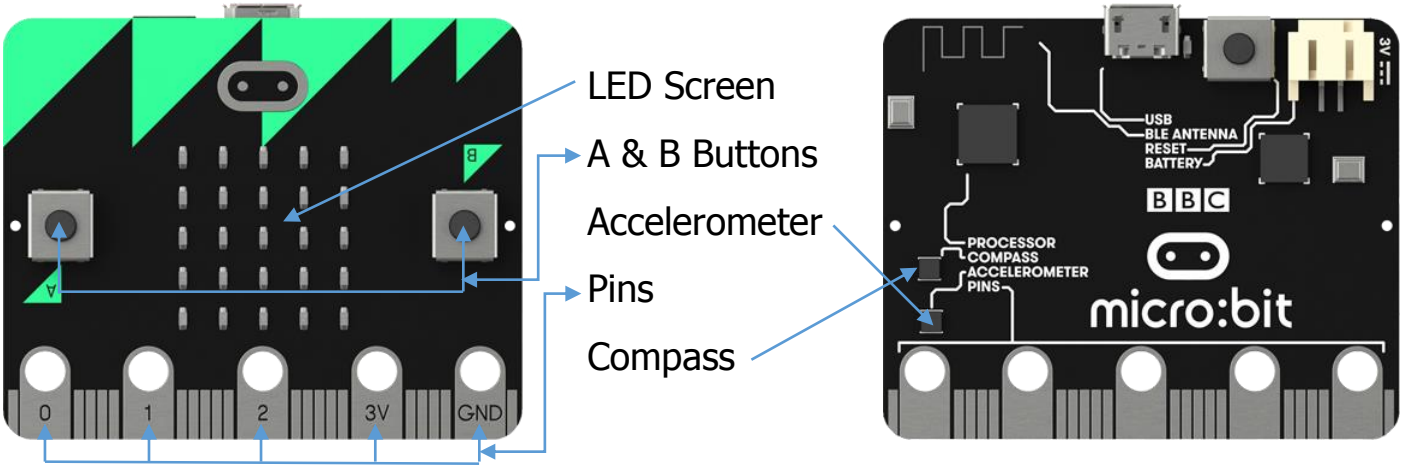



- ☐ Go to microbit.org/code
- ☐ Under Python Editor, click **Let's Code**

✓ You are now ready to start!

## *Section 1 – LED Screen*

✓ There is some code written for you in the editor already:

```
from microbit import *
```

This line is **very important**, as it allows you to use all of the features of the micro:bit shown in the diagram above, such as the display, the buttons and accelerometer. **Never delete this line.**

```
while True:
```

This line is a **loop**, it tells all the code **indented** underneath to happen over and over again.

```
    display.scroll('Hello, World!')
    display.show(Image.HEART)
    sleep(2000)
```

These lines are all **indented**, which means moved across 4 spaces, or one tab, to the right.

This means that they all 'belong' to the while loop above, and so will repeat over and over again.

✓ It is important that you type code **exactly** in Python. Check your Capital Letters & colons **:**

✓ `display.scroll('Hello, World!')` – this line of code means that the words in the speech marks will scroll across the micro:bit's display.

□ Change the code so that it will show a different word. Try your name!

✓ `display.show(Image.HEART)` – this line of code means that the micro:bit's display will show an image of a heart. There are many built in images that you can use.

□ Change the code so that instead of `HEART`, the micro:bit shows `PACMAN`.

✓ `sleep(2000)` – this line of code means that the micro:bit will **wait** for 2000 milliseconds before it moves on to the next line. 2000 milliseconds is 2 seconds.

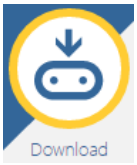
□ Change the code so that it waits for a different amount of time.

✓ You are now going to download your code onto the microbit!

**It is very important that, once your micro:bit is connected to the computer, you hold your micro:bit by the outside edges, so that you don't accidentally damage the circuitry. Leave it flat on the table when you are not using it.**

□ Plug the micro USB end of your cable into your micro:bit.

□ Plug the USB end of your cable into the computer.

□ Click 


□ Save the .hex file onto the micro:bit. If it automatically downloads, you will need to **drag and drop** the .hex file onto the micro:bit, which will show up like a USB stick.

? Ask for help here if you need it.

✓ A light on the back of the micro:bit will flash yellow while your file is being downloaded. When the light stops, your program is ready to try!

□ Does it do what you expect?

? If you find any errors, go back to your code and see if you can find the problem.

✓ You are now going to write new code to make a heart picture to flash on and off, to make a heartbeat.

- Delete the line `display.scroll()`, and change the `PACMAN` image back to `HEART`.
- After your `sleep()` line, make a new line, remember the **indent**.
- Type this code on your new line: `display.clear()`
- Underneath this, on a new line, add another `sleep()`, making the number in the brackets the same as in your previous `sleep()`.

