Welcome to the Labs

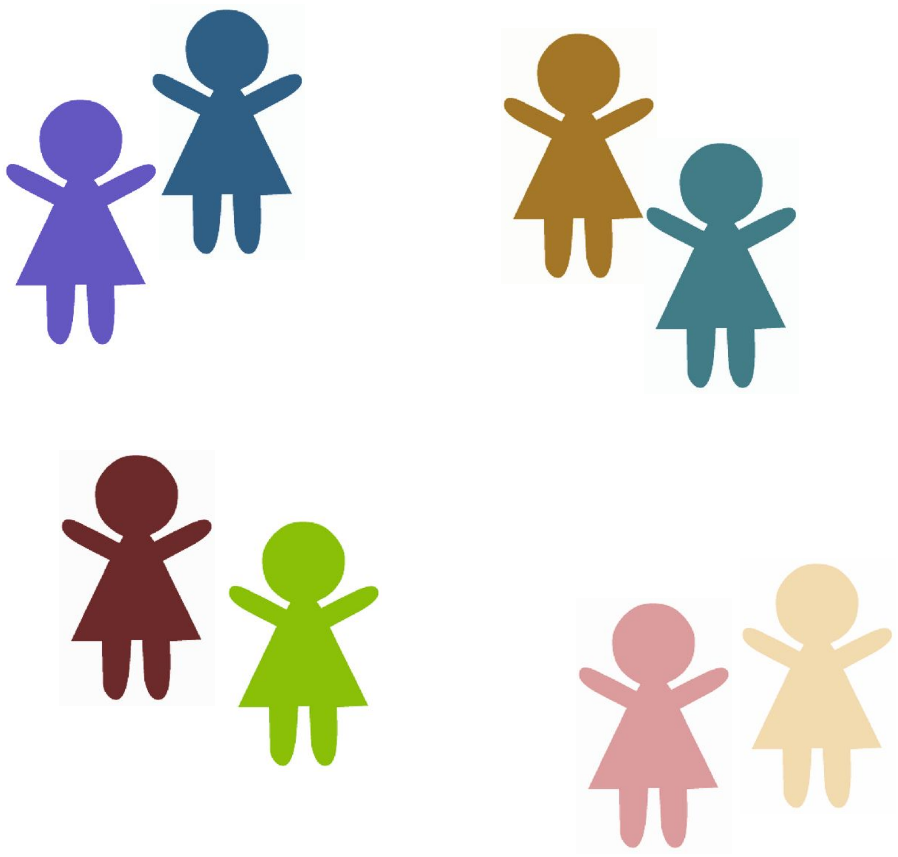
Bop It! - Micro:Bit

Who are the tutors?

Who are you?

Two Truths and a Lie

1. Get in a group of 3-5 people
2. Tell them three things about yourself:
   a. Two of these things should be true
   b. One of these things should be a lie!
3. The other group members have to guess which is the lie



Log on

Log on and jump on the GPN website

girlsprogramming.network/workshop

You can see:

- These **slides** (to take a look back or go on ahead).
- A digital copy of your **workbook**.
- Help bits of text you can **copy and paste**!

There's also links to places where you can do more programming!

Tell us you're here!

Click on the

Start of Day Survey

and fill it in now!

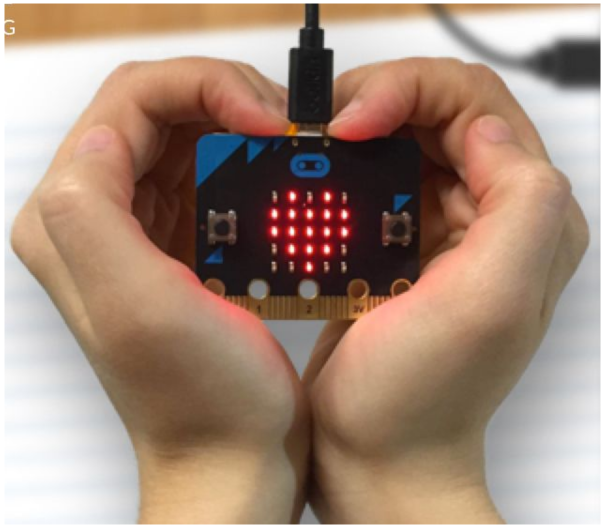
Today’s project!

Bop It! - Micro:Bit

Micro:Bits - IRL

Today we have real life MicroBits to use!

But sad you can't keep them at the end of the day. 😥



If you want one for home (maybe for christmas or your birthday!) they're about $25 .

Find out where to buy them here: https://microbit.org/

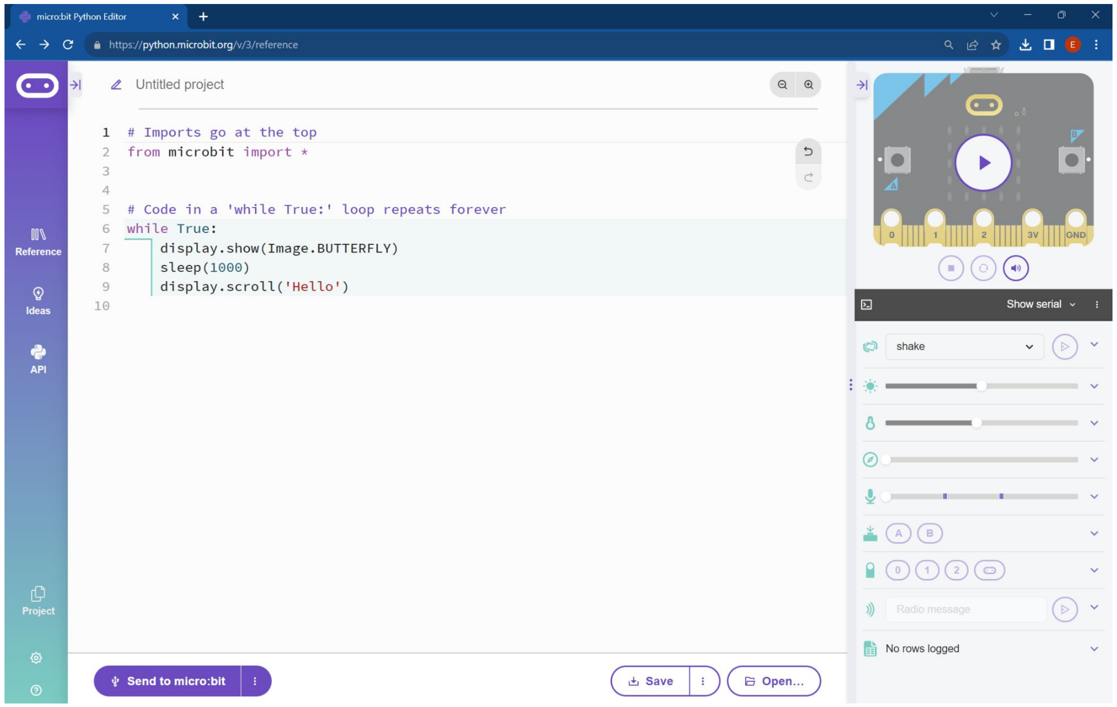
Micro:Bits - Digital

We also have an emulator on python.microbit.org! Which you can use after the workshop! 🎉

The simulator is a fast way to test the code without downloading it.

Use it while you're still working on your code. And then try it in real life.

(Works with Edge and Chrome)



```
# Imports go at the top
from microbit import *


# Code in a 'while True:' loop repeats forever
while True:
    display.show(Image.BUTTERFLY)
    sleep(1000)
    display.scroll('Hello')
```

Using the workbook!

The workbooks will help you put your project together!

Each **Part** of the workbook is made of tasks!

Tasks - The parts of your project

Follow the tasks **in order** to make the project!

Task 6.2: Add a blah to your code!

This has instructions on how to do a part of the project

1. Start by doing this part
2. Then you can do this part

Hints - Helpers for your tasks!

Stuck on a task, we might have given you a hint to help you **figure it out**!

The hints have **unrelated** examples, or tips. **Don't copy and paste** in the code, you'll end up with something **CRAZY**!

