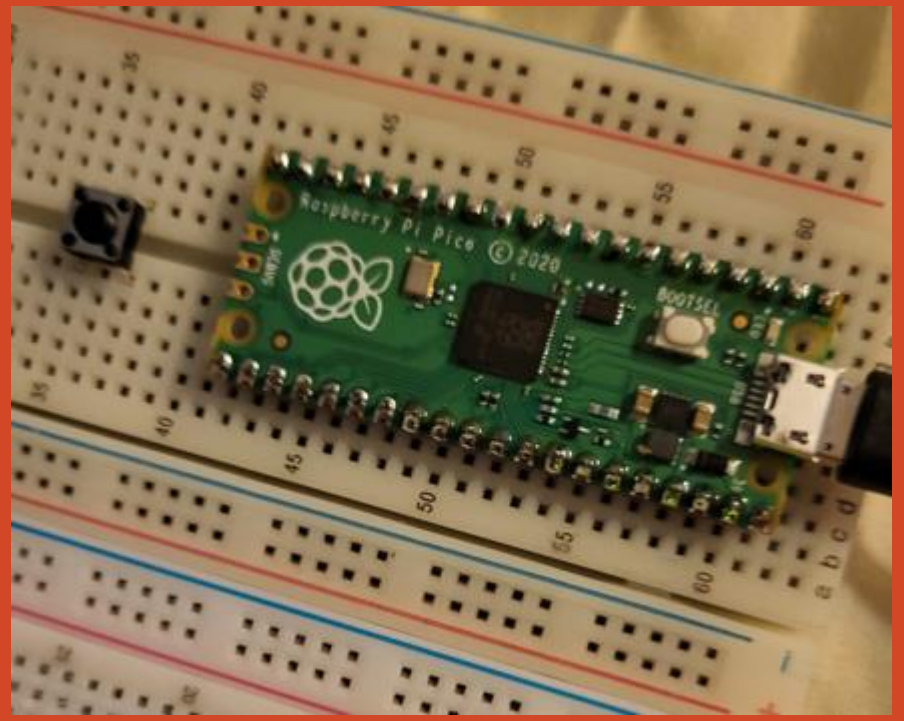
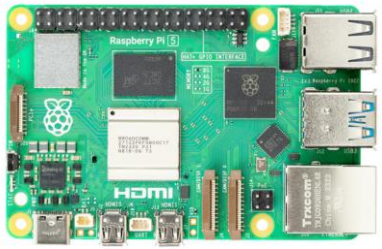
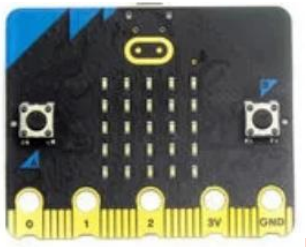
DOJOCON 2025

