2	1	1	1	1	2	2	1	1	0
0	1	3	1	1	3	3	3	3	1	1	3	1	0
0	1	3	3	3	3	3	3	3	3	3	3	1	0
0	0	1	1	1	1	1	1	1	1	1	1	0	0
0	0	1	3	3	3	3	3	3	3	3	1	0	0
0	0	0	1	3	3	3	3	3	3	1	0	0	0
0	0	0	0	1	3	3	3	3	1	0	0	0	0
0	0	0	0	1	3	3	3	3	1	0	0	0	0
0	0	0	0	0	1	3	3	1	0	0	0	0	0
0	0	0	0	0	0	1	1	0	0	0	0	0	0

Figure 506-pixelart.pdf

Les couleurs originales sont données dans la palette de couleurs ci-dessous au travers de la variable `couleur`.

```
couleur=[(255,255,255),
         (0,0,0),
         (255,50,50),
         (255,150,150),
         (255,255,0)]
```

Palette de couleurs

1. Donner les dimensions de l'image.
2. Utilisez le lien suivant pour retrouver les couleurs de la palette et pour reconstruire la figure :

   506-palette.pdf

   506-palette-hex.pdf

E.503

On s'intéresse au code ci-dessous :

```
from PIL import Image
def pixel(x,y,c):
    for i in range(s):
        for j in range(s):
            image.putpixel((x*s+i,y*s+j),c)
listeCouleur = [(206, 177, 152), ...,(30, 28, 29)]
listePixel = [[9, 9, 9, 8, 8, ..., 4, 4, 4]]
w = len(listePixel[0])
h = len(listePixel)
s=10
image = Image.new("RGB",(w*s,h*s))
for i in range(w):
    for j in range(h):
```

```
        numCouleur = listePixel[j][i]
        couleur = listeCouleur[numCouleur]
        pixel(i,j,couleur)
image.show()
```

que vous pouvez récupérer à l'aide de l'URL :

503-code.py

1. Modifiez la variable `s` et exécutez le code. À quoi sert cette variable?

.....

.....

2. Utilisez les instructions `print(w)` et `print(h)`. À quoi correspondent ces deux variables dans le code?

.....

.....

3. a. Modifiez quelques valeurs de la liste `listeCouleur` *(attention les valeurs possibles sont entre 0 et 255)*. Qu'observez-vous?

.....

.....

   b. Modifiez quelques valeurs de la liste `listePixel` *(attention les valeurs possibles sont entre 0 et 15)*. Qu'observez vous?

.....

.....

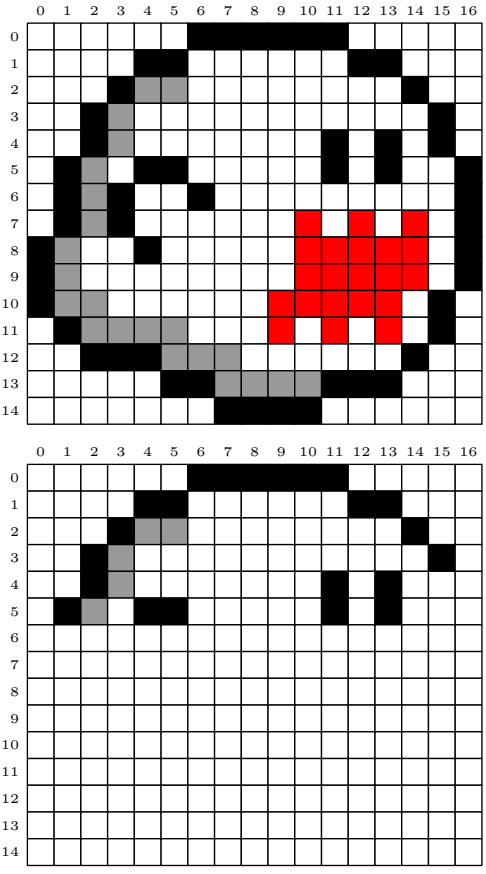
   c. Quel lien entre votre image et les listes `listeCouleur` et `listePixel`?

.....

.....