Task 6.1: Make the thing do blah!

Make your project do blah ….

Hint

A clue, an example or some extra information to help you figure out the answer.

```
print('This example is not part of the project’ )
```

Using the workbook!

The workbooks will help you put your project together!

Check off before you move on from a **Part**! Do some bonuses while you wait!

Checklist - Am I done yet?

Make sure you can tick off every box in this section before you go to the next Part.

CHECKPOINT

If you can tick all of these off you're ready to move the next part!

☐ Your program does blah
☐ Your program does blob

Lecture Markers
This tells you you'll find out how to do things for this section during the names lecture.

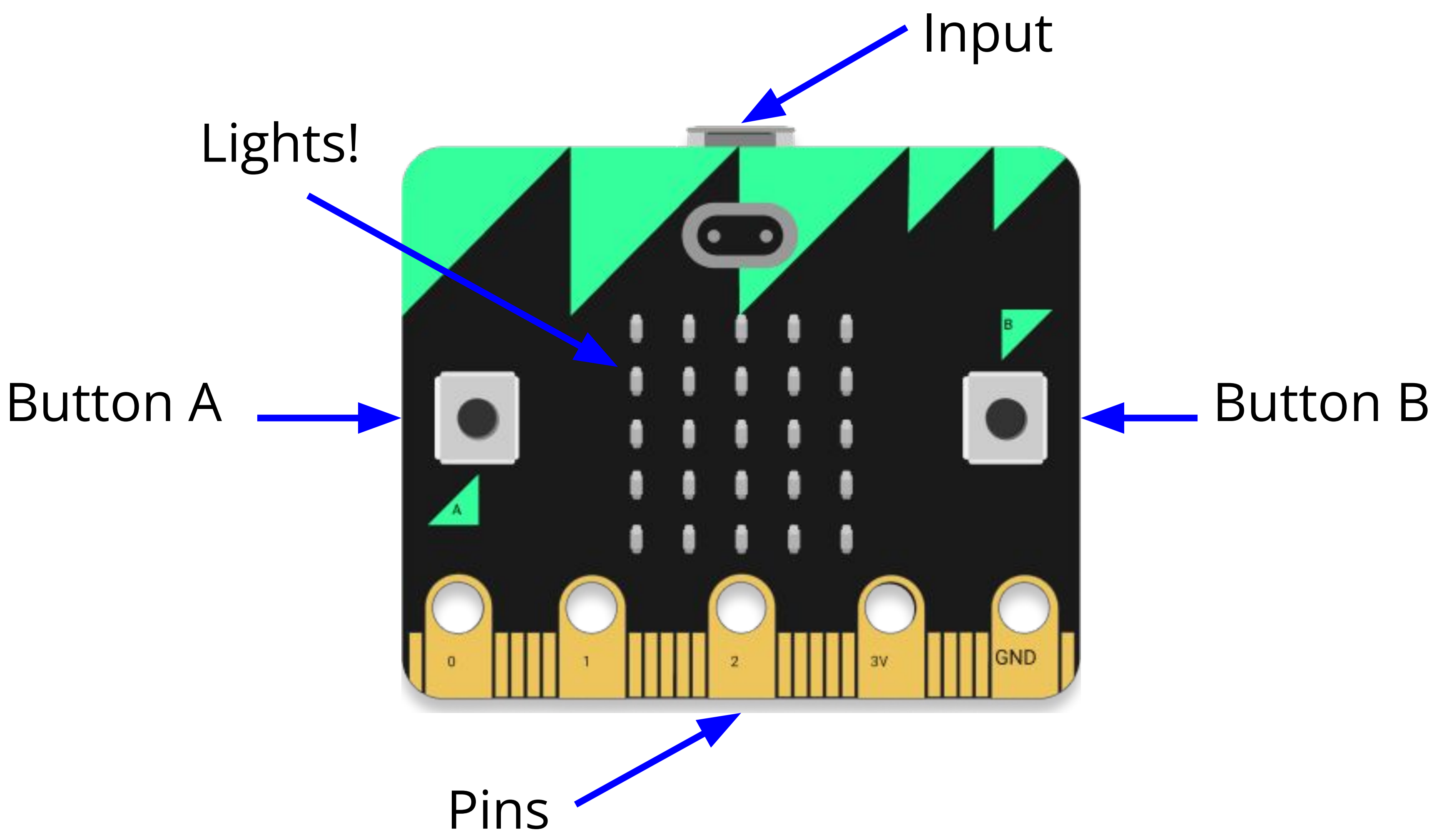
Bonus Activities
Stuck waiting at a lecture marker?
Try a purple bonus. They add extra functionality to your project along the way.

★ BONUS 4.3: Do some extra!

Something to try if you have spare time before the next lecture!

Intro to Micro:Bit

What is a Micro:Bit?



What do the different parts do?

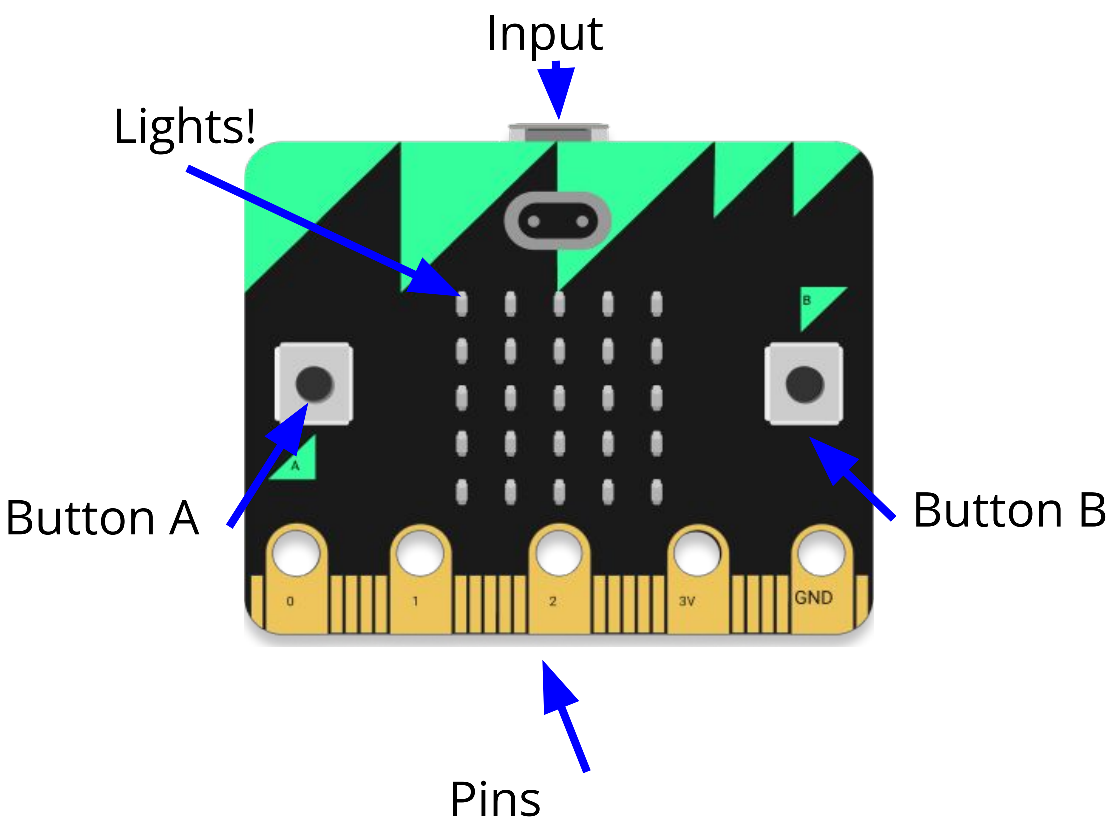
Input: This is how we get code onto our Micro:Bit and tell it what to do!

Buttons: We can press these and tell the Micro:Bit to do different things when we do

Lights: Each of these is a little light that we can turn on. When we turn them on in different patterns we can make images!

