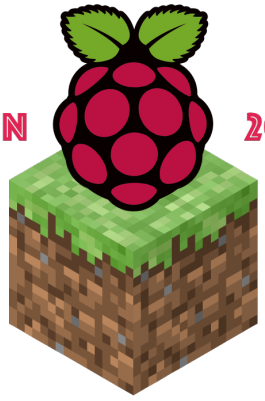
KIWI PYCON 2016 DUNEDIN



Young Coders

Supporters and Sponsors

Thanks to our awesome supporters and sponsors who have made Young Coders 2016 possible.

Workshop 1: Stamp Sheet

After you complete an exercise, ask a Young Coders helper to review your work and receive a stamp on your stamp sheet.

Exercise 1 Hello World	Exercise 2a Let's Teleport	Exercise 2b Teleportation Tour
Exercise 2c Less Nauseating Teleportation Tour	Exercise 3a Blockomancy	Exercise 3b Blockstacking
Bonus Blockomancy +		

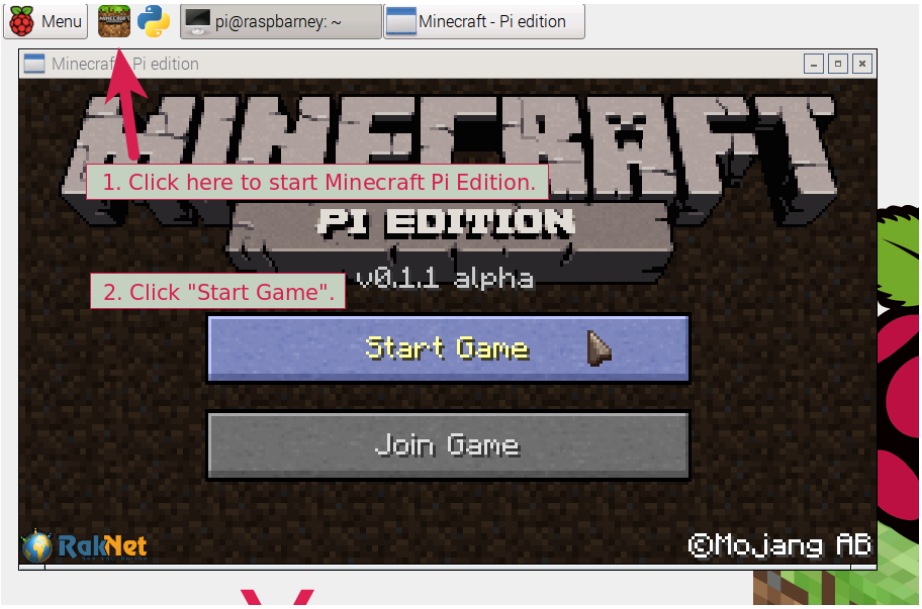
Workshop 1: Let's Learn Python!

Workshop 1, Exercise 1: Hello World

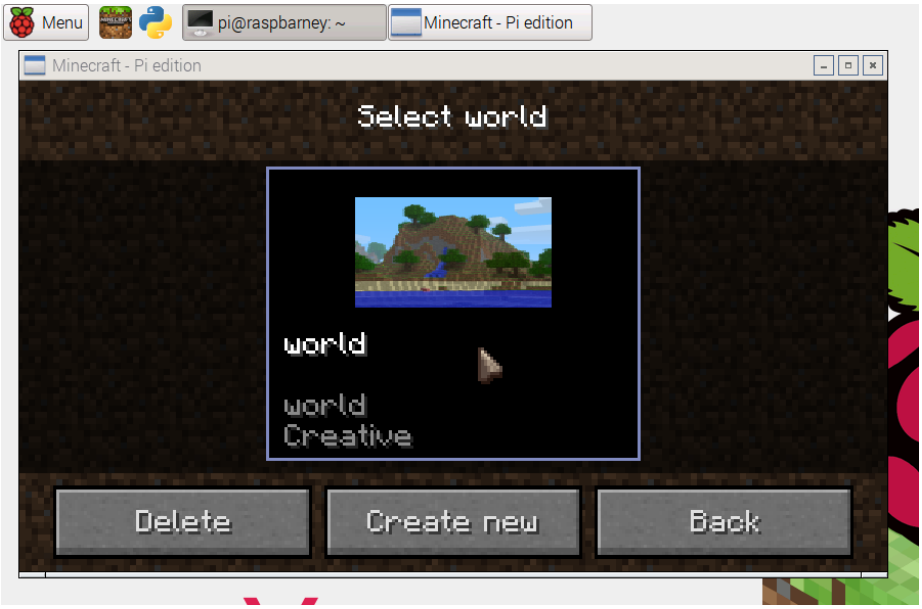
One of the first programs you usually write, when learning how to program is 'Hello World'. 'Hello World' is a simple program which prints the text 'Hello World!' somewhere - usually to a terminal or console window, but today we're going to print 'Hello World!' in Minecraft!

1. Make sure your Raspberry Pi is booted up. You should see a desktop with Young Coders wallpaper and a menu button in the top left. If you don't see a desktop, please ask a Young Coders helper for a hand.

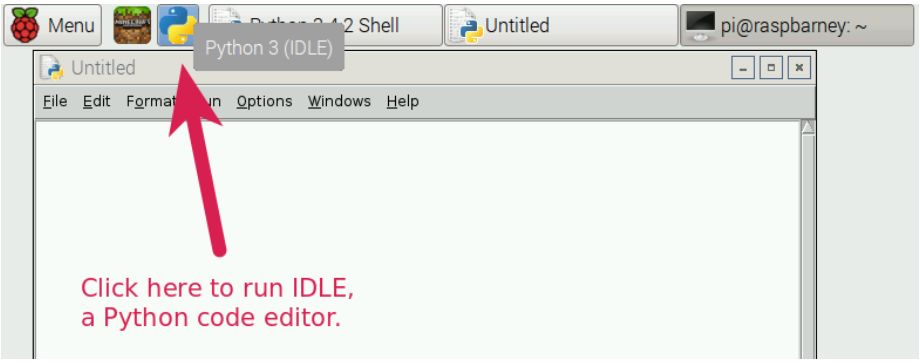
2. Open Minecraft Pi, and click *Start Game*.