E.505

On souhaite réaliser le pixel-art présenté sur la figure 1 :



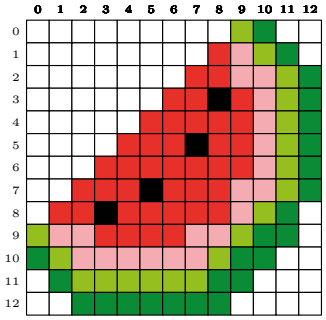
Le code ci-dessous permet de réaliser une partie du pixel-art souhaité. Téléchargez le code et complétez entièrement ce pixel-art.

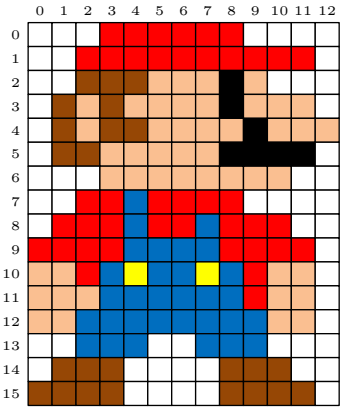
```
from PIL import Image
def pixel(x,y,c):
    for i in range(s):
        for j in range(s):
            image.putpixel((x*s+i,y*s+j),c)
listeCouleur = [(255,255,255),
           (0,0,0),
           (150,150,150)]
listePixel = [[0,0,0,0,0,0,1,1,1,1,1,1,0,0,0,0,0],
              [0,0,0,0,1,1,0,0,0,0,0,0,1,1,0,0,0],
              [0,0,0,1,2,2,0,0,0,0,0,0,0,1,0,0],
              [0,0,1,2,0,0,0,0,0,0,0,0,0,0,1,0],
              [0,0,1,2,0,0,0,0,0,0,0,1,0,1,0,0,0],
              [0,1,2,0,1,1,0,0,0,0,0,1,0,1,0,0,0]]
w = len(listePixel[0])
h = len(listePixel)
s=10
image = Image.new("RGB",(w*s,h*s))
for i in range(w):
    for j in range(h):
        numCouleur = listePixel[j][i]
        couleur = listeCouleur[numCouleur]
        pixel(i,j,couleur)
image.show()
```

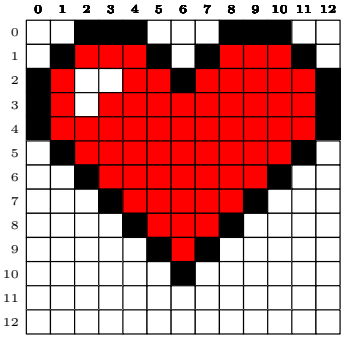
Le code est téléchargeable ici : 505-code.py

E.504

Dans cet exercice, on va reproduire un des pixels-arts ci-dessous avec Python :







1. Choisissez un des pixels-arts de votre choix et répondez aux questions suivantes :

   a. Quelles sont les dimensions de votre image?

.....

   b. Combien de couleurs contient votre image?

c Recherchez le code RGB de chacune des couleurs de votre image et notez leur code ci-dessous :

Couleur 1 : ……Couleur 2 : ……Couleur 3 : ……

Couleur 4 : ……Couleur 5 : ……Couleur 6 : ……

2 Pour coder cette image en Python, nous allons utiliser le code ci-dessous :

```
1  from PIL import Image
2  def pixel(x,y,c):
3      for i in range(s):
4          for j in range(s):
5              image.putpixel((x*s+i,y*s+j),c)
6  listeCouleur = [(255,0,0),
7              (0,255,0),
8              (0,0,255)]
9  listePixel = [[0, 1, 2], [1, 2, 0], [2,0,1]]
10 w = len(listePixel[0])
11 h = len(listePixel)
12 s=10
13 image = Image.new("RGB",(w*s,h*s))
14 for i in range(w):
15     for j in range(h):
16         numCouleur = listePixel[j][i]
17         couleur = listeCouleur[numCouleur]
18         pixel(i,j,couleur)
19 image.show()
```

récupérable via le lien : 504-code.py

Modifiez le code pour obtenir le pixel art de votre choix.

a Modifiez la palette de couleur à la ligne $\ell.4$ pour qu'elle corresponde à votre pixel-art.

b Redimensionnez correctement le tableau à la ligne $\ell.8$ correspondant aux pixels de votre image, puis la compléter correctement.