Pins: These let us connect the Micro:Bit to other devices like extra buttons

(Front)



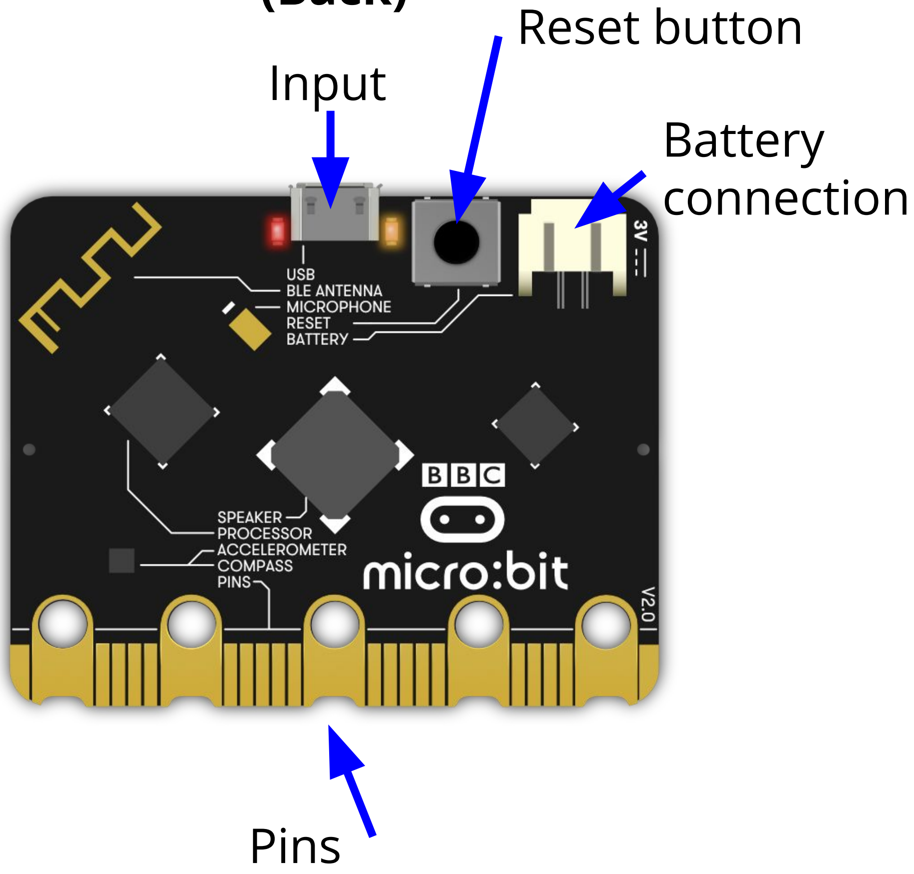
What do the different parts do?

Input: This is how we get code onto our Micro:Bit and tell it what to do!

Reset button: Stops your code and starts it up again

Pins: These let us connect the Micro:Bit to other devices like extra buttons

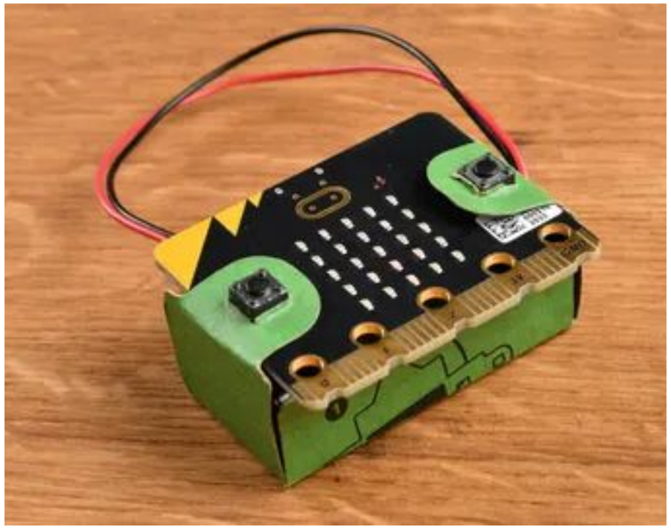
(Back)



Battery pack!

You can use your micro:bit even when it is not plugged into your computer!

You can ask you tutor for a battery pack if you need one.



How do we write code for it?

Micro:Bits use **Python**, which is the programming language that we usually teach here at GPN!

Because they have buttons, lights and other cool stuff we need to make sure that we tell Python that we want the extra stuff for Micro:Bits. We do this using this line of code:

```
from microbit import *
```

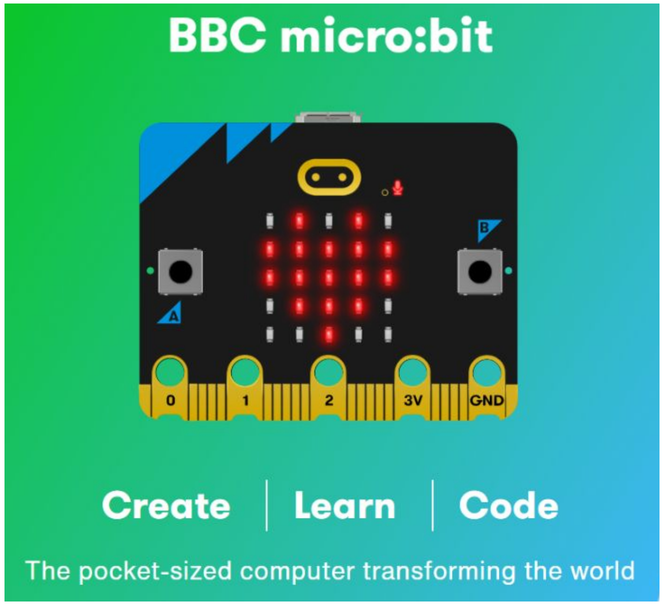
Always make sure this line is at the top of your code!

Using microbit.org!

Today we will be using **microbit.org** to program our Micro:Bits.

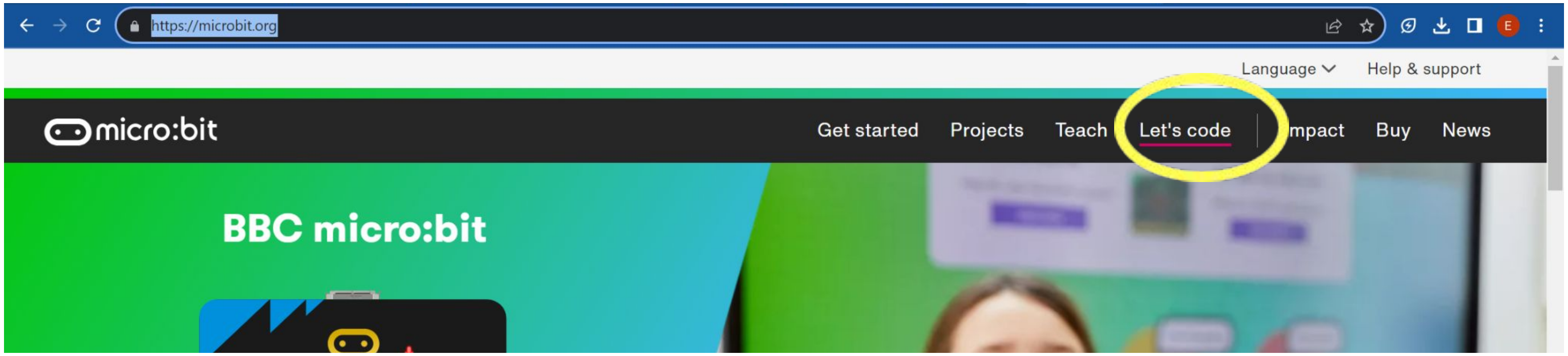
microbit.org has a great Micro:Bit simulator which makes learning how to program them really easy!

We can write and test code for the simulator **and** real micro:bit!