3. Open the selected world.



4. Open the python editor IDLE.



Note: To get your desktop cursor back from Minecraft, hit the ESC key.

5. Type the following code in the new editor window:

```
1 from mcpi.minecraft import Minecraft
2 mc = Minecraft.create()
3
4 mc.postToChat('Hello World!')
```

6. Click *File* > *Save* and save this file as `helloworld.py`

If you get a dialog box saying the file already exists, you can say *YES* to overwriting the file.

7. Click *Run* > *Run Module* in IDLE to run this program. You should see the text "Hello World!" appear in your Minecraft chat window.

Learning about variables

In Python, we can store data in something called a variable. We're going to store a *string* in a *variable*. In Python, a string is a bit of text inside either '' or " ". In our earlier code, the data was the *string* `'Hello World!'`.

Let's create a variable called `message` and assign it a different string.

We can then print the message out by calling `mc.postToChat` with it.

1. Edit your program so that it looks like the following one:

```
1 from mcpi.minecraft import Minecraft
2 mc = Minecraft.create()
3
4 message = 'Hello Minecraft World!'
5 mc.postToChat(message)
```

2. Click *Run* > *Run Module* in IDLE to run this program. If you see a dialog popup that says "Source must be saved.", click OK.

You should see the text "Hello Minecraft World!" appear in your Minecraft chat window.

Hint: to switch between Minecraft and IDLE windows, you can use *Alt+Tab*.

Joining string variables

We can also join strings together to print out messages with several variables.

1. Edit your program and make variables for your first name and last name.

```
1 from mcpi.minecraft import Minecraft
2 mc = Minecraft.create()
3
4 first_name = 'Type your first name here'
5 last_name = 'Type your last name here'
6 mc.postToChat('Hello ' + first_name + ' ' + last_name + '!')
```

2. Click *Run* > *Run Module* in IDLE to run this program. You should see yourself greeted in the Minecraft window.

Hint: A quick way to run your code is the *F5* key.

Asking for input

Instead of directly assigning our name to the variable `name`, we can ask our program to prompt us for a name using `input`.

1. Edit your program and assign the result of `input` to the `name` variable.

```
1 from mcpi.minecraft import Minecraft
2 mc = Minecraft.create()
3
4 name = input('What is your name?')
5 mc.postToChat('Hello ' + name + '!')
```

2. Click *Run* > *Run Module* in IDLE to run this program. Once again, save your file if prompted.

Look in the Python Shell window, and you should see your question. Type your name and hit enter. You should see yourself greeted in the Minecraft window.

Well done, you've completed the first exercise! Ask a Young Coders helper to review your code and get a stamp!

Workshop 1, Exercise 2a: Let's Teleport!

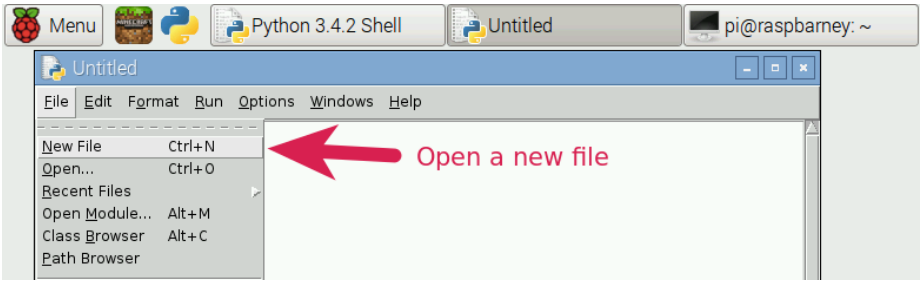
Now that we know about variables, we can use them to teleport Steve all over the place in style!

In Minecraft Pi, you can see your current position in the world in the top left of the screen starting with the label `pos:`

Coordinates have 3 parts which we can use to represent a position in 3d space: `x`, `y` and `z`. `x` and `z` are your horizontal position in the world and `y` is your height.

Let's create variables for `x`, `y` and `z`.

1. In IDLE, Select *File* > *New File* to open a new editor window.



2. Type in the following code:

```
1 from mcpi.minecraft import Minecraft
2 mc = Minecraft.create()
3
4 x = -5.4
5 y = 14.0
6 z = -21.6
7
8 mc.player.setTilePos(x, y, z)
```

Remember, if you want to go back to the start, change x, y and z back to their original values.

3. Click *File* > *Save* and save this file as `teleport.py`
4. Click *Run* > *Run Module* in IDLE to run this program. You should see Steve teleported to a new location.

You should see you have teleported a short distance forward, and are now standing closer to the Young Coders sign.

5. Change x, y and z back to your original coordinates (x = -5.4, y = 12, z= -14) and re-run your program.

6. Not try to teleport to the top of the tower on the left by guessing the coordinates. Remember if you go in the wrong direction, you can change your coordinates back to reset them. A good way to figure out the coordinates for the tower, is to walk towards it and check the top left hand corner for your coordinates. Once you think you have x and z correct, try to add a bigger value for y which is your height.

Once you have reached the top of the tower, you'll find a secret message. Whisper the secret message to a Young Coders helper, to complete the exercise and receive a stamp.

Workshop 1, Exercise 2b: Teleportation Tour

You may have noticed that there are two towers in our world. Let's write some code that let's us teleport from our starting location, to the top of the tower on the left, to the top of the tower on the right, and finally back to the start!