✓ Your code should now look something like this:

```
from microbit import *

while True:
    display.show(Image.HEART)
    sleep(1000)
    display.clear()
    sleep(1000)
```

- Download your code to your micro:bit.

? The micro:bit's display should now be flashing a heart picture on and off.
If not, now is a good time to ask for help.

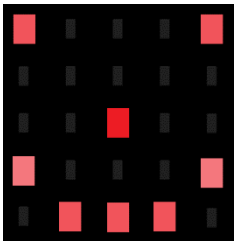
✓ You can also make your own images, by choosing which of the display's pixels to turn on. `display.show(Image('00000:00000:00000:00000:00000'))` has all of the pixels off.

`display.show(Image('99999:99999:99999:99999:99999'))` has all of the pixels on at maximum brightness. You can use numbers between `0` and `9` to choose how bright each one is.

- Write new code to show your own pictures, and see what they look like on the micro:bit.

e.g.

```
display.show(Image('07070:00000:00900:50005:07770'))
```



# *Section 2 – A & B Buttons*

✓ You are now going to tell the micro:bit to display an picture ONLY when you press A.

☐ Delete the last lines of your code, so that all you have left is:

```
from microbit import *

while True:
    display.show(Image.HEART)
    sleep(1000)
```

☐ Make a new line between `while True` and `display.show()`.

☐ Type this code on your new line: `if button_a.was_pressed():`

☐ Now add an extra indent to the code underneath, so that `display.show()` 'belongs' to the `if` statement, which 'belongs' to the `while` loop.

✓ Your code should now look like this:

```
while True:
    if button_a.was_pressed():
        display.show(Image.HEART)
        sleep(1000)
```

✓ This is asking the question – is button A pressed? If the answer is yes, a heart will display.

☐ Download your code to your micro:bit to make sure it works as you expect.

✓ You now want to ask the question – **is button B pressed**? Then tell it what to do if the answer is yes, and what to do if the answer to **both questions** is no.

✓ The code that you will need to write for this, takes the following format:

```
if button_a.was_pressed: <------- # have you pressed button a?
    do something
elif button_b.was_pressed: <----- # have you pressed button b?
    do something
else: <---------------------------- # have you done neither?
    do something
```

☐ Write new code to tell the micro:bit to scroll a message if button B is pressed, and to clear the screen if neither button is pressed.

✓ Hint – if you can't remember how to do this, turn back to Section 1.

# *Section 3 – Accelerometer & Pins*

✓ Using the Accelerometer, the micro:bit knows if it is being moved.

☐ Between your `elif` code and `else` code, add a new line.

☐ Type this code on your new line: `elif accelerometer.was_gesture("shake"):`

☐ Expand this code so that when the micro:bit is shaken, it shows `ANGRY` on the display.

✓ Your code should now look something like this:

```
while True:
    if button_a.was_pressed():
        display.show(Image.HEART)
        sleep(1000)
    elif button_b.was_pressed():
        display.scroll('Hello')
    elif accelerometer.was_gesture("shake"):
        display.show(Image.ANGRY)
        sleep(1000)
    else:
        display.clear()
```

☐ Download your code to your micro:bit to make sure it works as you expect.

✓ Using the Pins, the micro:bit knows if it is being touched.

☐ Add another `elif` statement, and add the following code: `pin0.is_touched()`

✓ Hint – remember your colons **:**

☐ Expand this code so that when pin0 is touched, the micro:bit scrolls a message.

☐ Download your finished code to your micro:bit.

☐ With your right hand, tightly grip the GND pin with your thumb and first finger. Make sure that you are only touching GND and not the other pins.

☐ With your left hand, tightly grip Pin 0 with your thumb and first finger. Make sure that you are only touching Pin 0 and not the other pins.

☐ Does this run your `pin0.is_touched()` code?

✓ If you find any errors, go back to your code and see if you can find the problem.

# *Section 4 – Compass & Thermometer*

- ☐ Delete all of your previous code except `from microbit import *`
- ✓ Depending on which direction the micro:bit is facing, it will have a compass heading in degrees, between 0º and 360º.
  This compass heading is given as a number, but the `display.scroll()` function can only show strings, which means words, so you need to convert it.
- ☐ On a new line, type the following code:

```
while True:
    display.scroll(str(compass.heading()))
```

- ☐ Download your code to the micro:bit.

- ✓ Once your code is on the micro:bit, it will ask you to calibrate the compass.
  To do this you need to move the micro:bit in a big flat circle.
  This is easier to do if the micro:bit is unplugged from the computer.
- ☐ Carefully remove the micro USB end of your cable from the micro:bit.
- ☐ Collect your battery pack and plug it into your micro:bit.