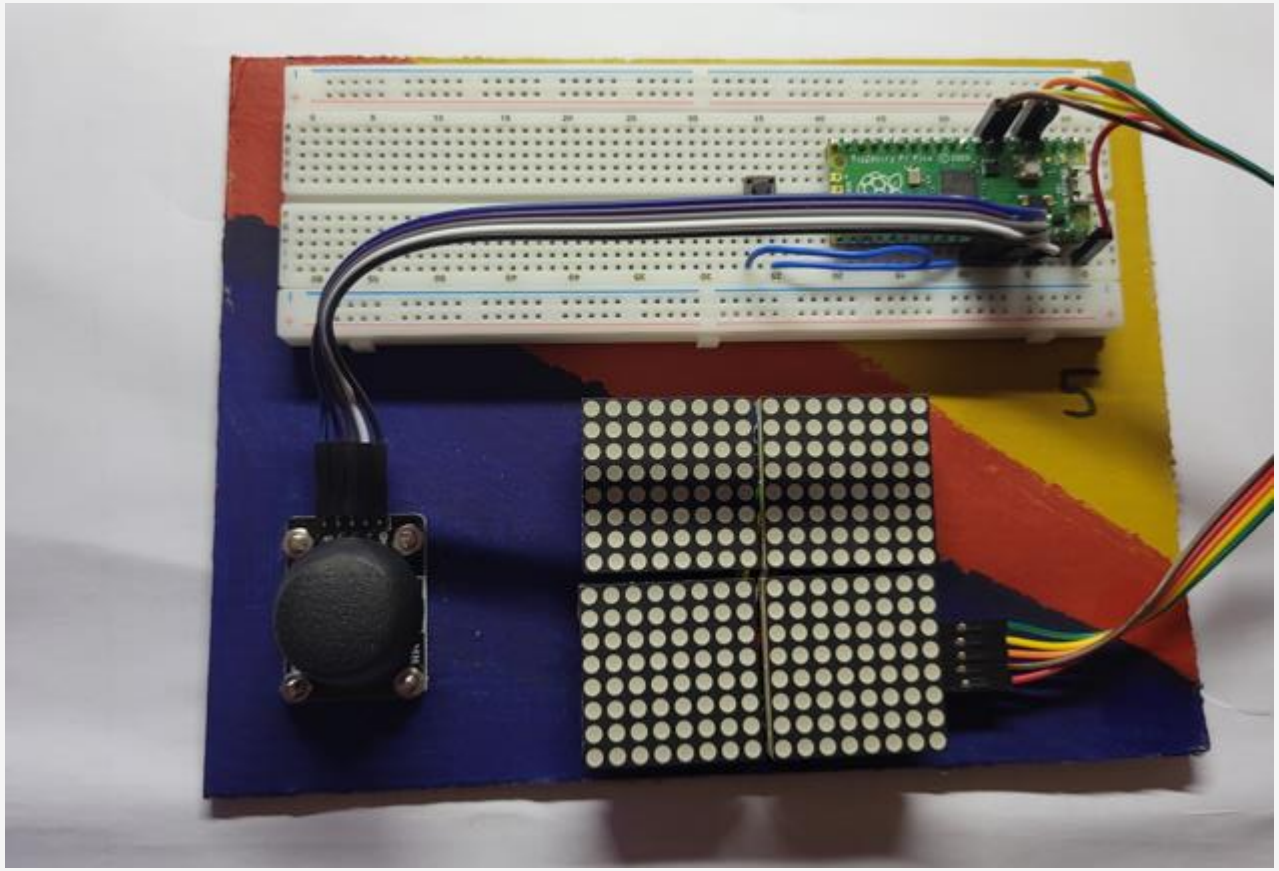
Using the Raspberry Pi Pico



The Raspberry Pi Pico

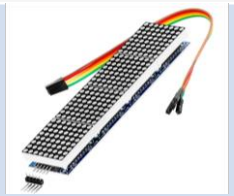
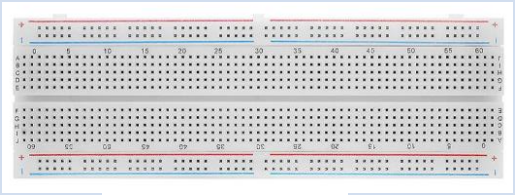
	Raspberry pi 5	Raspberry pi pico	Microbit
			
	Micro ordinateur	**Micro Contrôleur**	Micro Contrôleur
CPU	Quad core Arm Cortex A76 2.4GHz	**RP20240 Dual core ARM cortex M0+ 133MHz**	nRF52833 ARM Cortex M4 64MHz
Mémoire	2 à 16 GBytes	**264kBytes**	128kBytes
Operating System	Linux	**Pas d'operating system**	Pas d'operating system
Affichage	GPU avec 2 sorties HDMI	**Pas d'affichage**	5x5 LED matrix
Stockage	Carte SD	**2 MBytes Flash**	0.5 Mbytes Flash
I/O	26 I/O sur pin header	**26 I/O sur pin header**	3 IO + 16 sur PCB

