



Bibliothèque et boucle

```
from microbit import *
```

→ importe la bibliothèque Micro:bit

```
while True:
```

→ boucle infinie (qui ne se termine jamais)

Gestion des DEL de la matrice d'affichage

```
display.set_pixel(x,y,v)
```

→ allume la Del en position (x,y) à une intensité v (entre 0 et 9)

```
display.clear()
```

→ éteint toutes les DEL

```
display.scroll("message")
```

→ fait défiler un message sur les Del

```
display.show(Image)
```

→ Affiche une image sur les Del

Gestion des boutons programmables A et B

```
button_a.is_pressed()
```

→ retourne vrai si le bouton « A » est pressé au moment de l'appel

```
button_a.was_pressed()
```

→ retourne vrai si le bouton « A » a été pressé depuis le dernier appel

Gestion des capteurs

```
temperature()
```

→ récupère la température de la carte

```
compass.calibrate()
```

→ fonction de calibration du compas

```
compass.heading()
```

→ retourne le cap dans lequel est orientée la carte de 0 (Nord) à 360

```
accelerometer.get_x()
```

→ donne l'accélération selon l'axe x (fonctionne aussi pour y et z)

```
accelerometer.get_values()
```

→ retourne les valeurs d'accélération dans un tuple de trois valeur (x, y, z)

```
display.get_pixel(x, y)
```

→ retourne l'intensité lumineuse captée sur la led (x, y)

Gestion de la fonction radio

```
import radio
```

→ importe les fonctions pour la radio

```
radio.on()
```

→ allume la radio

```
radio.off()
```

→ éteint la radio

```
radio.send("A")
```

→ envoie le message texte « A »

```
radio.receive()
```

→ réception d'un message



Gestion des connecteurs externes (broches)

<code>pin0.read_digital()</code>	→	renvoie la valeur 0 ou 1 de la broche 0
<code>pin0.write_digital(n)</code>	→	écrit la valeur logique « n » (0 ou 1) sur la broche 0
<code>pin0.read_analog()</code>	→	mesure la tension sur la broche et retourne un entier entre 0 et 1023 proportionnel à la tension mesurée
<code>pin0.write_analog(y)</code>	→	génère un signal proportionnel à la valeur y (compris entre 0 et 1023)
<code>pin0.set_analog_period(x)</code>	→	définit la période du signal généré sur le pin analogique 0
<code>pin0.set_pull(pin0.PULL_UP)</code>	→	Configure le pin 0 en Pull Up
<code>pin0.set_pull(pin0.PULL_DOWN)</code>	→	Configure le pin 0 en Pull Down

Neopixel

<code>import neopixel</code>	→	importe la bibliothèque neopixel
<code>np = neopixel.NeoPixel(pin , n)</code>	→	initialise une nouvelle bande de nombre de n LEDs neopixels contrôlés via une broche pin.
<code>np.clear()</code>	→	efface tous les pixels.
<code>np.show()</code>	→	affiche les pixels. Doit être appelé pour que les mises à jour deviennent visibles.
<code>np[n] = (R, G, B)</code>	→	met le pixel n à une couleur RGB

Fonctions de contrôle

<code>sleep(n)</code>	→	suspend le programme pendant n millisecondes (ms)
<code>running_time()</code>	→	retourne le temps en ms depuis le démarrage de la carte