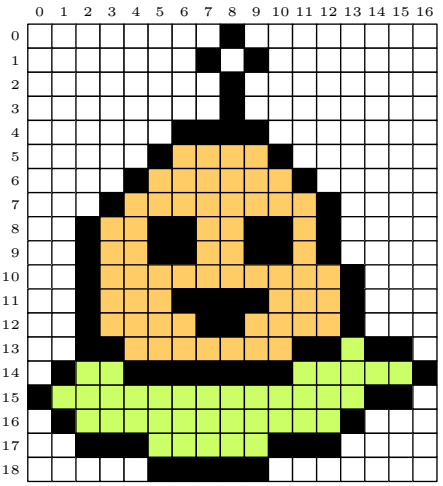
3 Pour enregistrer votre image sur votre disque, dur, vous pouvez rajouter l'instruction suivante à la fin de votre code :

```
from PIL import Image
canvas.postscript(file = "fichier-postscript.eps")
img = Image.open("fichier-postscript.eps")
img.save("fichier-image.png")
```

E.754

Le code ci-dessous permet de réaliser le pixel-art ci-contre

On souhaite que l'image réalisée ait pour dimension 80×95 pixels



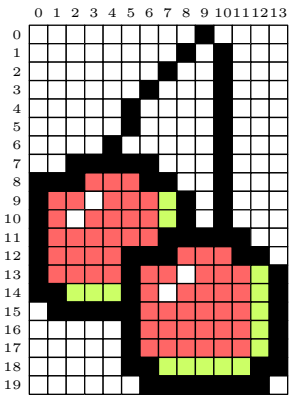
```
1  from PIL import Image
2  def pixel(x,y,c):
3      for i in range(s):
4          for j in range(s):
5              image.putpixel((x*s+i,y*s+j),c)
6  listeCouleur = [(255,255,255),
7                  (0,0,0), ...]
8  listePixel = [[0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0],
9                [0,0,0,0,0,0,0,1,0,1,0,0,0,0,0,0,0],
10               [0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0],
11               [0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0],
12               [0,0,0,0,0,0,1,1,1,1,0,0,0,0,0,0,0],
13               ...,
14               ...,
15               [0,0,0,1,3,3,3,3,3,3,3,3,1,0,0,0,0],
16               [0,0,1,3,3,1,1,3,3,1,1,3,1,0,0,0,0],
17               [0,0,1,3,3,1,1,3,3,1,1,3,1,0,0,0,0],
18               [0,0,1,3,3,3,3,3,3,3,3,3,3,1,0,0,0],
19               [0,0,1,3,3,3,1,1,1,1,3,3,3,1,0,0,0],
20               [0,0,1,3,3,3,3,1,1,3,3,3,3,1,0,0,0],
21               [0,0,1,1,3,3,3,3,3,3,3,3,1,1,2,1,1,0],
22               [0,1,2,2,1,1,1,1,1,1,1,2,2,2,2,2,1],
23               [1,2,2,2,2,2,2,2,2,2,2,2,2,2,2,1,1,0],
24               [0,1,2,2,2,2,2,2,2,2,2,2,2,2,1,0,0,0],
25               [0,0,1,1,1,2,2,2,2,2,1,1,1,0,0,0,0],
26               [0,0,0,0,0,1,1,1,1,1,0,0,0,0,0,0,0]]
27 w = len(listePixel[0])
28 h = len(listePixel)
29 s=20
30 image = Image.new("RGB",(w*s,h*s))
31 for i in range(w):
32     for j in range(h):
33         numCouleur = listePixel[j][i]
34         couleur = listeCouleur[numCouleur]
35         pixel(i,j,couleur)
36 image.show()
```