The 16x16 pico console



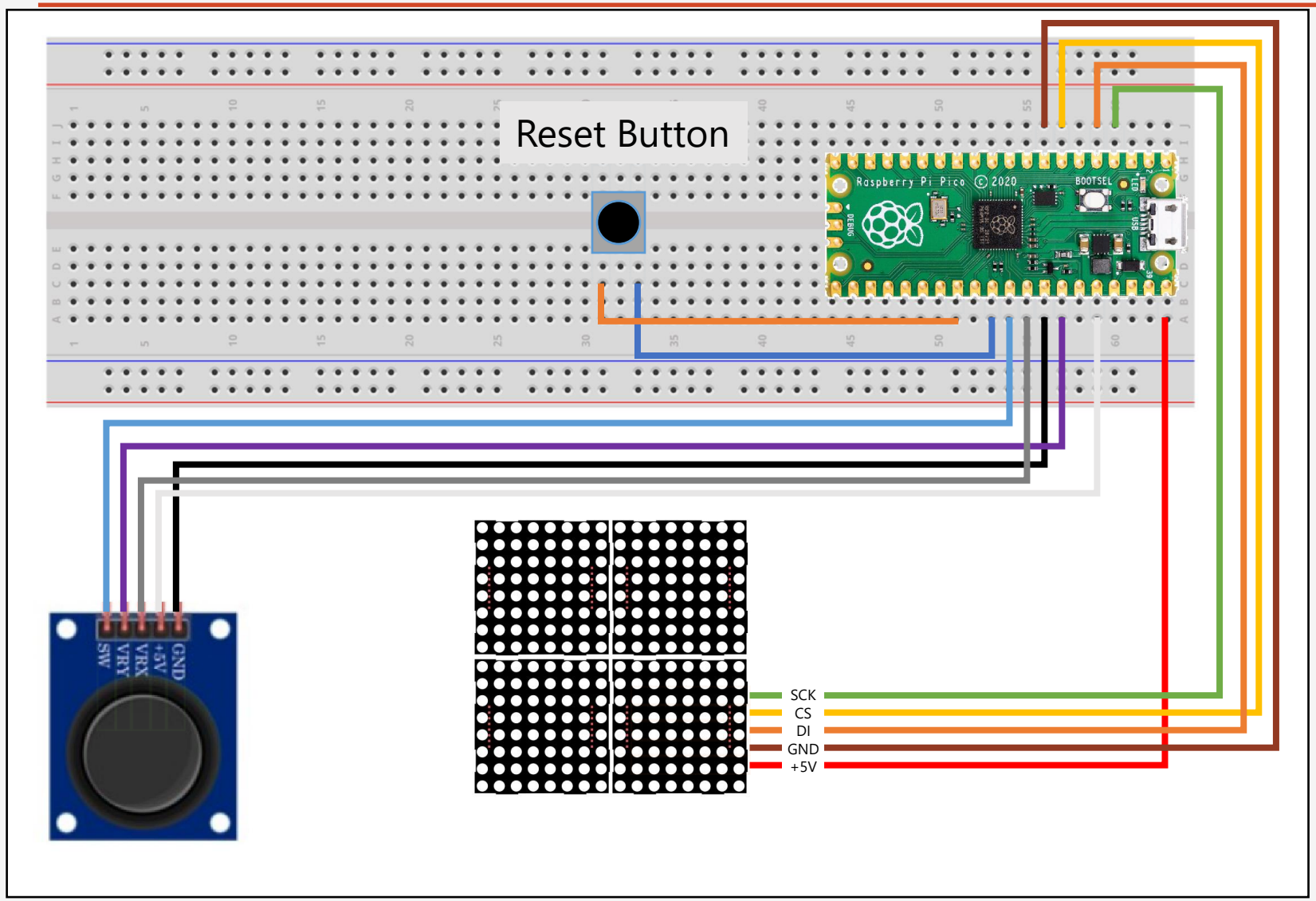
- ❑Bouton Reset
- ❑Joystick Connecté sur ADC 27 et ADC 28
- ❑Affichage LED 16x16 connecté par bus SPI

Matériel

	LED MATRIX	11.99€
	Joystick	5.49€
	BreadBoard	5.49€
	Raspberry pi pico	11.99€
	Reset Button	0.35€
	Flat cable	4.99€

Total: 40.30€

Schéma de la PiConsole16x16

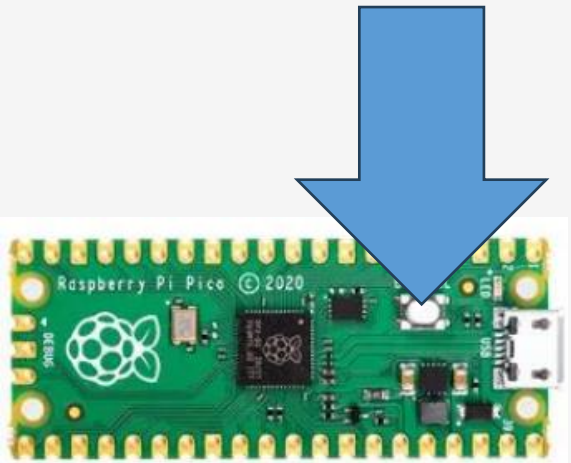


MicroPython

Micropython = Python 3 pour micro contrôleur

Installation:

1. Download *RPI_PICO-20250911-v1.26.1.uf2* sur https://micropython.org/download/RPI_PICO/

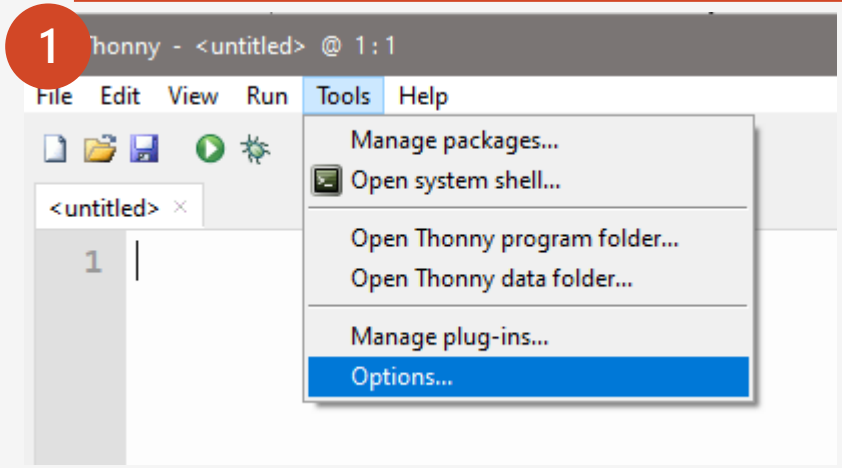


2. Brancher le Raspberry Pi Pico en maintenant le bouton BOOTSEL enfoncé

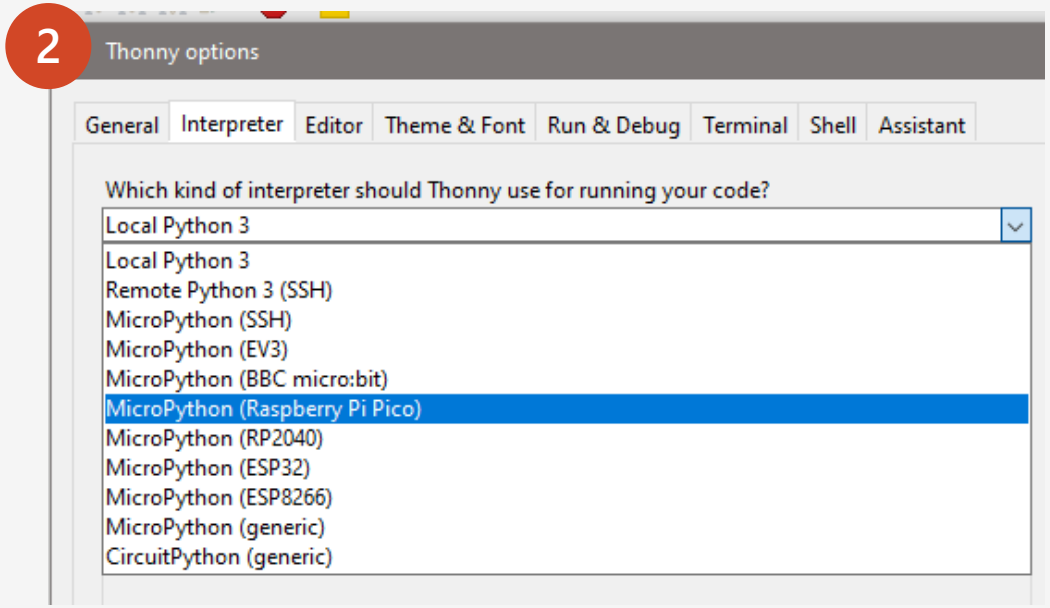
3. Copier le fichier *RPI_PICO-20250911-v1.26.1.uf2* sur le Disque du Raspberry pi Pico