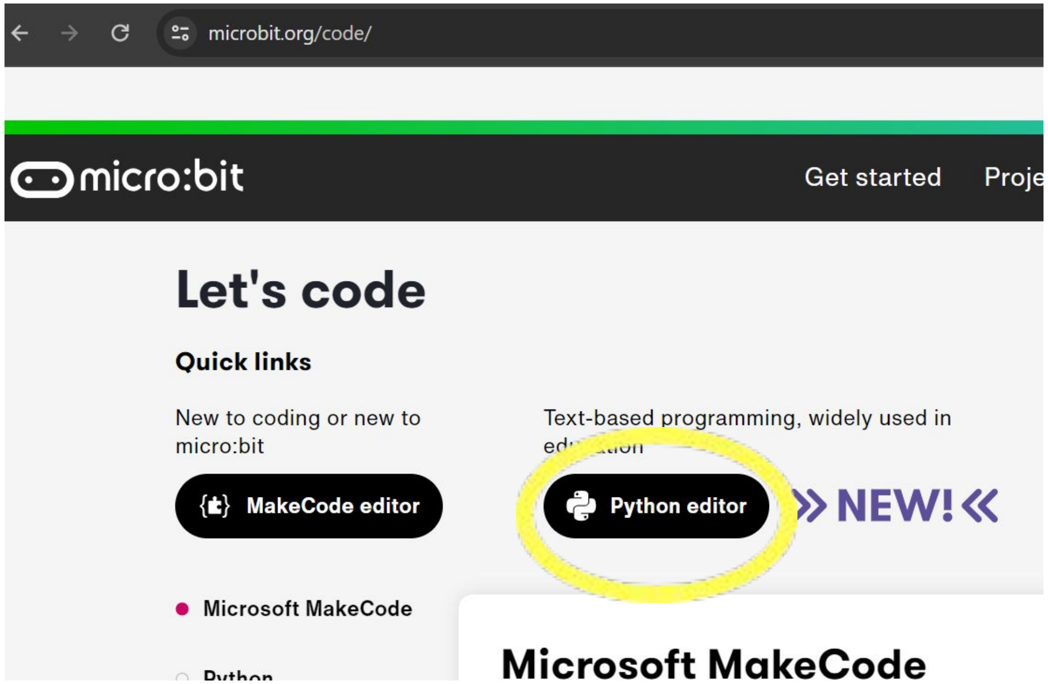
Getting to the website!

Go to microbit.org



Getting to the website!

Click on 'Python editor'



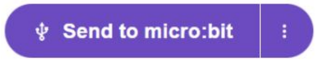
Now we can start programming!

Using a Micro:Bit IRL

It’s fun to mess around with the Micro:Bit online, but it’s also really fun to see your code on a Micro:Bit in real life!

How to send your code to the Micro:Bit

(Please use Google Chrome!)

1. Go to **microbit.org > Let’s Code > Python editor** (or directly to python.microbit.org)
2. **Plug** your Micro:Bit into your computer
3. Click the **send to micro:bit** button on the website to download your code. 

4. Follow the steps that appear.
5. **Wait for the red light** at the back to stop flashing and the code should be running!

- If you want your code to start again from the beginning, press the black “**reset**” button on the back

The Display

Your Micro:Bit has a display! It is the 5 by 5 grid of little red LEDs on the front! You can do some cool stuff with the display like:

Scroll the words “Hello World” across the display

```
display.scroll(“Hello World”)
```

Show an image, like a happy face!

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

Project Time!

Let's get started!

Try to do Part 0 - 1

The tutors will be around to help!

Variables, lists and random!

Storing things for later!

In our game we might have things we want to remember for later!

For example, a score or the list of moves.

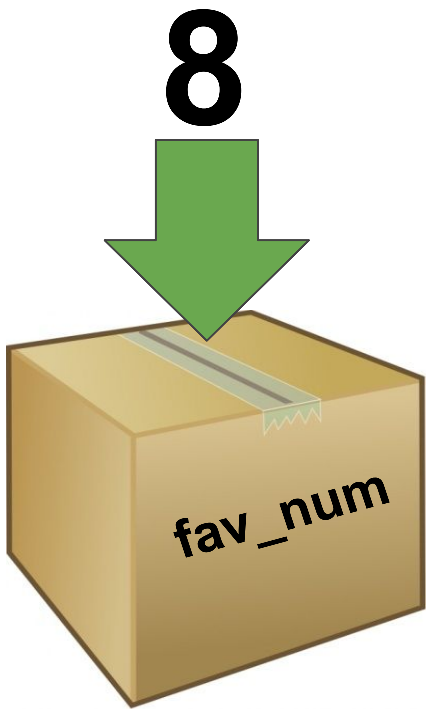
We might even want to change these things throughout the game (like increasing the score)

No Storing is Boring!

It’s useful to be able to remember things for later!
Computers remember things in **"variables"**

Variables are like putting things into a **labeled cardboard box**.

Let’s make our favourite number 8 today!



Variables

Instead of writing the number 8, we can write fav_num.

```
fav_num - 6
   => 2
```

```
fav_num + 21
   => 29
```

```
fav_num * 2
   => 16
```

```
fav_num / 2
   => 4
```

Variables

Instead of writing the number 8, we can write fav_num.

```
fav_num - 6
    => 2

fav_num + 21
    => 29

fav_num * 2
    => 16
```

f   2

We'll come back to this later!

But writing 8 is much shorter than writing `fav_num`???

Using variables

You set variables using one **=** symbol

You can update it by doing the same

Can you guess what each `print` will do?

```
>>> x = 3
>>> print(x)

>>> print(x + x)

>>> y = x
>>> print(y)

>>> y = y + 1
>>> print(y)
```

Using variables

You set variables using one **=** symbol

You can update it by doing the same

Can you guess what each `print` will do?

```
>>> x = 3
>>> print(x)
3
>>> print(x + x)

>>> y = x
>>> print(y)

>>> y = y + 1
>>> print(y)
```

Using variables

You set variables using one **=** symbol

You can update it by doing the same

Can you guess what each `print` will do?

```
>>> x = 3
>>> print(x)
3
>>> print(x + x)
6
>>> y = x
>>> print(y)

>>> y = y + 1
>>> print(y)
```

Using variables

You set variables using one **=** symbol

You can update it by doing the same

Can you guess what each `print` will do?

```
>>> x = 3
>>> print(x)
3
>>> print(x + x)
6
>>> y = x
>>> print(y)
3
>>> y = y + 1
>>> print(y)
```

Using variables

You set variables using one **=** symbol

You can update it by doing the same

Can you guess what each `print` will do?

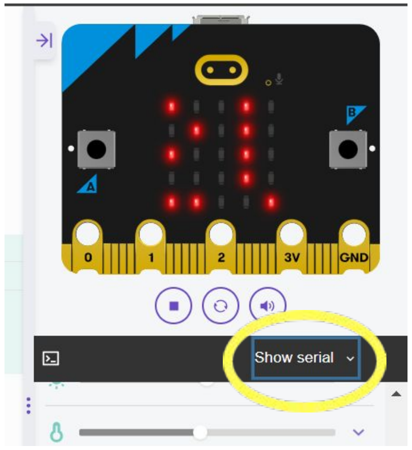
```
>>> x = 3
>>> print(x)
3
>>> print(x + x)
6
>>> y = x
>>> print(y)
3
>>> y = y + 1
>>> print(y)
4
```

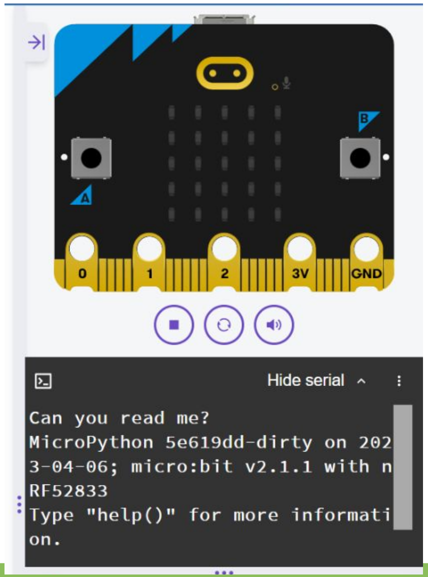
Printing bits

On the microbit.org page, we can print out things too!

We can use `print('put some text here')` to print words and sentences and symbols.