754-code.py

E.755

Le code ci-dessous permet de réaliser le pixel-art ci-contre

On souhaite que l'image réalisée ait pour dimension 80×95 pixels



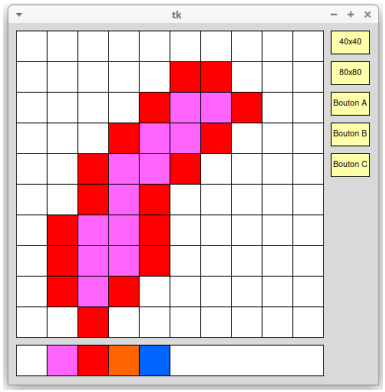
```
from PIL import Image
from tkinter import *
listeCouleur = [(255,255,255),
                (0,0,0),
                ..... ]
listePixel = [[0,0,0,0,0,0,0,0,0,1,0,0,0,0],
              [0,0,0,0,0,0,0,0,1,0,1,0,0,0],
              [0,0,0,0,0,0,0,1,0,0,1,0,0,0],
              [0,0,0,0,0,0,1,0,0,0,1,0,0,0],
              [0,0,0,0,0,1,0,0,0,0,1,0,0,0],
              [0,0,0,0,0,1,0,0,0,0,1,0,0,0],
              ...
              ...
              [1,1,1,3,3,3,1,1,0,0,1,0,0,0],
              [1,3,3,0,3,3,3,2,1,0,1,0,0,0],
              [1,3,0,3,3,3,3,2,1,0,1,0,0,0],
              [1,3,3,3,3,3,3,1,1,1,1,1,0,0],
              [1,3,3,3,3,1,1,1,3,3,3,1,1,0],
              [1,3,3,3,3,1,3,3,0,3,3,3,2,1],
              [1,1,2,2,2,1,3,0,3,3,3,3,2,1],
              [0,1,1,1,1,1,3,3,3,3,3,3,2,1],
              [0,0,0,0,0,1,3,3,3,3,3,3,2,1],
              [0,0,0,0,0,1,3,3,3,3,3,3,2,1],
              [0,0,0,0,0,1,1,2,2,2,2,2,1,1],
              [0,0,0,0,0,0,1,1,1,1,1,1,1,0]]
a=...
b=...
c=...
fen = Tk()
canvas = Canvas(fen,width=w*s,height=h*s)
canvas.pack()
for i in range(w):
 for j in range(h):
  numCouleur = listePixel[j][i]
  couleur = listeCouleur[numCouleur]
  col = f"#{couleur[0]:02x}
              {couleur[1]:02x}{couleur[2]:02x}"
  canvas.create_rectangle(i*s,j*s,(i+1)*s,
                          (j+1)*s,fill=col)
fen.mainloop()
```

755-code.pyy

D. *Mini-projet: logiciel pixel art (version module)*

E.1349

Dans cet exercice, nous allons créer un logiciel d'édition des pixels arts :



Le canvas de dessin a pour dimension 400x400 pixels. Vous pouvez voir certaines de ces fonctionnalités via la vidéo suivante :

1349-video.mp4

Pour cela, on utilisera les deux fichiers :

- 1349-code.py
- 1349-module.py

Ces deux fichiers doivent se trouver dans le même dossier et on doit leur retirer leur préfixe *(1349-)*.

Voici les fonctions que vous devez compléter :

- `dessineGrille()` : dessine la grille. Cette fonction doit adapter la taille de la grille en fonction de la valeur retournée par `getTailleCase()`
- `clickGrille(x,y)` : cette fonction est activée lorsque le client clique sur la grille : ses paramètres sont alors les coordonnées, en pixels, du clic
- `dessinePixelArt()` : redessine le pixel art.
- `initPalette()` : dessine les cases de couleurs dans la zone de la palette de couleurs.
- `clickPalette(num)` : cette fonction est activée lorsque le client clique sur la zone de la palette de couleurs. Il retourne alors la position du clic.
- `bouton40()` : définit des cases de dimensions 40x40
- `bouton80()` : définit des cases de dimensions 80x80
- `boutonA()`, `boutonB()`, `boutonC()`, ces fonctions pourront recevoir les fonctionnalités de votre choix.

