

Introduction

What you will do

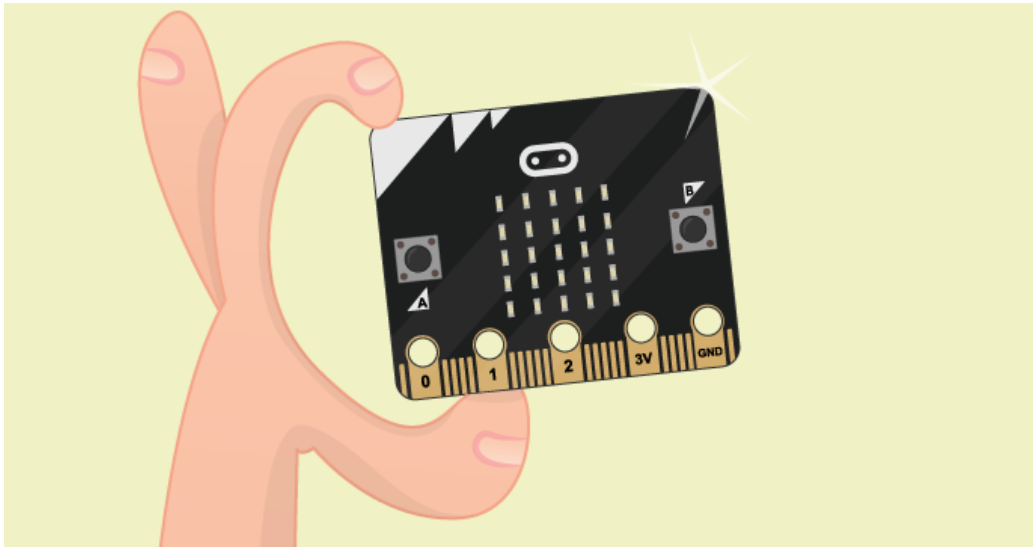


Figure 1:

In this 60 minute workshop you will use the **mu** editor on a Raspberry Pi to write MicroPython programs and run them on the micro:bit.

MicroPython is a small but very fast version of Python 3 that has been specially designed to work on microcontrollers such as those found on the micro:bit.

The experiments will show you some of the things the micro:bit can do. You can use them as a base to build your own projects.

At the end of this workbook you will find some links to help you explore once the workshop is over

You don't have to finish everything today

I hope you'll have plenty of fun things to do over the next 60 minutes, but don't feel you have to do all of the experiments today. You will be able to

keep this workbook and use it once the workshop is over.

Let's get started with *mu*!

1. On the Raspberry Pi, open *Mu* from the main menu under **Programming**.
2. A new window should open up that looks like this:

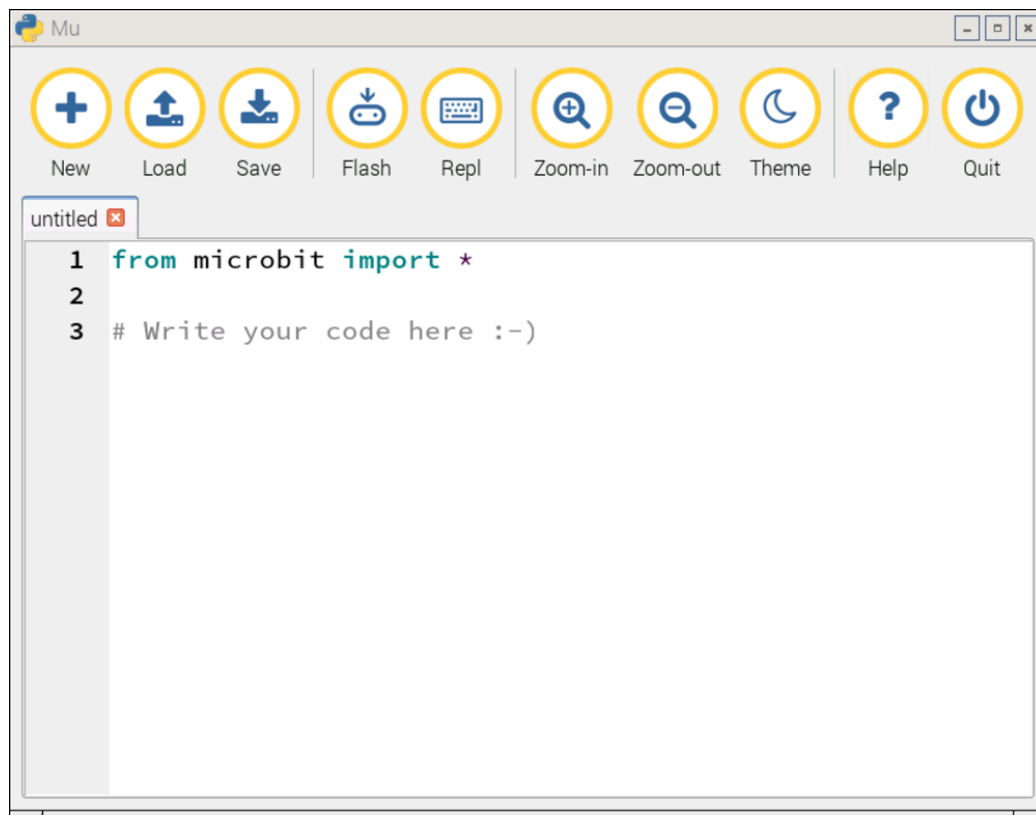


Figure 2: mu screenshot

Plugging in your micro:bit

The micro:bit has a micro USB port that you can use to connect it to your Raspberry Pi. This will provide a power and data connection.

1. Connect your Raspberry Pi to the micro:bit using a USB A-to-micro-B cable, as shown below:

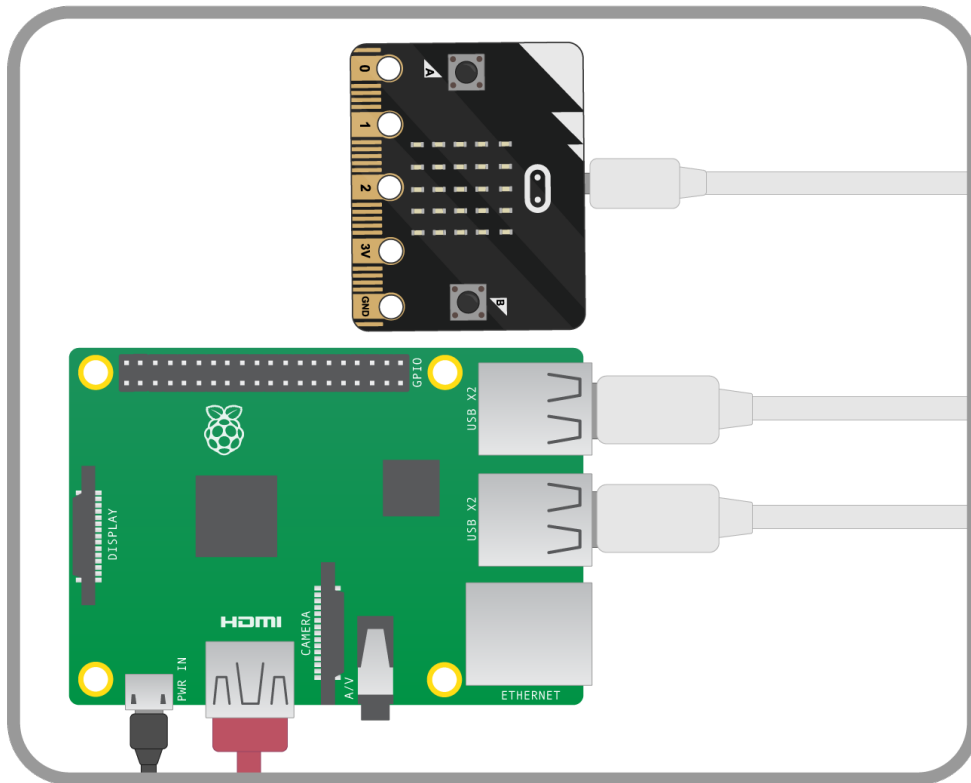


Figure 3: usb setup

2. You'll know that the micro:bit has connected to your Raspberry Pi, because a dialogue box should pop up like the one below:
3. This dialogue box might pop up a few times while you're playing with the micro:bit. You can simply click on **Cancel** when it does.