This will be displayed in the *serial* on the right side. You can see this by clicking **Show serial**.

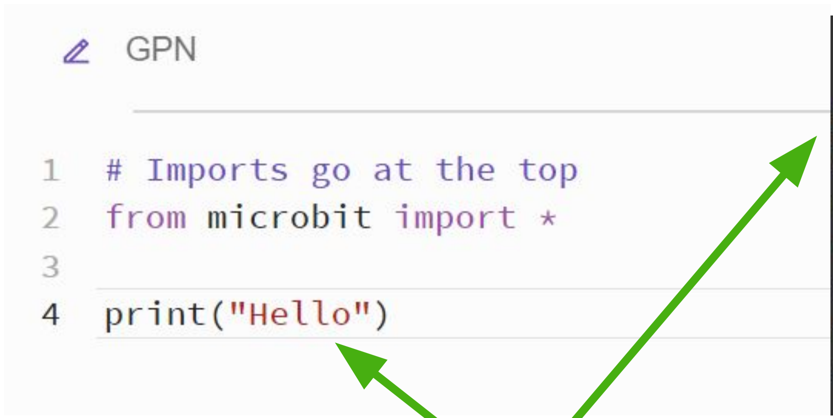




```
Can you read me?
MicroPython 5e619dd-dirty on 202
3-04-06; micro:bit v2.1.1 with n
RF52833
Type "help()" for more informati
on.
```

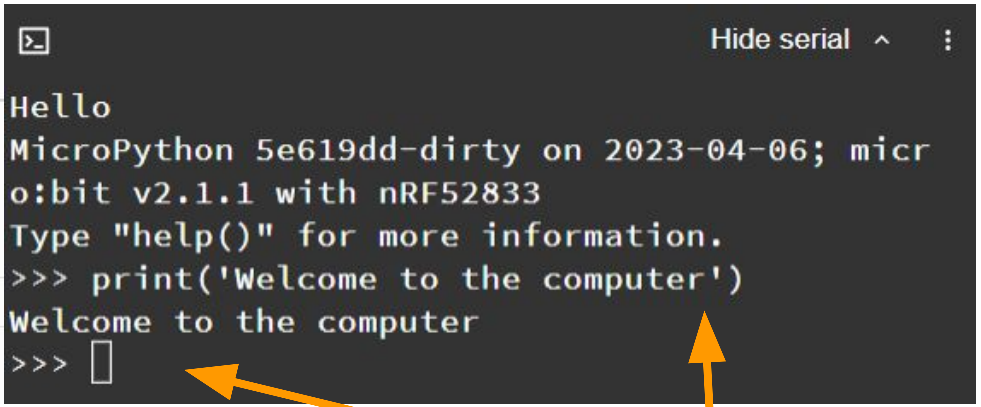
Printing bits

Make sure to press the play button to activate the serial and communicate with the computer directly!



```
# Imports go at the top
from microbit import *

print("Hello")
```



```
Hello
MicroPython 5e619dd-dirty on 2023-04-06; micr
o:bit v2.1.1 with nRF52833
Type "help()" for more information.
>>> print('Welcome to the computer')
Welcome to the computer
>>> 
```

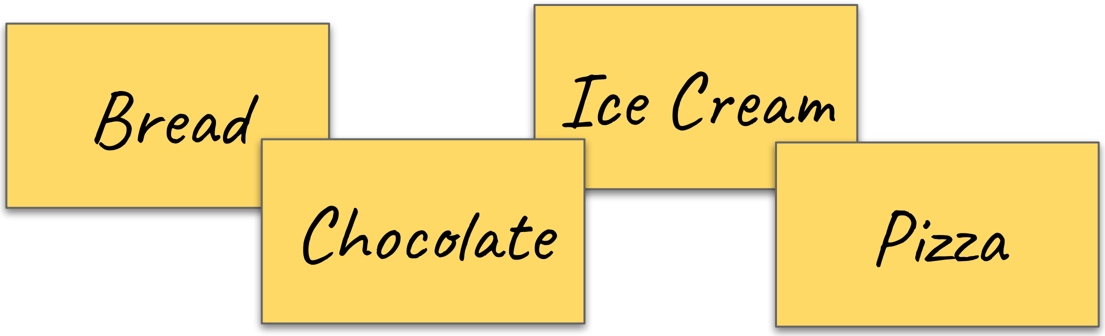
The code gets shown in in the serial using print()

We can type more code here too!

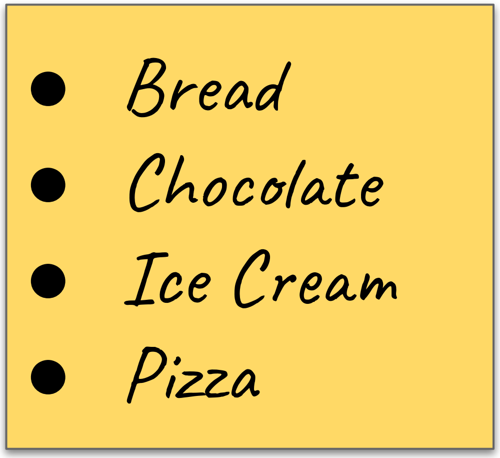
Storing lists of things

When we go shopping, we write down what we want to buy!

But we don't store it on lots of little pieces of paper!



We put it in one big shopping list!



Lists

It would be annoying to store it separately when we code too!

```
>>> shopping_item1 = "Bread"
>>> shopping_item2 = "Chocolate"
>>> shopping_item3 = "Ice Cream"
>>> shopping_item4 = "Pizza"
```

So much repetition!!

Instead we use a python list!

```
>>> shopping_list = ["Bread", "Chocolate", "Ice Cream",
"Pizza"]
```

You can put (almost) anything into a list

- You can have a list of integers

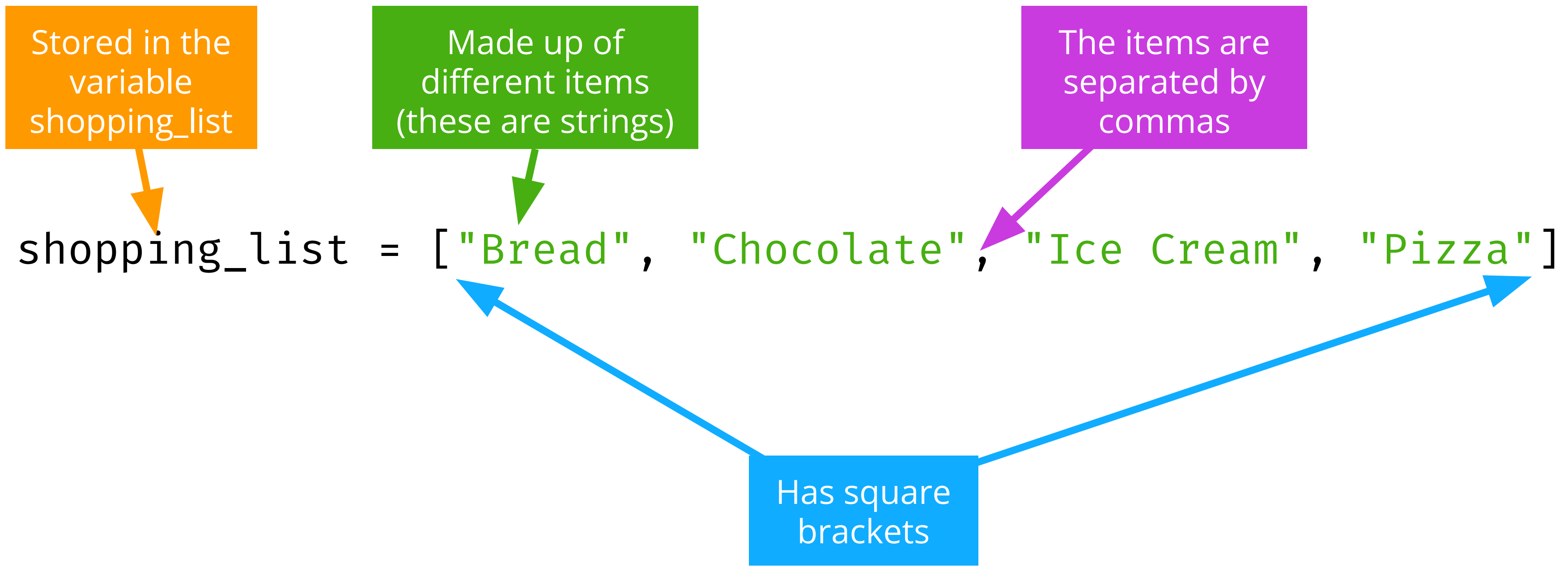
```
>>> primes = [1, 2, 3, 5, 11]
```

- You can have lists with mixed integers and strings

```
>>> mixture = [1, 'two', 3, 4, 'five']
```