Configurer Thonny

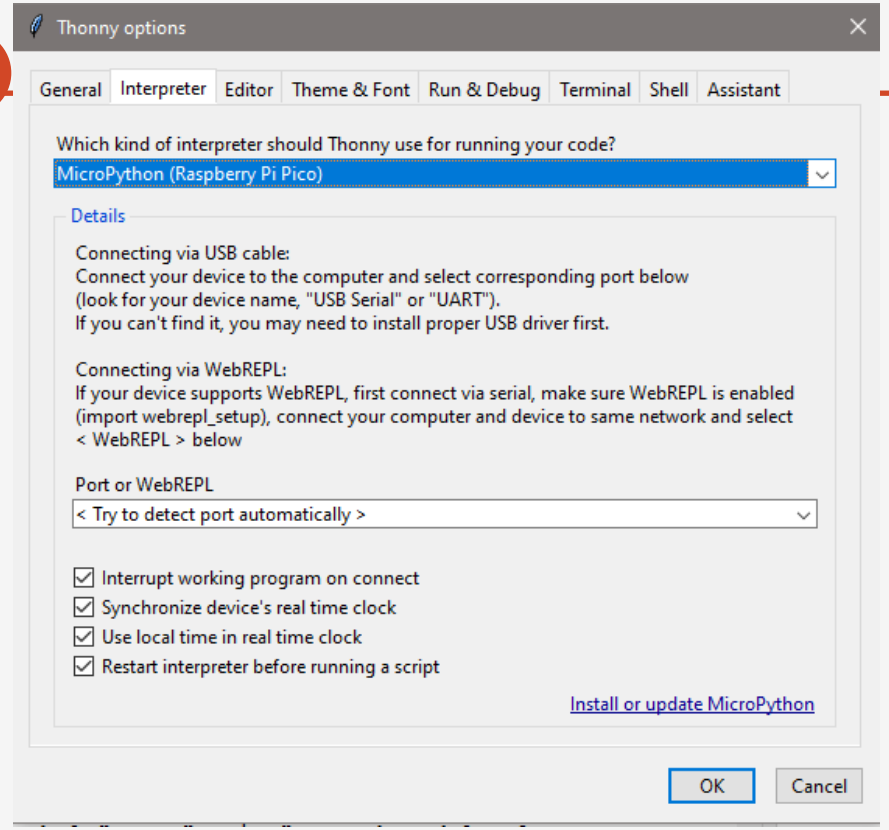
1



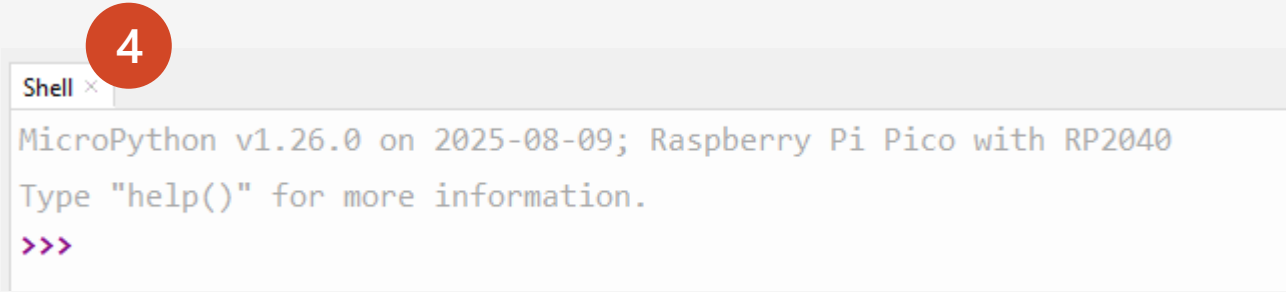
2



3

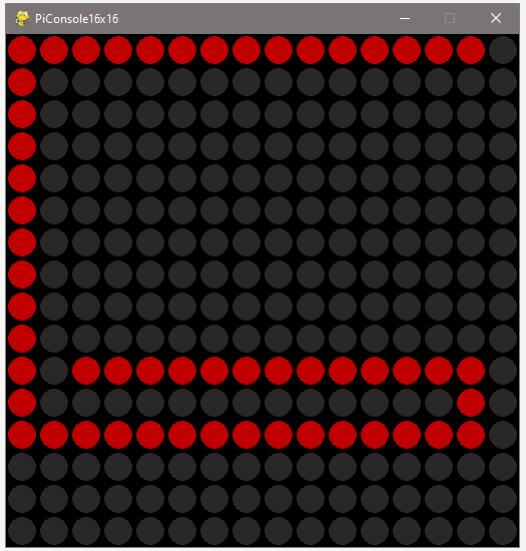


4

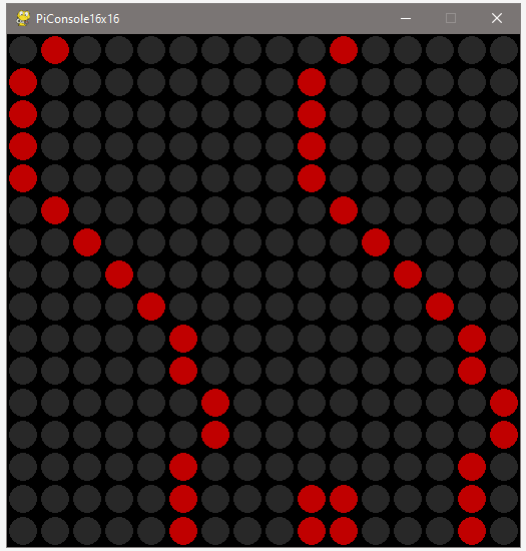


Des jeux sur un afficheur de seulement 16x16

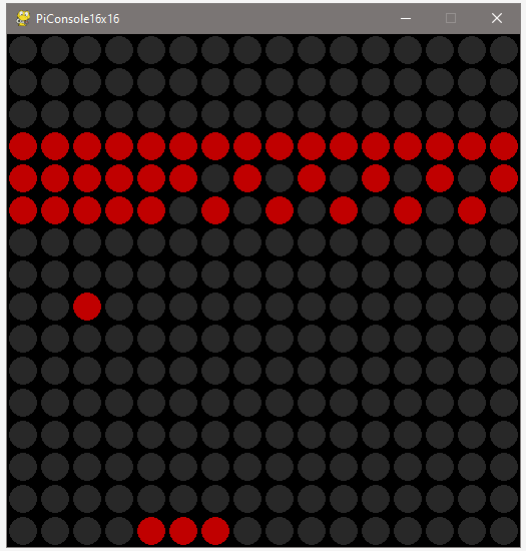
1. Snake



2. Race



3. Casse Brique