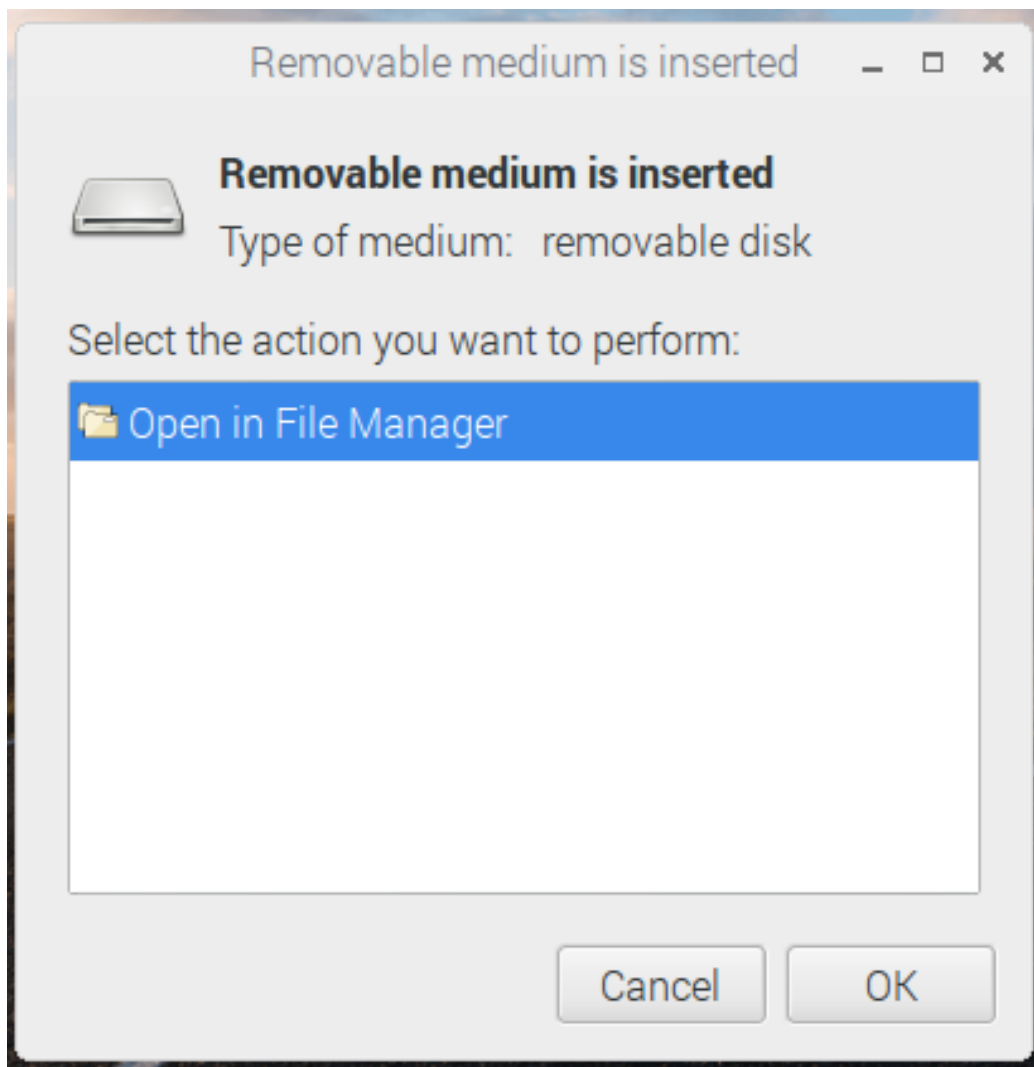


Figure 4: screen2

Using mu

The mu software has been designed with young learners in mind. It has a very easy to use interface, and most of the menu items should be self-explanatory.