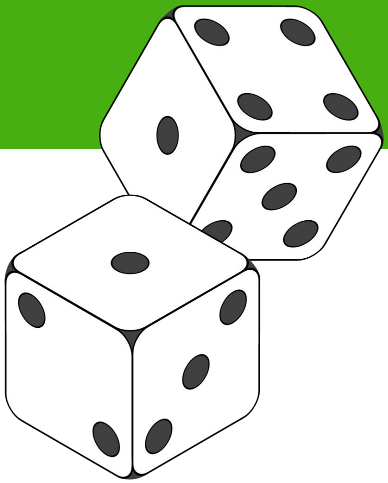
- But this is almost never a good idea! You should be able to treat every element of the list the same way.

List anatomy


Stored in the variable shopping_list
Made up of different items (these are strings)
The items are separated by commas
Has square brackets

```
shopping_list = ["Bread", "Chocolate", "Ice Cream", "Pizza"]
```

That’s so random!



There’s lots of things in life that are up to chance or random!

We want the computer to be random sometimes!

Python lets us **import** common bits of code people use! Today we’re going to use the **random** module!

Using the random module

Let's choose something randomly from a list!

This is like drawing something out of a hat in a raffle!

Try this!

1. Import the random module!
```
>>> import random
```

2. Copy the shopping list into the serial
```
>>> shopping_list = ["Bread", "Chocolate", "Ice Cream",
     "Pizza"]
```

3. Choose randomly! Try it a few times!
```
>>> random.choice(shopping_list)
```

Using the random module

You can also assign your random choice to a variable

```
>>> import random
>>> shopping_list = ["Bread", "Chocolate", "Ice Cream",
     "Pizza"]
>>> random_food = random.choice(shopping_list)
>>> print(random_food)
```

Project Time!

Raaaaaaaaaandom! Can you handle that?

Let’s try use it in our project!

Try to do Part 2

The tutors will be around to help!

If Statements

Conditions!

Conditions let us make decision.

First we test if the condition is met!

Then maybe we'll do the thing

<u>If it's raining</u> take an umbrella

Yep it's raining

…… take an umbrella

Booleans (True and False)

Computers store whether a condition is met in the form of

`True` and `False`

To figure out if something is `True` or `False` we do a comparison

What do you think these are?	
5 < 10	"Dog" == "dog"
3 + 2 == 5	"D" in "Dog"
5 != 5	"Q" not in "Cat"

Booleans (True and False)

Computers store whether a condition is met in the form of

`True` and `False`

To figure out if something is `True` or `False` we do a comparison

What do you think these are?

`True`	5 < 10	"Dog" == "dog"
	3 + 2 == 5	"D" in "Dog"
	5 != 5	"Q" not in "Cat"

Booleans (True and False)

Computers store whether a condition is met in the form of

`True` and `False`

To figure out if something is `True` or `False` we do a comparison

What do you think these are?

True	5 < 10	"Dog" == "dog"
True	3 + 2 == 5	"D" in "Dog"
	5 != 5	"Q" not in "Cat"

Booleans (True and False)

Computers store whether a condition is met in the form of

`True` and `False`

To figure out if something is `True` or `False` we do a comparison

What do you think these are?

`True` 5 < 10		"Dog" == "dog"
`True` 3 + 2 == 5		"D" in "Dog"
`False` 5 != 5		"Q" not in "Cat"

Booleans (True and False)

Computers store whether a condition is met in the form of

`True` and `False`

To figure out if something is `True` or `False` we do a comparison

What do you think these are?

True	5 < 10	**False**	"Dog" == "dog"
True	3 + 2 == 5		"D" in "Dog"
False	5 != 5		"Q" not in "Cat"

Booleans (True and False)

Computers store whether a condition is met in the form of

`True` and `False`

To figure out if something is `True` or `False` we do a comparison

What do you think these are?

True	5 < 10	**False**	"Dog" == "dog"
True	3 + 2 == 5	**True**	"D" in "Dog"
False	5 != 5		"Q" not in "Cat"

Booleans (True and False)

Computers store whether a condition is met in the form of

`True` and `False`

To figure out if something is `True` or `False` we do a comparison

What do you think these are?

True	5 < 10	**False**	"Dog" == "dog"
True	3 + 2 == 5	**True**	"D" in "Dog"
False	5 != 5	**True**	"Q" not in "Cat"

Booleans (True and False)

Python has some special comparisons for checking if something is **`in`** something else. **What do you think of these?**

```
"A" in "AEIOU"
"Z" in "AEIOU"
"a" in "AEIOU"
```

```
animals = ["cat", "dog", "goat"]
"banana" in animals
"cat" in animals
```

Booleans (True and False)

Python has some special comparisons for checking if something is **`in`** something else. **Try these!**

```
True   "A" in "AEIOU"
       "Z" in "AEIOU"
       "a" in "AEIOU"
```

```
animals = ["cat", "dog", "goat"]
"banana" in animals
"cat" in animals
```

Booleans (True and False)

Python has some special comparisons for checking if something is **`in`** something else. **Try these!**

```
True    "A" in "AEIOU"
False   "Z" in "AEIOU"
        "a" in "AEIOU"
```

```
animals = ["cat", "dog", "goat"]
"banana" in animals
"cat" in animals
```

Booleans (True and False)

Python has some special comparisons for checking if something is **`in`** something else. **Try these!**

```
True    "A" in "AEIOU"
False   "Z" in "AEIOU"
False   "a" in "AEIOU"
```

```
animals = ["cat", "dog", "goat"]
"banana" in animals
"cat" in animals
```

Booleans (True and False)

Python has some special comparisons for checking if something is **`in`** something else. **Try these!**

```
True    "A" in "AEIOU"
False   "Z" in "AEIOU"
False   "a" in "AEIOU"
```

```
animals = ["cat", "dog", "goat"]
"banana" in animals
"cat" in animals
                                  False
```

Booleans (True and False)

Python has some special comparisons for checking if something is **`in`** something else. **Try these!**

```
"A" in "AEIOU"
"Z" in "AEIOU"
"a" in "AEIOU"
```

```
True
False
False
```

```
animals = ["cat", "dog", "goat"]
"banana" in animals
"cat" in animals
```

```
False
True
```

Conditions

So to know whether to do something, they find out if it's `True`!

```
fave_num = 5
if fave_num < 10:
    print("that’s a small number")
```

Conditions

So to know whether to do something, they find out if it's True!

```
fave_num = 5
if fave_num < 10:
    print("that's a small number")
```

That's the condition!

Conditions

So to know whether to do something, they find out if it's True!

```
fave_num = 5
if fave_num < 10:
    print("that's a small number")
```

That's the condition!

Is it True that fave_num is less than 10?

- Well, fave_num is 5
- And it's True that 5 is less than 10
- So it is True!

Conditions

So to know whether to do something, they find out if it's `True`!

```
fave_num = 5
if True                :
    print("that's a small number")
```

Put in the answer to the question

Is it True that fave_num is less than 10?

- Well, fave_num is 5
- And it's True that 5 is less than 10
- So it is True!