1. Modify our previous program, to teleport Steve from one place to another. Don't delete your x y and z for the first tower - we can rename and use them again. (If you do make a mistake and want to undo anything in IDLE you can use CTRL+Z).

The code below is *incomplete* - you will need to fill in the blanks where the code is commented. Comments look like:

```
# Todo: Add variable and x coordinates for tower_2
```

Look for comments in the code, starting with '#' to figure out what you need to fix. You can run your program to test it at any time (F5) while you work on it.

First work out teleporting from your start position to the top of the first tower. Next, figure out how to teleport to the second tower. Finally, finish your program off by teleporting *back* to your start position.

Hint: To work out the rough coordinates for the towers, you can walk up to them and have a look at your coordinates in the world on the top left of your screen. You'll have to adjust your `y` value quite a bit though to make sure you get to the top of the towers!

```
from mcpi.minecraft import Minecraft
mc = Minecraft.create()

start_x = -5.4
start_y = 12.0
start_z = -14.4

# Hint: since you've already worked out how to get to the top of
    tower 1,
# you can just rename your x, y and z to tower_1_x etc.
tower_1_x = 0 # Todo: Add x coordinates for tower_1
tower_1_y = 0 # Todo: Add y coordinates for tower_1
tower_1_z = 0 # Todo: Add z coordinates for tower_1

# Todo: Add variable and x coordinates for tower_2
# Todo: Add variable and y coordinates for tower_2
# Todo: Add variable and z coordinates for tower_2

mc.player.setTilePos(start_x, start_y, start_z)
mc.player.setTilePos(tower_1_x, tower_1_y, tower_1_z)
```

```
20 # Todo: teleport to tower 2 using mc.player.setTilePos
21 # Todo: teleport back to start using mc.player.setTilePos
```

Got it working? Ask a Young Coders helper to review your code!

Workshop 1, Exercise 2c: Less Nauseating Teleportation Tour

You might have noticed that the last bit of code made Steve teleport around pretty quickly - the poor guy's going to be sick!

Let's see if we can make his Teleportation Tour a little more comfortable…

Python has a handy module called `time` which we can import to do all sorts of time related things, including pausing our program!

1. Add another import statement to the top of your code:

```
1 import time
2
3 from mcpi.minecraft import Minecraft
```

2. Now use `time.sleep()` in your code to pause a few seconds after every time Steve teleports to a new location.

Hint: time.sleep() is a *function* which takes a number of seconds as it's parameter (similar to how `setTilePos` takes coordinates as parameters).

Figure it out? Ask a Young Coders helper if you're stuck, or think you've got it working!

Workshop 1, Exercise 3a: Blockomancy

Let's make blocks with code!

In Minecraft, every block type, from diamond to cactus, has a *block id*. (*Learn to Program with Minecraft* has a handy *block id* reference on page 285).

Let's make a program that creates a cobblestone block at Steve's location.

1. In IDLE, select *File* > *New File* to open a new editor window.
2. Type the following code in the new editor window:

```
1 from mcpi.minecraft import Minecraft
2 mc = Minecraft.create()
3
4 block_id = 4  # 4 is the id for cobblestone
5 pos = mc.player.getTilePos()
6
7 x = pos.x
8 y = pos.y
9 z = pos.z
10 mc.setBlock(x, y, z, block_id)
```

3. Click *File* > *Save* and save this file as `blocks.py`
4. Click *Run* > *Run Module* in IDLE to run your program.
5. Experiment with changing the value of `block_id` and see what other types of blocks you can create.

Workshop 1, Exercise 3b: Blockstacking

Let's extend our block program, so that we can create stacks of blocks.

1. Edit your program

```
from mcpi.minecraft import Minecraft
mc = Minecraft.create()

block_id = 4  # 4 is the id for cobblestone
pos = mc.player.getTilePos()

x = pos.x
y = pos.y
z = pos.z
mc.setBlock(x, y, z, block_id)

y = y + 1 # increase the height of our block by 1 unit
mc.setBlock(x, y, z, block_id)
```

2. Try to write a program that will create an even larger stack of blocks.

Got it working? Ask a Young Coders helper to have a look.

If you still have time left, you can try the following bonus exercice!

Workshop 1, Bonus Exercise: Blockomancy Plus

Let's improve our block creating program, so that it asks for user input. We've got a tricky situation though - the input() function returns a `string`, but the setBlock() function requires a *number* which in programming we call an *integer*.

Learning about Strings and Integers

First let's see what happens when we try to add integers together as strings in the IDLE shell. The IDLE shell is the handy tool you used before to answer your input question. Any code you add after the `>>>` prompt will run after you hit *enter*. It isn't a great way to write long programs, but it can be very handy for learning and exploring.

1. Find the IDLE shell window. It will have a prompt starting with `>>>`.
2. Let's try adding some integers together as strings:

```
>>> '1' + '2'
```

What is the result? The answer you get back is correct, because we have joined the strings `'1'` and `'2'` together, but it won't help us solve any maths problems!

To make the code above return `3`, we can use the `int()` function. When you pass a string to `int()` e.g. `int('1')` it will convert it to an integer.

```
>>> int('1') + int('2')
```

You can also of course type:

```
>>> 1 + 2
```

Updating our Block program to use input() and int()

1. In IDLE, Select *File* > *New File* to open a new editor window.

The program below is incomplete, and you'll need to fix the bits that say `# TODO:`!

2. Type and update the program below to ask for user input for the block_id:

```
from mcpi.minecraft import Minecraft
mc = Minecraft.create()

block_id = # TODO: get block_id from user input, remember to convert
    to an integer!
pos = mc.player.getTilePos()

x = pos.x
y = pos.y
z = pos.z
mc.setBlock(x, y, z, block_id)
```

3. Click *File* > *Save* and save this file as `blocks_plus.py`
4. Click *Run* > *Run Module* in IDLE to run your program.