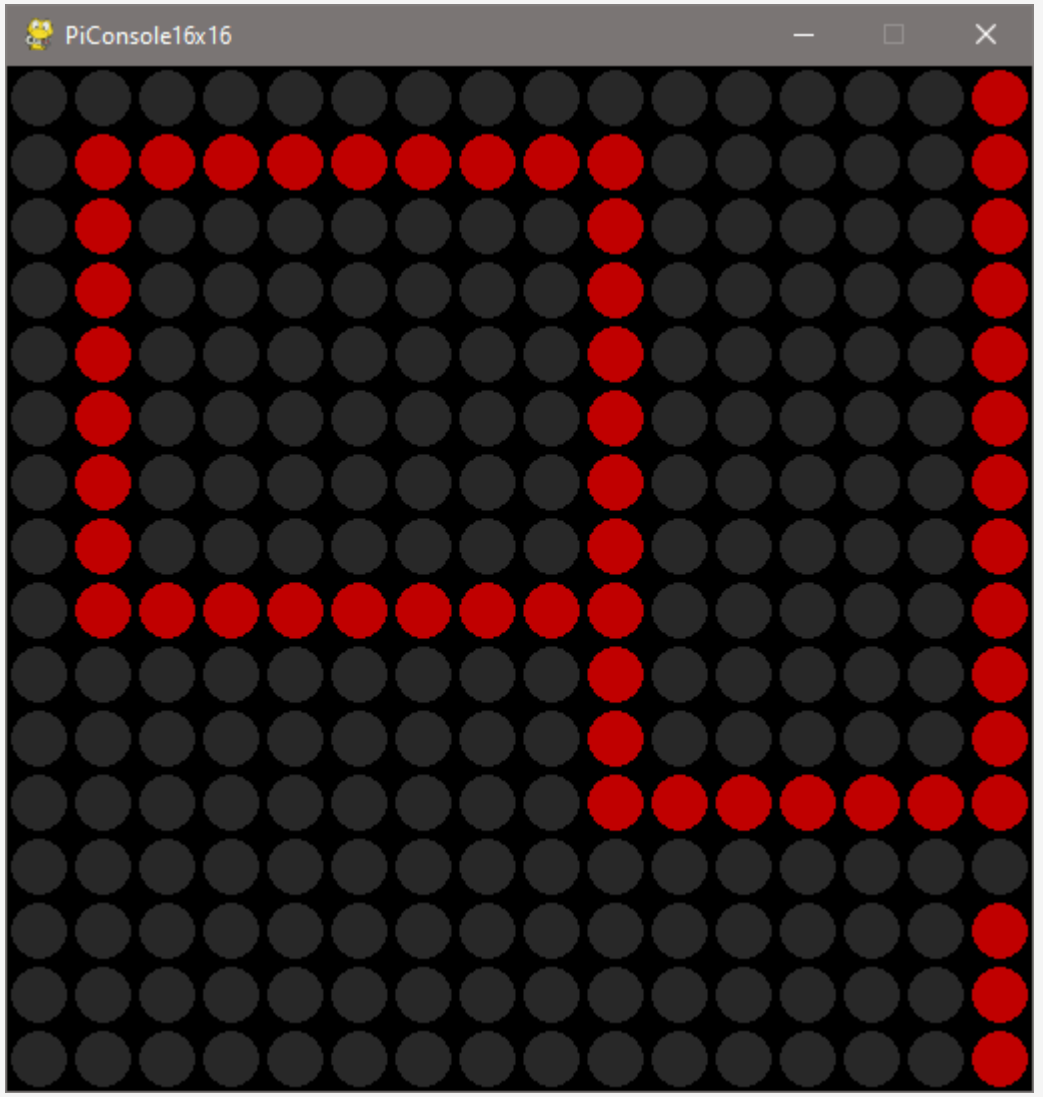
4. Tetris

5. Flappy bird

... ...

Simulateur de PiConsole16x16 (en pygame)

Si le hardware n'est pas disponible, un simulateur en pygame est disponible



PiConsole16x16 API

```
from PiConsole16x16 import *

console = PiConsole16x16()
```

```
console.setPixel(x,y,1)
console.setPixel(x,y,0)
```