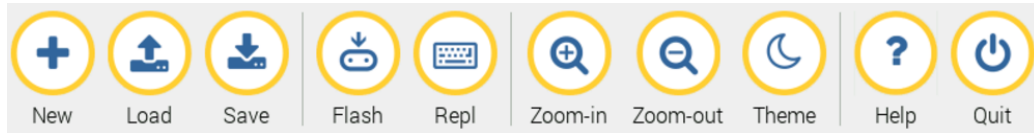


Figure 5: screen3

1. The **New** button will open a *new* file. In mu this is done in a new tab. Have a go opening a few new files, and then closing them again.
2. The **Load** button is for opening existing code that you have written.
3. The **Save** button saves any work you have in the visible tab.
4. The **Flash** button will push your code onto the micro:bit. You'll learn more about this later on.
5. The **Repl** button opens an **interactive shell**. This is covered in the next section.
6. The **Zoom** buttons will alter the size of the text in the window.
7. The **Theme** button switches between **light** and **dark** themes. You can choose your preference.
8. The **Help** button will open the Epiphany web browser and take you to the help pages.
9. The **Quit** button will close mu.

Hello World!

Typing in a program

In the **mu** editor, click the + (new) icon. You should see a new tab open in the editor.

That's where you will type your code.

The first program you'll run is the micro:bit version of *Hello World*.

Here's the short program:

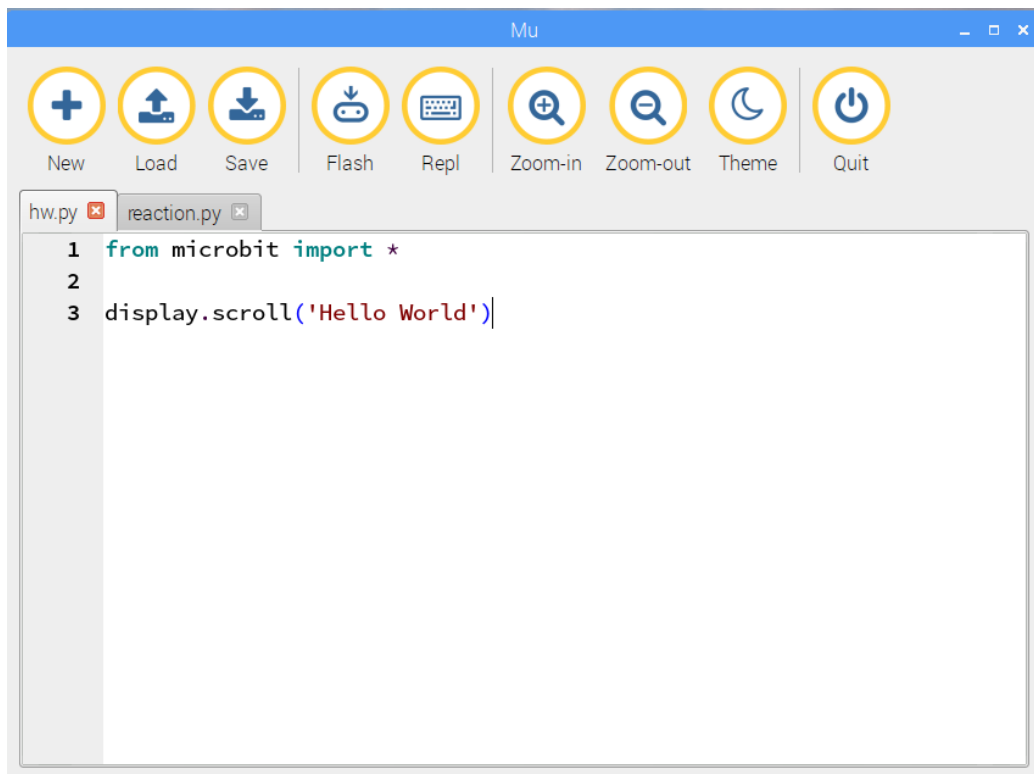


Figure 6: Hello World program

Here's what each line means:

1. `from microbit import *` tells MicroPython that you want to use the

microbit module. (In Python, a module is some code that is already written, ready for you to use if you want to.)

2. `display.scroll('Hello World')` will scroll the string 'Hello World!' across the LEDs on the micro:bit.

Save it, Flash it, Watch it run!