Does your program work? Ask a helper to check your code.

Well done!

Workshop 2: Stamp Sheet

After you complete an exercise, ask a Young Coders helper to review your work and receive a stamp on your stamp sheet.

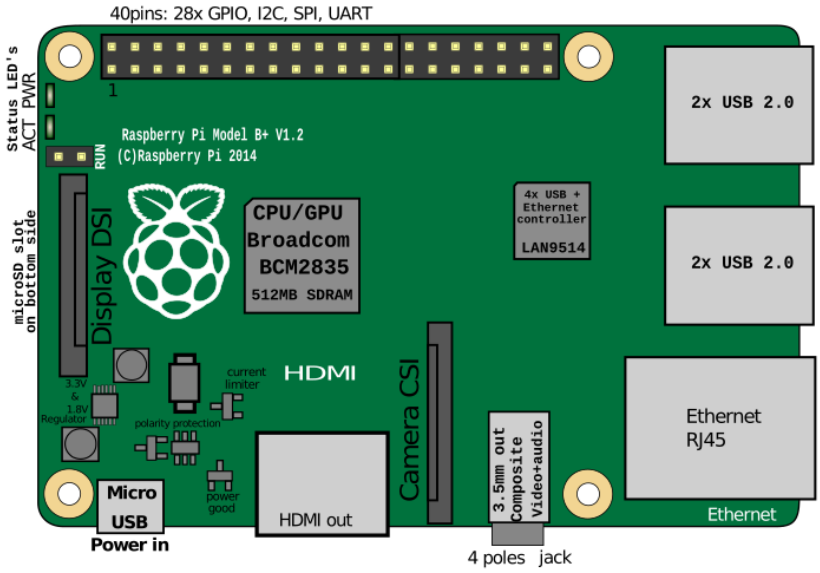
Exercise 1 Creating a Circuit	Exercise 2 Turning the LED on	Exercise 3 Turning the LED on in Minecraft

Workshop 2: Magic Door

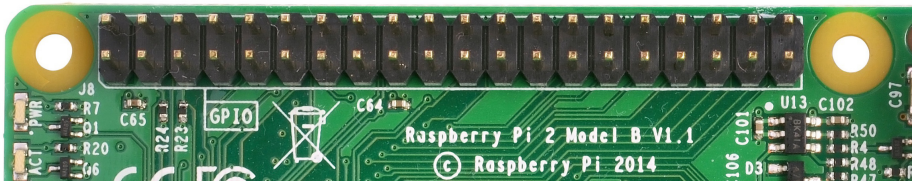
Getting started with the Raspberry Pi

Raspberry Pis are tiny but capable little computers, that have similar stuff inside to what you would find in a desktop PC or Mac:

- CPU (you can think of this as the brain of the computer)
- RAM (the computer's memory)
- HDMI Video port (for connecting a monitor)
- USB ports (for connecting a keyboard, mouse and other devices)
- Audio port (for connecting headphones, or speakers)

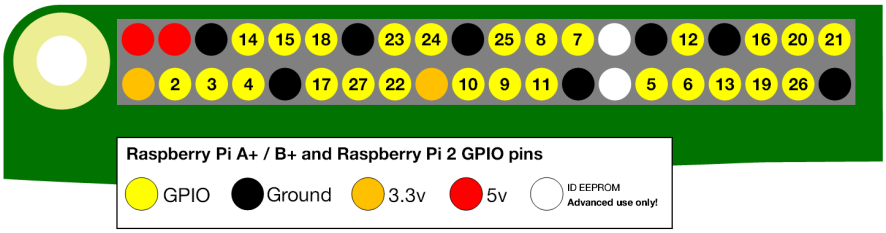


A Raspberry Pi is also a bit different from a desktop computer, as it has one very special feature - the *GPIO Port*.



The GPIO port is the set of 24 pins you see in the picture above. GPIO stands for General-purpose Input Output, and is the way the Raspberry Pi can talk to the outside world, controlling special devices, sensors and other electronic

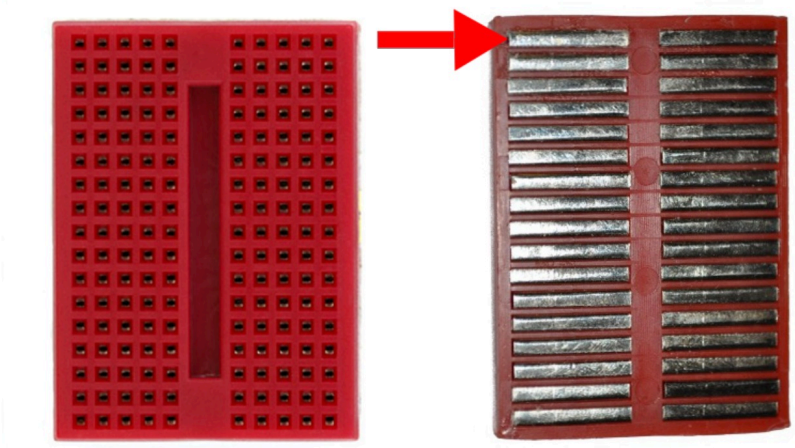
doodads. With GPIO, the Raspberry Pi goes from being just a little computer, to the heart and brain of all sorts of amazing inventions.



In this workshop, we'll be creating a Magic Door in Minecraft, using the Pi's GPIO port and some electronics components. This magic door will turn an LED connected to our PI on and off every time the door opens and shuts in Minecraft.