1: allume la LED
0: éteint la LED

```
console.clear()
```

Éteint toutes les LEDs

```
p = console.getPixel(x,y)
```

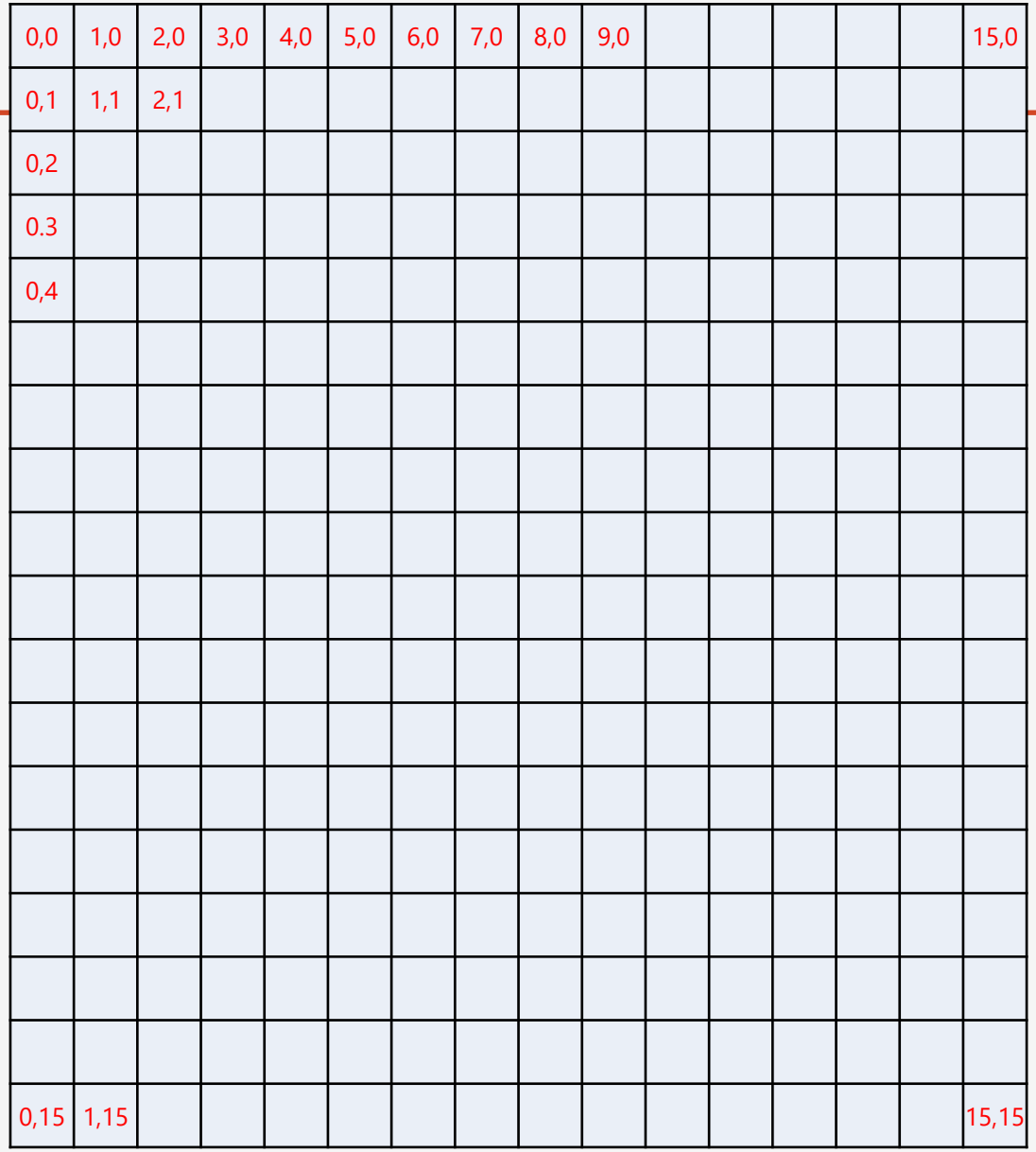
Lit la valeur d'un pixel

```
console.refresh()
```

Met à jour l'affichage

```
x = console.joyX()
y = console.joyY()
```

Valeur comprise entre -7 et +7



Structure d'un jeu (exemple: snake)