1. Now you can save your file. Click on *mu*'s **Save** button. Call the file `hw.py`
2. Next, you need to use the mysterious **Flash** button. Press the button and a dialogue box should appear:

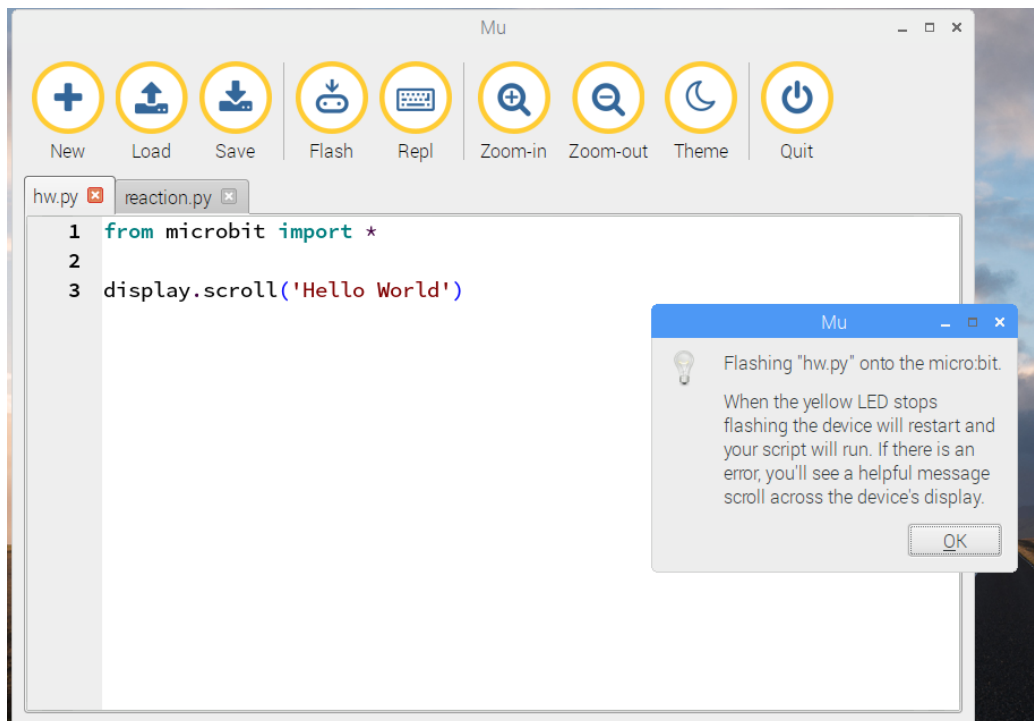


Figure 7: Flashing dialogue

After a few seconds, another dialogue will open, telling you about the re-mountable file system. You can close both dialogues.

Once your program has been *flashed* (installed in the micro:bit's memory), you should see 'Hello World' scroll across the LEDs on the front of the micro:bit.

Troubleshooting

The display will finish quite quickly. If you missed it, just press the button that is underneath the microbit next to the USB cable. This will restart the micro:bit and your program will re-run.

If you don't see the display that you expected, here are some steps to help you find out what went wrong.

1. Carefully check what you typed. It's easy to make a mistake, and the computer cannot guess what you meant to type! If you spot a mistake, correct it and flash the program again.
2. If you can't spot a mistake, ask someone else to check your program. It's often easier for them to spot a mistake than it is for you, because you may be seeing what you meant to type rather than what is actually there!
3. If that doesn't help, ask me. Don't be embarrassed. Just about everyone makes some mistakes when they program.

Images

The micro:bit only has 25 LEDs but that's enough for it to display recognisable images.

In this experiment you will write code to

1. display images on the LEDs.
2. change the display while your program is running,
3. display an animation and
4. create your own images

You may not have time to do all these activities today but you can try them out later.

Displaying a happy face

Open a new tan in *mu* and type in this program:

```
from microbit import *  
display.show(Image.HAPPY)
```

You've seen the first line before. It says you want to use the mircobit module (library).

The second line tells the micro:bit display to show an image. In this case, the image is pre-defined in the microbit module, and you can refer to it as `Image.HAPPY`

Save your program as `happy.py` and click *mu*'s **Flash** button. Two dialogs will appear again. Close the dialogs as before. When the flashing stops you should see a happy face displayed on the micro:bit.

Changing the display to a sad face

The next program is a little longer, but it starts the same as the previous one.