Components

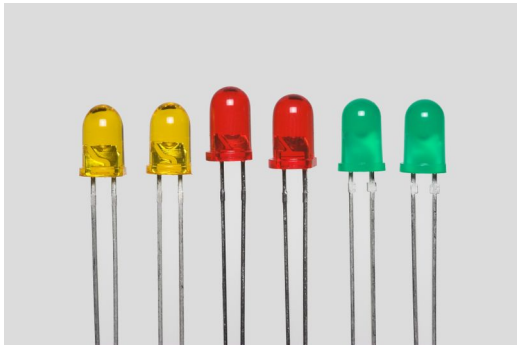
Breadboards



A breadboard lets us create circuits without having to solder components together. This is great when you want to experiment and change components around (the fancy name for this is *prototyping*). You can't see them under your breadboard because of the adhesive backing, but each row of 5 small holes is connected by metal clips.

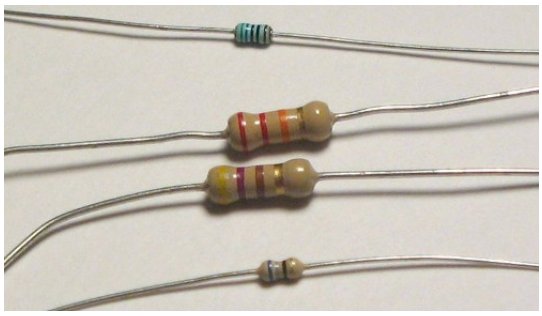
You might be wondering why it is called a breadboard. In the early days of electronics, hobbiests sometimes made circuits on *actual* breadboards. I bet their parents weren't very impressed!

LEDs (Light Emitting Diodes)



LEDs are electronic components that light up when a current is passed through them.

Resistors



A resister is a component that limits the amount of current passing through a circuit. We're going to use one to prevent our LED from burning out too soon.

Male to Female jumper cables

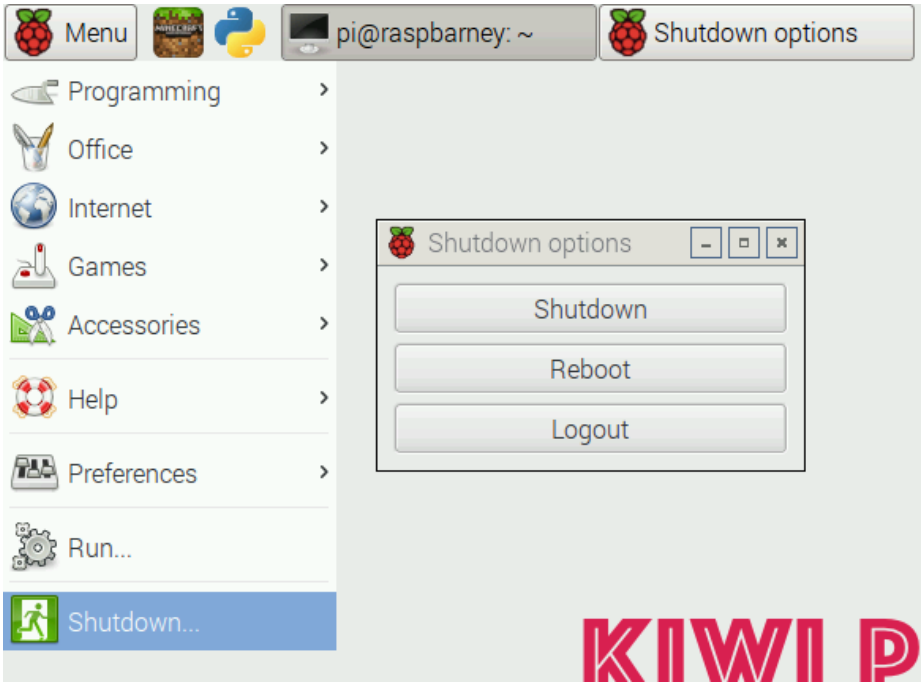
Jumper cables allow us to create a circuit from the GPIO pins to our breadboard.

Workshop 2, Exercise 1: Creating a Circuit

We're going to make a simple circuit using our breadboard now.

While we're making our circuit, it's important that the Raspberry Pi is switched *off* to help prevent damage to our electronic components.

1. Shutdown your Raspberry Pi. Click *Menu* > *Shutdown...* > *Shutdown*.



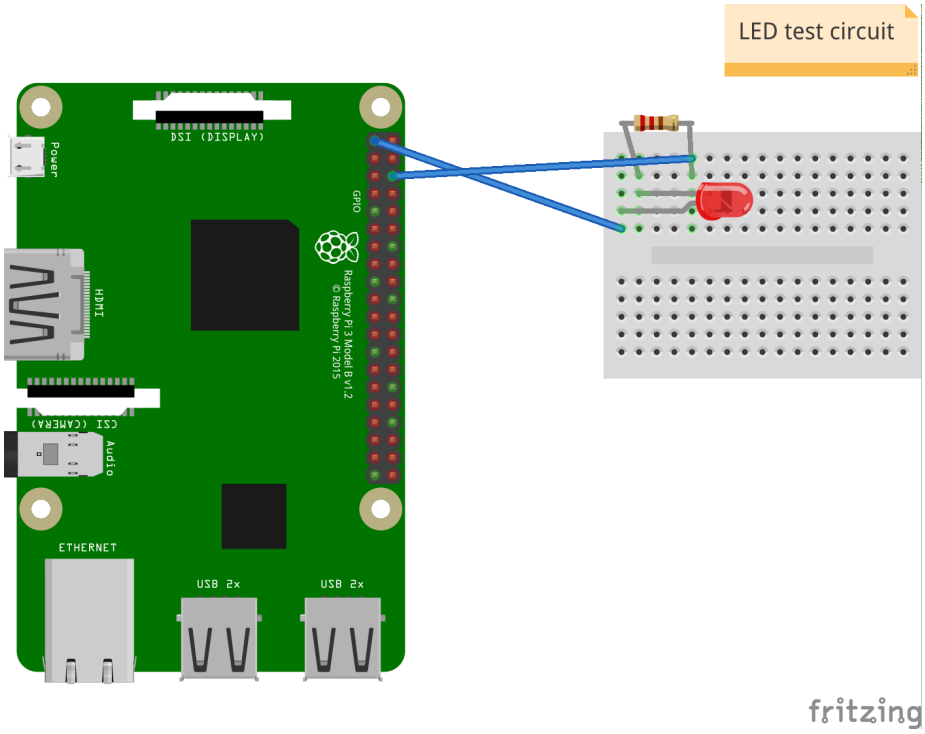
2. Wait a moment, and then disconnect the power.