- Initialisation
- Boucle
  - Déplacements
  - Dessin
  - Affichage
  - Pause

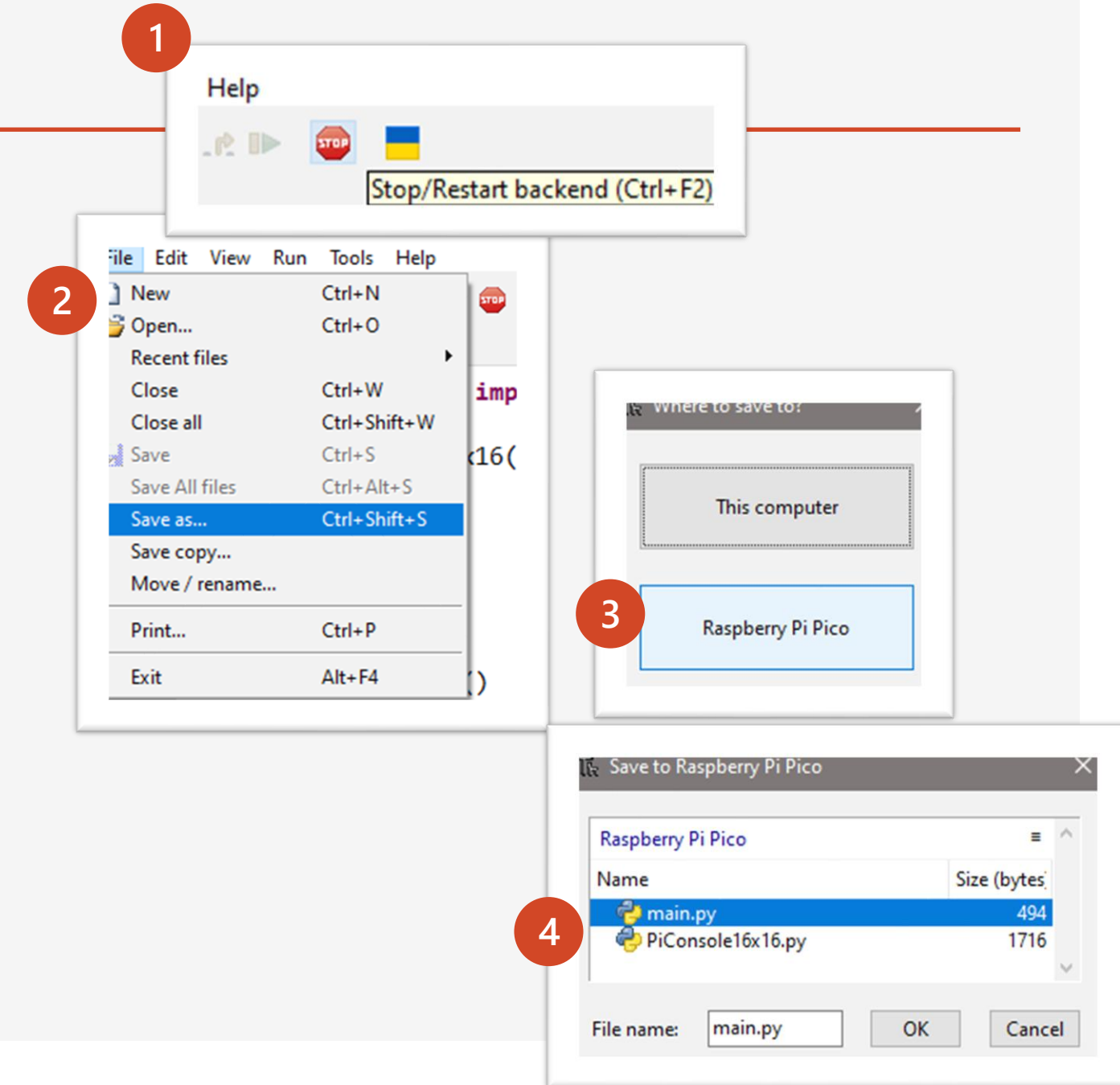
```
from PiConsole16x16 import *
import time
console=PiConsole16x16()
perdu=False
px=8
py=8
vx=0
vy=0
while not perdu:
    jx=console.joyX()
    jy=console.joyY()
    if abs(jx)>abs(jy):
        vy=0
        vx=1 if jx>0 else -1
    elif abs(jy)>abs(jx):
        vx=0
        vy=1 if jy>0 else -1
    px=(px+vx)%16
    py=(py+vy)%16
    if (vx+vy)!=0 and console.getPixel(px,py)==1:
        perdu=True
    console.setPixel(px,py,1)
    console.refresh()
    time.sleep(0.1)
```

Sauver le Jeu sur le Raspberry PI

Note: si le programme s'appelle

main.py ,il sera lancé

automatiquement après chaque reset



Améliorons le Snake

Limitons la longueur du Snake à 25 Pixels

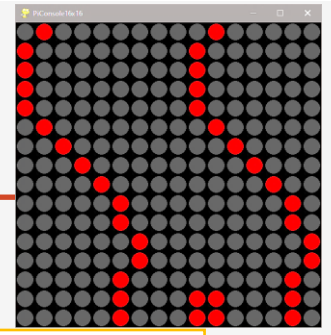
1/ Déclarons 2 nouvelles variables

```
snake=[[px,py]]
longueur=55
while not perdu:
```

2/ Ajoutons au dessin l'effacement de la queue du serpent

```
    console.setPixel(px,py,1)
    if (vx+vy)!=0:
        snake.append([px,py])
    if len(snake)>longueur:
        console.setPixel(snake[0][0],snake[0][1],0)
        snake.pop(0)
    console.refresh()
```

Jeu de course



Initialisation

```
1   from PiConsole16x16 import *
2   import time
3   import random
4   console=PiConsole16x16()
5   perdu=False
6   px=8
7   left=[3]*16
8   right=[13]*16
9   w=10
10  distance=0
```

Position du joueur

Bords gauche et droite de la route

Largeur de la route

Distance parcourue (sert a diminuer progressivement la largeur de la route)

Boucle de jeu

```
11  while not perdu:
12      jx=console.joyX()
13      if jx>1 and px <14:
14          px+=1
15      elif jx<-1 and px>1:
16          px-=1
17      l=left[15]+random.randint(-1,1)
18      if l<0:
19          l=0
20      if l+w>15:
21          l-=1
22      left.append(l)
23      left.pop(0)
24      right.append(l+w)
25      right.pop(0)
26      distance+=1
27      if distance%50==0:
28          w-=1
29      if px+1>=right[0] or px<=left[0]:
30          perdu=True
31      console.clear()
32      for i in range(16):
33          console.setPixel(left[i],15-i,1)
34          console.setPixel(right[i],15-i,1)
35      for i in range (14,16):
36          console.setPixel(px,i,1)
37          console.setPixel(px+1,i,1)
38      console.refresh()
39      time.sleep(0.1)
```

Déplace le joueur

Calcule la route

Détecte les collisions

Dessine le jeu