Conditions

So to know whether to do something, they find out if it's `True`!

```
fave_num = 5
if True           :
    print("that's a small number")
```

What do you think happens?

```
>>>
```

Conditions

So to know whether to do something, they find out if it's True!

```
fave_num = 5
if True          :
    print("that's a small number")
```

<u>What do you think happens?</u>

```
>>> that's a small number
```

Conditions

How about a different number???

```
fave_num = 9000
if fave_num < 10:
    print("that’s a small number")
```

Conditions

Find out if it's `True`!

```
fave_num = 9000
if False    :
    print("that's a small number")
```

Put in the answer to the question

Is it `True` that fave_num is less than 10?

- Well, fave_num is 9000
- And it's not `True` that 9000 is less than 10
- So it is `False`!

Conditions

How about a different number???

```
fave_num = 9000
if fave_num < 10:
    print("that’s a small number")
```

What do you think happens?

```
>>>
```

Conditions

How about a different number???

```
fave_num = 9000
if fave_num < 10:
    print("that's a small number")
```

What do you think happens?

```
>>>
```

Nothing!

If statements

```
fave_num = 5
if fave_num < 10:
    print("that's a small number")
```

This line …

… controls this line

If statements

Actually .....

```
fave_num = 5
if fave_num < 10:
    print("that’s a small number")
    print("and I like that")
    print("A LOT!!")
```

This line ...

... controls anything below it that is indented like this!

If statements

```
fave_num = 5
if fave_num < 10:
    print("that’s a small number")
    print("and I like that")
    print("A LOT!!")
```

<u>What do you think happens?</u>

```
>>>
```

If statements

What do you think happens?

```
fave_num = 5
if fave_num < 10:
    print("that’s a small number")
    print("and I like that")
    print("A LOT!!")
```

```
>>> that’s a small number
>>> and I like that
>>> A LOT!!
```

If statements

```
word = "GPN"
if word == "GPN":
  print("GPN is awesome!")
```

What happens?

If statements

```
word = "GPN"
if word == "GPN":
  print("GPN is awesome!")
```

What happens?

```
>>> GPN is awesome!
```

Else statements

else statements means something still happens if the if statement was False

```
word = "Chocolate"
if word == "GPN":
  print("GPN is awesome!")
else:
  print("The word isn't GPN :(")
```

What happens?

Else statements

else statements means something still happens if the if statement was False

```
word = "Chocolate"
if word == "GPN":
  print("GPN is awesome!")
else:
  print("The word isn’t GPN :(")
```

What happens?

```
>>> The word isn’t GPN :(
```

Elif statements

elif
Means we can give specific instructions for other words

```
word = "Chocolate"
if word == "GPN":
  print("GPN is awesome!")
elif word == "Chocolate":
  print("YUMMM Chocolate!")
else:
  print("The word isn’t GPN :(")
```

What happens?

Elif statements

elif
Means we can give specific instructions for other words

```
word = "Chocolate"
if word == "GPN":
  print("GPN is awesome!")
elif word == "Chocolate":
  print("YUMMM Chocolate!")
else:
  print("The word isn’t GPN :(")
```

What happens?

```
>>> YUMMM Chocolate!
```

Project Time!

You now know all about `if` and `else`!

See `if` you can do Part 3

The tutors will be around to help!

While Loops

Loops

We know how to do things on repeat!

Sometimes we want to do some code on repeat!

Introducing ... `while` loops!

What do you think this does?

```
i = 0
while i < 3:
    print("i is " + str(i))
    i = i + 1
```

Introducing ... `while` loops!

What do you think this does?

```
i = 0
while i < 3:
    print("i is " + str(i))
    i = i + 1
```

```
i is 0
i is 1
i is 2
>>>
```

Introducing … `while` loops!

Stepping through a while loop…

Introducing … `while` loops!

One step at a time!

```
i = 0
while i < 3:
    print("i is " + str(i))
    i = i + 1
```

MY VARIABLES

```
i = 0
```

Set the variable

Introducing ... while loops!

One step at a time!

0 is less than 3 !

```
i = 0
while i < 3:
    print("i is " + str(i))
    i = i + 1
```

MY VARIABLES

```
i = 0
```

Introducing ... while loops!

One step at a time!

```
i = 0
while i < 3:
    print("i is " + str(i))
    i = i + 1
```

```
i is 0
```

MY VARIABLES

```
i = 0
```

Introducing … while loops!

One step at a time!

```
i = 0
while i < 3:
    print("i is " + str(i))
  ◆ i = i + 1
```

MY VARIABLES

~~i = 0~~
i = 1

UPDATE TIME !

```
i is 0
```

Introducing … `while` loops!

One step at a time!

Take it from the top!

```
i = 0
while i < 3:
    print("i is " + str(i))
    i = i + 1
```

```
i is 0
```

MY VARIABLES

~~i = 0~~
i = 1

Introducing ... while loops!

One step at a time!

```
i = 0
while i < 3:
    print("i is " + str(i))
    i = i + 1
```

MY VARIABLES

~~i = 0~~

i = 1

```
i is 0
```

Introducing ... while loops!

One step at a time!

Print !

```
i = 0
while i < 3:
    print("i is " + str(i))
    i = i + 1
```

```
i is 0

i is 1
```

MY VARIABLES

~~i = 0~~
i = 1

Introducing … `while` loops!

One step at a time!

```
i = 0
while i < 3:
    print("i is " + str(i))
◆   i = i + 1
```

MY VARIABLES

~~i = 0~~

~~i = 1~~

i = 2

UPDATE TIME !

```
i is 0

i is 1
```

Introducing ... `while` loops!

One step at a time!

Take it from the top!

```
i = 0
while i < 3:
    print("i is " + str(i))
    i = i + 1
```

MY VARIABLES

~~i = 0~~

~~i = 1~~

i = 2

```
i is 0

i is 1
```

Introducing ... `while` loops!

One step at a time!

2 is less than 3 !

```
i = 0
while i < 3:
    print("i is " + str(i))
    i = i + 1
```

MY VARIABLES

~~i = 0~~

~~i = 1~~

i = 2

```
i is 0

i is 1
```

Introducing ... `while` loops!

One step at a time!

Print !

```
i = 0
while i < 3:
    print("i is " + str(i))
    i = i + 1
```

```
i is 0

i is 1

i is 2
```

MY VARIABLES

~~i = 0~~

~~i = 1~~

i = 2

Introducing … while loops!

One step at a time!

```
i = 0
while i < 3:
    print("i is " + str(i))
  ◆ i = i + 1
```

MY VARIABLES

~~i = 0~~
~~i = 1~~
~~i = 2~~
i = 3

UPDATE TIME !

```
i is 0

i is 1

i is 2
```

Introducing … while loops!

One step at a time!

Take it from the top !

```
i = 0
while i < 3:
    print("i is " + str(i))
    i = i + 1
```

```
i is 0

i is 1

i is 2
```

MY VARIABLES

~~i = 0~~

~~i = 1~~

~~i = 2~~

i = 3

Introducing … `while` loops!

One step at a time!