3. Wire up a test circuit

Our first circuit is going to test that our components are working correctly and we have our circuit wired up the right way. Follow the diagram below to setup your components correctly.

The *female* (the end with a slot) end of your jumpers will connect to the pins on the GPIO port, and the *male* (the end with the pin) end will connect to the breadboard.

Make sure the long leg of the LED is connected to the first row on the breadboard.



Remember, if you get stuck or need some help, raise your hand and talk to a Young Coders helper.

4. Turn your Pi back on by reconnecting the power.

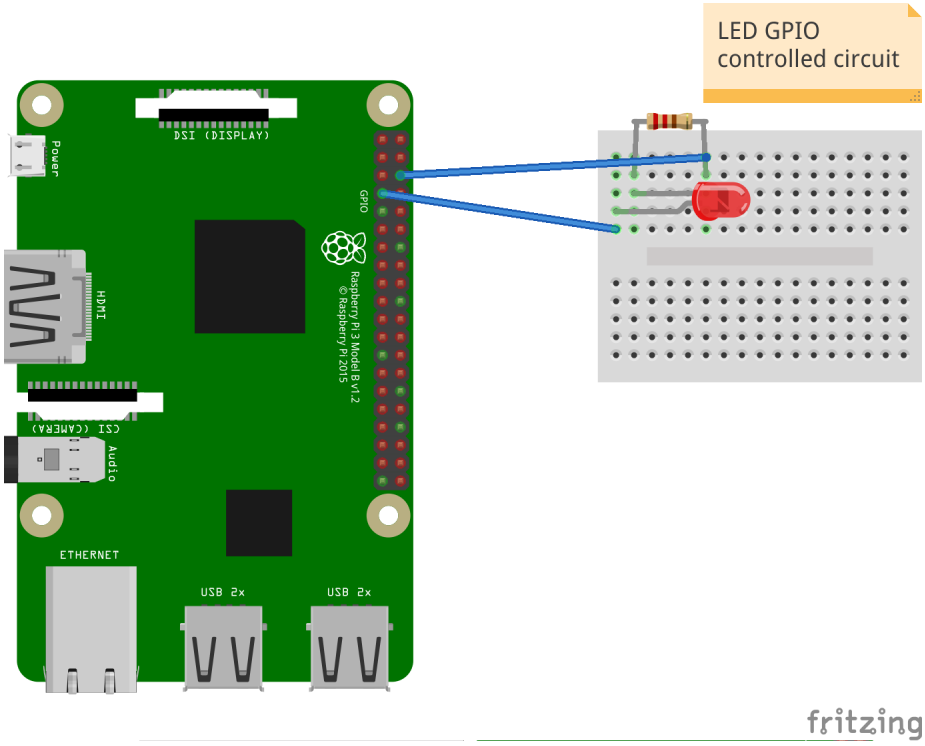
If your circuit is wired up correctly, the LED should turn on when the Pi starts.

5. Shutdown, and turn your Pi off again by disconnecting the power.

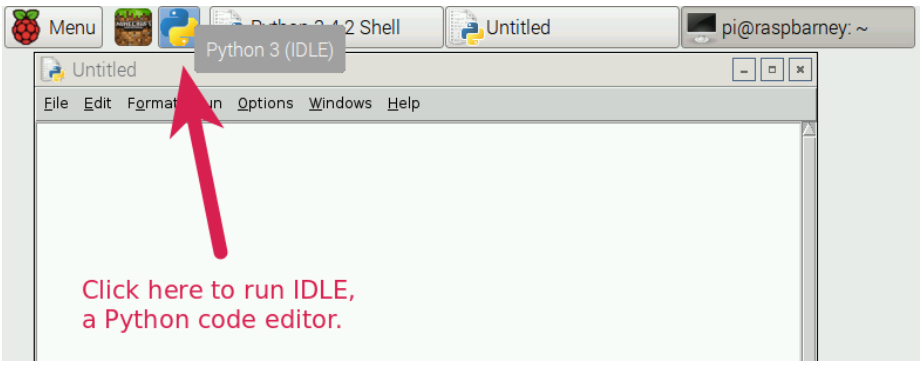
Workshop 2, Exercise 2: Turning the LED on and off with code

1. Now we're going to move the first wire from pin 1, to the GPIO controlled pin 7.

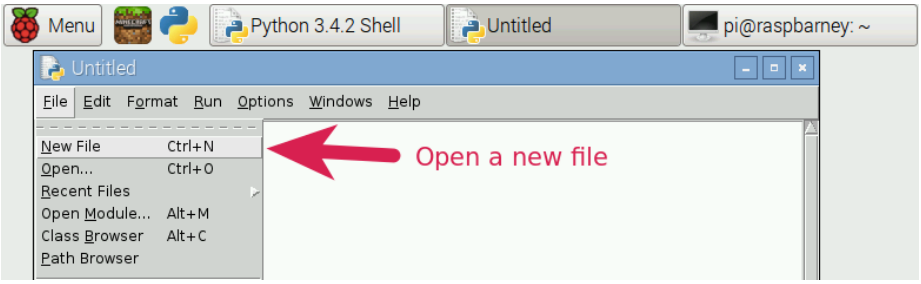
This will allow us to turn our circuit on and off with code.