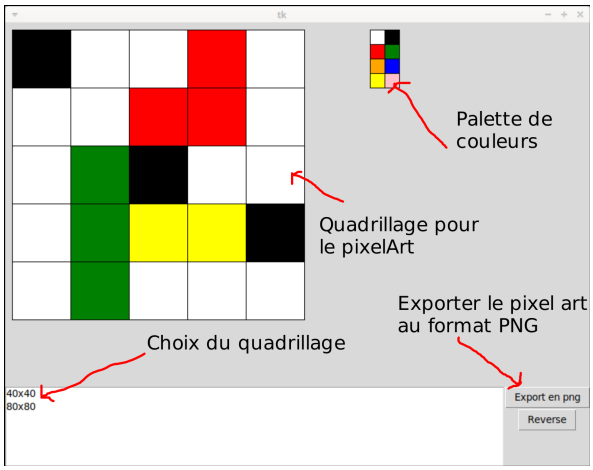
Voici les fonctions vous permettant d'interagir avec l'interface :

- `setTailleCase(nbr)` : stocke la valeur `nbr` dans le module qui sera utilisé pour la taille de chaque pixel.
- `getTailleCase()` : renvoie la taille d'un pixel.
- `ligne(x,y,xx,yy)` : trace la ligne qui relie les deux points $A(x,y)$ et $B(xx,yy)$.
- `rectangle(x,y,xx,yy,c)` : trace le rectangle dont les points $A(x,y)$ et $B(xx,yy)$ sont deux sommets opposés. La couleur `c`, triplet au format RGB, sera utilisée pour remplir le rectangle.
- `casePalette(num,c)` : trace un rectangle dans la zone de la palette de couleurs à la position `num` avec la couleur `c`, triplet au format RGB.

E. *Mini-projet: logiciel pixel art (version complete)*

E.826

Le projet est de créer une interface permettant de créer facilement des interfaces.
Voici une capture d'écran de cette interface :



Elle doit comprendre quatre parties :

- un quadrillage représentant l'espace de travail de notre pixelArt.
  Lorsque l'utilisateur cliquera sur les cases du quadrillage, celles-ci prendront la couleur par défaut.
  La dimension du quadrillage est de 400×400 pixels
- il est possible de modifier la taille des cases du quadrillage :
  - avec une taille de cases de 40×40, notre espace de travail sera composé de 10×10 cases.
  - avec une taille de cases de 80×80, notre espace de travail sera composé de 5×5 cases.
- une palettre de couleurs va permettre à l'utilisateur de choisir la couleur par défaut.
- Le bouton "*exporter en png*" permet à l'utilisateur de récupérer son image au format `png`.

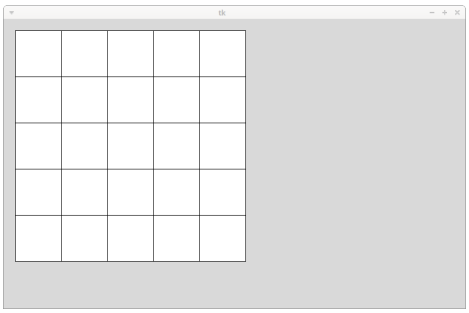
Voici une vidéo présentant cette interface :
826-video.mp4

Pour réaliser cette interface, nous utilisons le package `Tkinter`.
Voici quelques usages de `Tkinter`

- Comment récupérer un clic dans Tkinter?
  https://chinginfo.fr/829
- Comment exporter un canvas vers une image au format png?
  https://chinginfo.fr/830
- Comment créer un bouton dans Tkinter?
  https://chinginfo.fr/831

E.844

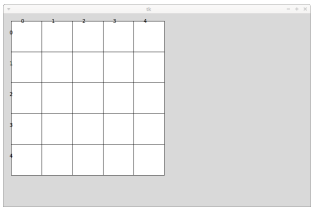
Téléchargez le code ci-dessous :
844-code.py
et modifiez-le pour faire apparaitre les 25 cases comme indiqué ci-dessous :



où chaque case a pour dimension 80×80 pixels et la marge gauche et haut a pour dimensions 40 pixels.

E.845

1 Nous créons notre propre repère pour localiser chacune des cases de notre interface :



Nous souhaitons que lorsqu'on clique une case, appa-

raisse dans la console les coordonnées de la case avec notre nouveau repère comme indiqué dans la vidéo ci-dessous :

845-video.mp4

Pour commencer, nous pouvons utiliser le code ci-dessous :

845-code.py

2 Utilisez le code précédent pour que le clic sur une case la rende noire *(en fait, qu'une case noire soit dessinée)* comme le montre la vidéo ci-dessous :

845-video2.mp4

E.867

Utilisez le code ci-dessous pour reconstruire votre logiciel de pixelart :

867-code.py