**It is important that the micro:bit is not connected to the computer while the battery pack is plugged in. This may send too much power through it, and it may overheat. If you need to download new code, remove the battery pack first.**

- ☐ Follow the micro:bit's instructions to calibrate it, and ask for help if you need it.

- ✓ You can divide a circle into four quadrants, to be North, South, East and West. You are going to tell the micro:bit to show an arrow when it is facing towards the North quadrant.
- ☐ Try the following code:

```
if compass.heading() >= 315 or compass.heading() <= 45:
    display.show(Image.ARROW_N)
```

- ☐ Download this code to your micro:bit, does it correctly point North?
- ✓ Hint – you will need to write another couple of lines to clear the arrow when it is not pointing North! If you can't remember how to do this, turn back to Section 3.

- ✓ You can use the techniques that you learnt to make the compass, to make a thermometer.
- ✓ The code you will need is: `temperature()`
- ☐ Write new code to display the temperature on the micro:bit's display.

- ✓ The temperature that the micro:bit is showing is the temperature of its processor chip, which means that it will be a little bit warmer than the room.

*To get the real temperature, you can do some scientific testing to work out the difference!*

## *Section 5 – Using the Pins*

- ✓ You can connect extra things to the micro:bit's pins:

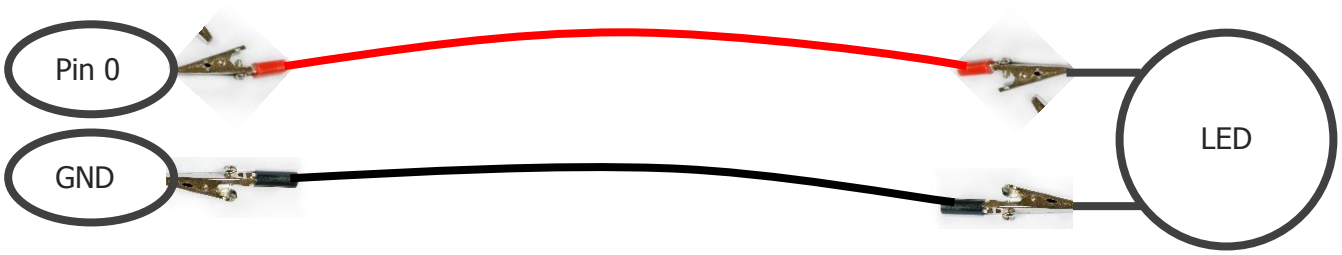
Inputs, such as buttons and sensors can be controlled with this code: `pin0.read_digital()`

Outputs, such as lights and speakers can be controlled with this code: `pin0.write_digital()`

- ✓ If you have crocodile clips and an LED, follow these steps to make it flash.
- ☐ Type in the following code:

```
while True:
    pin0.write_digital(1)
    sleep(100)
    pin0.write_digital(0)
    sleep(100)
```

- ✓ This code will tell whatever is connected to Pin 0 to turn on `(1)`, then turn off again `(0)`.
- ☐ Connect one end of each crocodile clip to each LED leg – one leg is longer than the other.
- ☐ Connect the other end of the clip on the **shorter** leg to the GND pin on the micro:bit.
- ☐ Connect the other end of the clip on the **longer** leg to Pin 0 on the micro:bit.
- ☐ Download your code to the micro:bit and check that it works as you expect.

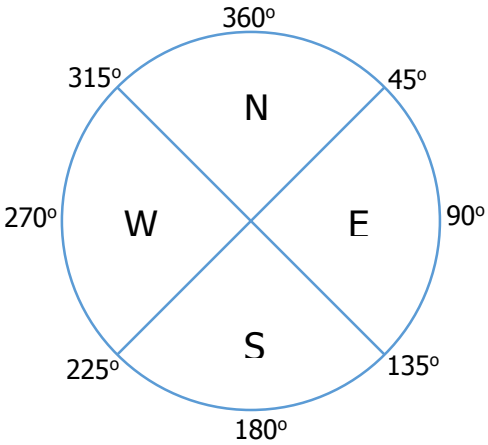



**Well done! You have finished the BBC micro:bit Tour!**

# *Challenges!*

## *Choose one of the challenges below, or create your own idea!*

A. Make a working compass, so that it tells you when the micro:bit is facing North, East, South or West. Then upgrade your compass to also show NE, SE, SW and NW.


B. Make a temperature monitor, that tells the temperature, but also gives a warning when the temperature is very hot, or very cold.

C. Make a 'multi-tool', that can tell you the direction, the temperature, and also give a warning when the micro:bit is moved (using the accelerometer) or touched (using the pins).

D. Make a sensory device, which shows different pictures or messages when different buttons are pressed and pins are touched.

**For more ideas, visit:**

**microbit.org/ideas**

**microbit-micropython.readthedocs.io**