2. Turn your Pi back on by reconnecting the power cable.
3. Once your Pi has rebooted, open the python editor IDLE.



4. Select *File* > *New File* to open a new editor window.



5. Type the following code:

```
1 import RPi.GPIO as GPIO
2
3
4 GPIO.setmode(GPIO.BOARD)
5 GPIO.setup(7, GPIO.OUT)
6 GPIO.output(7, True)
```

6. Click *File* > *Save* and save this file as `led.py`
7. Click *Run* > *Run Module* in IDLE to run this program. You should see the LED turn on.

To turn the LED off again, you can make a change to your program, and run it again:

Change:

```
1 GPIO.output(7,True)
```

to:

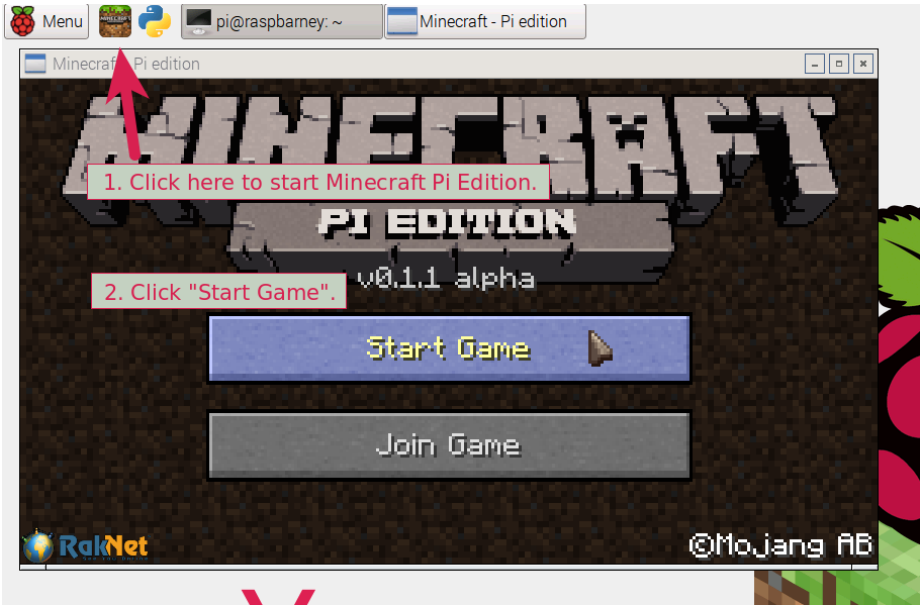
```
1 GPIO.output(7,False)
```

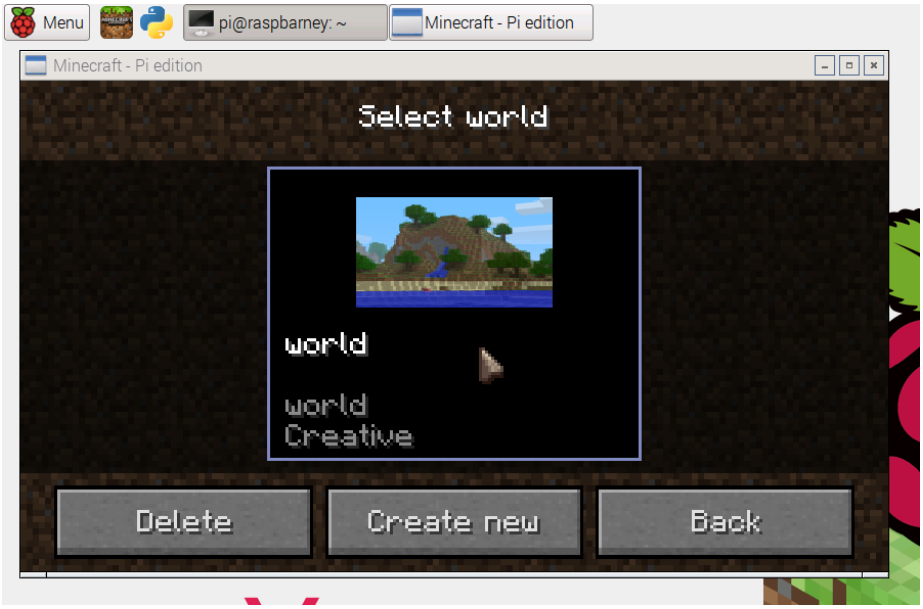
You can also experiment with adding `time.sleep(1)` to your code to make the LED blink on and off.

Workshop 2, Exercise 3: Turning the LED on and off in Minecraft

Now we're going to make a slightly more complicated program that can turn our LED on and off in Minecraft!

1. Open Minecraft Pi, and open the selected world.





2. Create a door. It doesn't matter where it goes really, so just in front of you will do.

3. Select *File* > *New File* in IDLE to open a new editor window.

4. Type in the following code:

```
import time

from mcpi.minecraft import Minecraft
import RPi.GPIO as GPIO


GPIO.setmode(GPIO.BOARD)
GPIO.setup(7, GPIO.OUT)

mc = Minecraft.create()
LED = False

while True:
    events = mc.events.pollBlockHits()
    if len(events) > 0:
        if LED == False:
            GPIO.output(7, True)
            LED = True
        else:
            GPIO.output(7, False)
            LED = False
        mc.events.clearAll()
        time.sleep(1)
```

5. Click *File > Save* and save this file as `ledminecraft.py`
6. Click *Run > Run Module* in IDLE to run this program.
7. In Minecraft, right click the door you made earlier with a sword equiped and you should see the LED turn on. Hit the door again, and the LED should turn off!

Got it working? Ask a young coders helper to check your work, and receive your final stamp!