In the `happy.py` window, add tow lines to the end of the program:

```
sleep(1000)  
display.show(Image.SAD)
```

Your editor window should look like this:

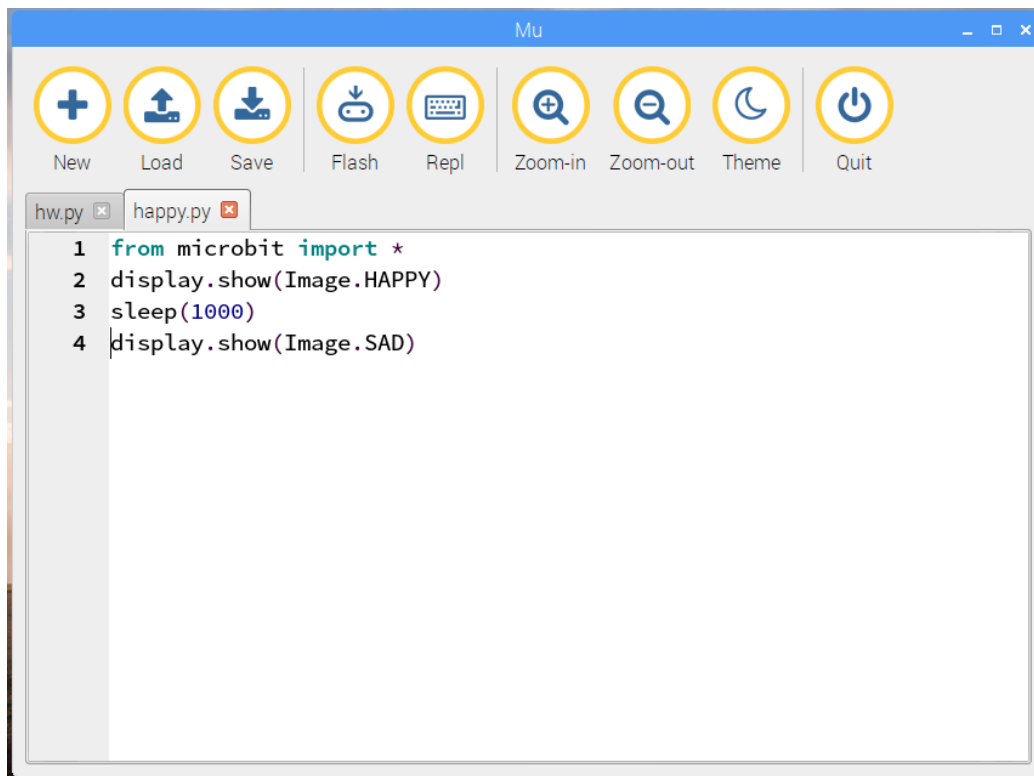


Figure 8: Happy then sad

Use *mu*'s **Flash** button to install your program.

When the program runs you should see a happy face appear for a second. Then it will be replaced by a sad face which will stay on the display until you restart the micro:bit.

Making your own images

Animation - images that change

Buttons

What you will do

Here's why!

Let's go!

Checking a button

While loops

Moving and shaking

What you will do

Here's why!

Let's go!

Measuring acceleration

Spirit Level

What's next

More stuff the micro:bit can do

Compass

Radio

Talking to the Pi

Reaction Timer

Fortunes on the micro:bit

Talking to Chips

Robotics

C2Pi

Where to find out more

micro:bit foundation

Raspberry Pi foundation

microbit news

RAREblog

Moving and shaking

What's next

If you want to explore the workbook later you will need a micro:bit and another computer to program it. The *mu* editor runs on the Pi, but it is also available for Microsoft Windows, MacOS and other versions of Linux.

More stuff the micro:bit can do

Where to find out more

Licence

Unless otherwise specified, everything in this repository is covered by the following licence:



MicroPython and micro:bit in 60 minutes by Romilly Cocking is licenced under a Creative Commons Attribution 4.0 International License.

Based on works at <https://github.com/raspberrypilearning/getting-started-with-microbits> and <https://github.com/romilly/pi-towers-workshop>

Getting started with micro:bits by the Raspberry Pi Foundation is licenced under a Creative Commons Attribution 4.0 International License.