3 IS NOT less than 3 !

```
i = 0
while i < 3:
    print("i is " + str(i))
    i = i + 1
```

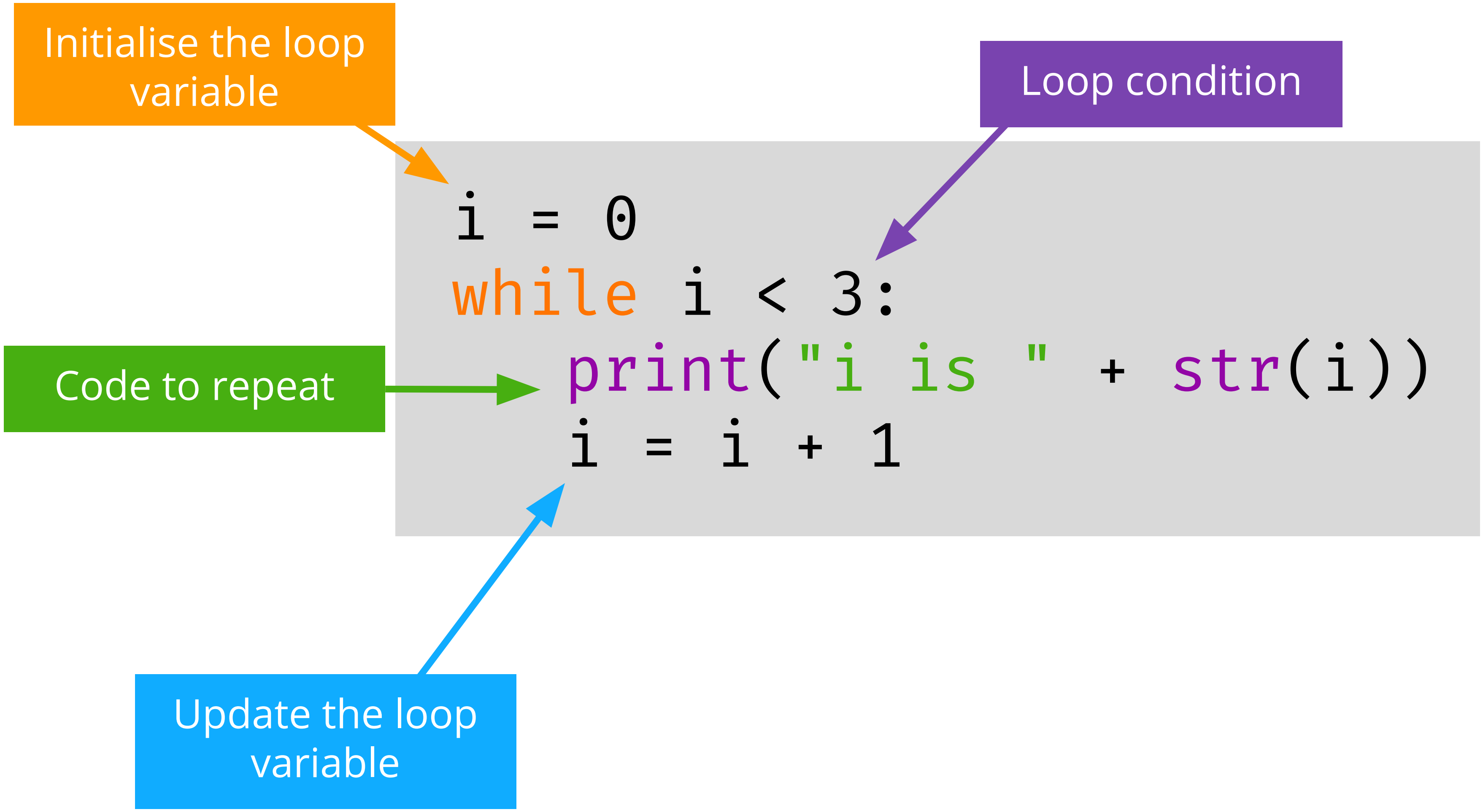
MY VARIABLES

~~i = 0~~
~~i = 1~~
~~i = 2~~
i = 3

We are are done with this loop!

```
i is 0
i is 1
i is 2
```

Introducing ... `while` loops!



```
i = 0
while i < 3:
    print("i is " + str(i))
    i = i + 1
```

What happens when.....

What happens if we forget to update the loop variable?

```
i = 0
while i < 3:
    print("i is " + str(i))
```

What happens when.....

What happens if we forget to update the loop variable?

```python
i = 0
while i < 3:
    print("i is " + str(i))
```

```
i is 0
i is 0
i is 0
i is 0
i is 0
i is 0
i is 0
i is 0
i is 0
i is 0
i is 0
i is 0
i is 0
```

Infinite loop!

Sometimes we want our loop to go forever!

So we set a condition that is always True!

We can even just write True!

```
while True:
    print("Are we there yet?")
```

Give me a break!

But what if I wanna get out of a loop early?

That's when we use the break keyword!

```
number = 0
while number != 42 :
    number = input("Guess a number: ")

    if number = "I give up":
        print("The number was 42")
        break

    number = int(number)
```

Continuing on

How about if I wanna skip the rest of the loop body and loop again? We use continue for that!

```
number = 0
while number != 42 :
    number = input("Guess a number: ")

    if not number.isnumeric():
        print("That’s not a number!")
        print("Try again")
        continue

    number = int(number)
```

Running Time

Sometimes you want to time things. Like, for example, if you wanted to put a time limit on a game and see how many points you can get in 30 seconds!

To figure out how long the Micro:Bit program has been running (in milliseconds) you can use this command:

```
time = running_time()
```

What would running_time() be after 4 seconds?

What about after **10 and a half** second?

Running Time

Sometimes you want to time things. Like, for example, if you wanted to put a time limit on a game and see how many points you can get in 30 seconds!

To figure out how long the Micro:Bit program has been running (in milliseconds) you can use this command:

```
time = running_time()
```

What would running_time() be after 4 seconds?

4000

What about after **10 and a half** second?

Running Time

Sometimes you want to time things. Like, for example, if you wanted to put a time limit on a game and see how many points you can get in 30 seconds!

To figure out how long the Micro:Bit program has been running (in milliseconds) you can use this command:

```
time = running_time()
```

What would running_time() be after 4 seconds?

4000

What about after **10 and a half** second?

10,500

Project Time!

while we're here:

Try to do Part 4!

The tutors will be around to help!

Micro:Bit Buttons

Buttons!

Your MicroBit has 2 buttons: Button A and Button B

You can use this code to check whether or not a button is pressed:

```
button_a.is_pressed()

button_b.is_pressed()
```

The statement will be **TRUE** if the button is being pressed at that time and it will be **FALSE** if it is *not* being pressed

Buttons!

What do you think this code does?

```
if button_a.is_pressed():
   display.show(Image.HAPPY)

if button_b.is_pressed():
   display.show(Image.SAD)
```

If ***button a*** is pressed when the Micro:Bit gets to this line of code then what happens?

If ***button b*** is pressed when the Micro:Bit gets to this line of code then what happens

What do you think happens if *both* button a AND button b are being pressed?

Buttons!

What do you think this code does?

```
if button_a.is_pressed():
   display.show(Image.HAPPY)

if button_b.is_pressed():
   display.show(Image.SAD)
```

If ***button a*** is pressed when the Micro:Bit gets to this line of code then what happens?

The Micro:Bit shows a Happy face

If ***button b*** is pressed when the Micro:Bit gets to this line of code then what happens

What do you think happens if *both* button a AND button b are being pressed?

Buttons!

What do you think this code does?

```
if button_a.is_pressed():
    display.show(Image.HAPPY)

if button_b.is_pressed():
    display.show(Image.SAD)
```

If ***button a*** is pressed when the Micro:Bit gets to this line of code then what happens?

The Micro:Bit shows a Happy face

If ***button b*** is pressed when the Micro:Bit gets to this line of code then what happens

The Micro:Bit shows a Sad face

What do you think happens if *both* button a AND button b are being pressed?

Project Time!

Does that press your buttons?

Try to do Part 5 and 6!

The tutors will be around to help!

Micro:Bit Radio

Radio

Your Micro:Bit can send messages to other Micro:Bits using radio waves!

It only takes a few lines of code to make this work!

1. We have to tell the Micro:Bit that we want to use the radio:

```
import radio
```

2. We need to turn the Radio on:

```
radio.on()
```

3. We need to send a message:

```
radio.send(“Hello World”)
```

4. We want to receive a message:

```
message = radio.receive()
```

Radio Groups

We need to set our radio to communicate on a certain group, otherwise all our Micro:Bits will try to talk to each other! This will get confusing for the Micro:Bit.

After you turn the radio on, set the group channel!

```
radio.config(group=100)
```

Your tutors will give you a group number to use.

Radio Example

What :do you think this code does?

Micro:Bit 1

```
import radio

radio.on()
radio.config(group=100)

while True:
    if button_a.is_pressed():
        radio.send(“Hello!”)

    if button_b.is_pressed():
        radio.send(“World!”)
```

Micro:Bit 2

```
import radio

radio.on()
radio.config(group=100)

while True:
    message = radio.receive()
    if message:
        display.scroll(message)
```

Why do you think it’s important to check the message?

Tell us what you think!

Click on the

End of Day Form

and fill it in now!