Well done - you've completed the Young Coders 2016 workshop! We hope you've learned lots of new things!

If there's still time left in the workshop, feel free to explore the world (there are some hidden coloured and metal blocks you can look for), or spend some more time experimenting with the coding you've learned today.

If you're keen to learn more about electronics and coding, check out some of the resources under *Local Education Resources* on the next page.

Additional Resources

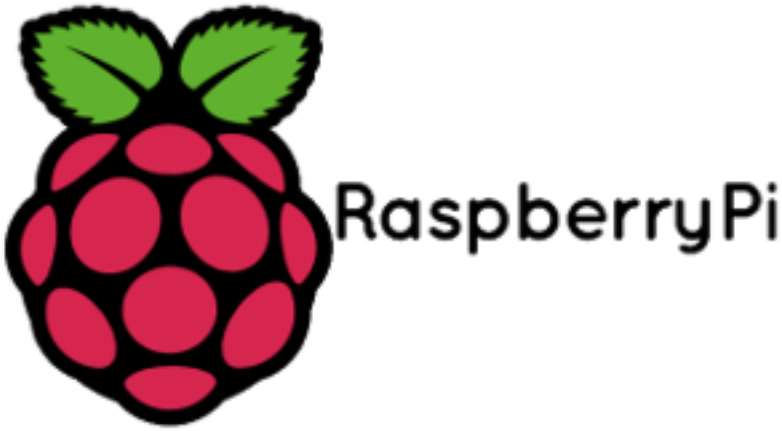
Below are some additional resources, both for inspiration and further learning.

Local Education Resources

Gasworks NZ (https://gasworks.nz) - A volunteer organisation based at the Gasworks in South Dunedin, providing an excellent range of coding, robotics and electronics classes and activities for kids.

Dunedin Makerspace (http://dspace.org.nz) - A community workshop, providing resources for electronics, crafts, art, programming and engineering.

Raspberry Pi



Vendors

Nicegear (https://nicegear.co.nz) - New Zealand based vendor of Raspberry Pi, and other electronics and embedded computing products with excellent customer support.

Tutorials and Blogs

Raspberry Pi StackExchange (http://raspberrypi.stackexchange.com) - An invaluable resource, providing answers to thousands of questions about Raspberry Pis.

Raspberry Pi Kid (https://raspberrypikid.wordpress.com) - A blog by a 14 year old, journaling their learning experiences with coding and electronics using their Raspberry Pi.

Python

Code Academy (https://www.codecademy.com/learn/python) - Self paced Python courses.

Invent with Python (https://automatetheboringstuff.com/) - A free online version of the popular book, showing you how to automate tasks on your computer while learning Python.

Official Python Documentation (https://www.python.org/doc/) - Official docs for python 2 and 3.

Commented Code

Below are commented versions of the code used in the workshops, to help explain what each line of code does.

Workshop 2: Magic Door

Basic LED program

```
1
2 # Import the GPIO library which will let us program the GPIO pins.
3 import RPi.GPIO as GPIO
4
5
6 # Use board pin numbering, counting from 1 from the top left pin.
7 GPIO.setmode(GPIO.BOARD)
8
9 # Set our output to GPIO Pin 7
10 GPIO.setup(7, GPIO.OUT)
11
12 # Turn on GPIO Pin 7
13 GPIO.output(7,True)
```

Full Magic Door program

```
1 # Import the time library, which will let us pause our program for 1
      second.
2 import time
3
4 # Import the Minecraft library, which lets us talk to Minecraft.
5 from mcpi.minecraft import Minecraft
6 # Import the GPIO library which will let us program the GPIO pins.
7 import RPi.GPIO as GPIO
8
9
10 # Use board pin numbering, counting from 1 from the top left pin.
11 GPIO.setmode(GPIO.BOARD)
12
13 # Set our output to GPIO Pin 7
14 GPIO.setup(7, GPIO.OUT)
15
16 # This creates a new connection to our Minecraft world, so we get
      information about what
17 # Steve is up to in the Minecraft world!
18 mc = Minecraft.create()
```

```
19
20 # We need to store the 'state' of our LED, which can be either
21 # ON or OFF.
22 # In code, we can represent ON and OFF with True and False.
23 # Since the LED is OFF to start with, we'll set this to False.
24 LED = False
25
26 # This is the start of an infinite loop.
27 # Code that follows 'while True' will continue to run until
28 # we stop our program.
29 # In game programming, sometimes this is called a 'main loop,
30 # or 'game loop'.
31 while True:
32     # Our Minecaft object 'mc' can tell us lots of different things
33     # about what is happening in Minecraft.
34     # Something happening in Minecraft is called an 'event'.
35     # The events we want to know about are
36     # doors opening and closing, which Minecraft stores as a
           BlockHit.
37     # mc.events.pollBlockHits() will create a List of events if any
           have occured.
38     events = mc.events.pollBlockHits()
39     # 'len' is a function which can tell us how many events are in
           the list of events.
40     # Here we want to check if there have been any events. If there
           haven't, the program
41     # will start over again and check for more events later.
42     if len(events) > 0:
43         # Here we want to check if our LED is turned OFF or ON.
44         # If our LED is turned OFF (False), we want to turn it on by
               setting
45         # GPIO pin 7 to True.
46         if LED == False:
47             GPIO.output(7, True)
48             # Now we need to record that we have turned ON our LED.
49             LED = True
50         # If our LED is already turned ON (True), we need to turn it
               OFF.
51         else:
52             # Send 'False' to GPIO pin 7 to turn it OFF.
53             GPIO.output(7, False)
54             # Record that we have turned our LED OFF.
55             LED = False
56         # Clear the list of events, since we have already processed
               them.
57         mc.events.clearAll()
```

```
58          # Pause our program for 1 second. After 1 second the loop
                will start
59          # from the top again.
60          time.sleep(1)
```