

Welcome to the Labs

Bop It! - Micro:Bit

Thank you to our Sponsors!

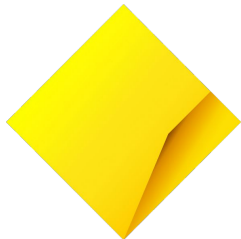
Platinum Sponsors:



Australian Government

Australian Signals Directorate

Gold Sponsor:



**Commonwealth
Bank**

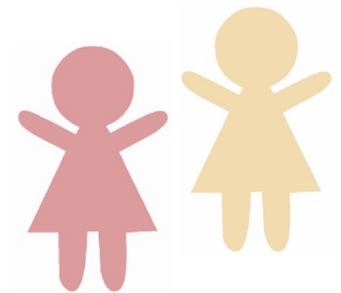
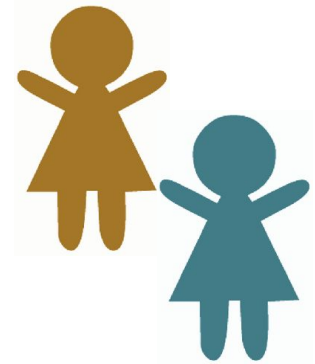


Who are the tutors?

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

Click on your node location

Click on your room.

From this page you can see:

- These **slides** (to take a look back or go on ahead).
- A digital copy of your **workbook**.
- Other helpful bits to use through the day!

Tell us you're here!

Click on the
Start of Day Survey
and fill it in now!



Start of Day Survey

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!

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 6.2: 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

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.

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.



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



★ BONUS 4.3: Do some extra!

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

Today's project!

Bop It! - Micro:Bit

Bop-It

Bop-It is a popular game where the game issues commands like

- PUSH THE BUTTON
- FLICK THE SWITCH
- PULL THE LEVER

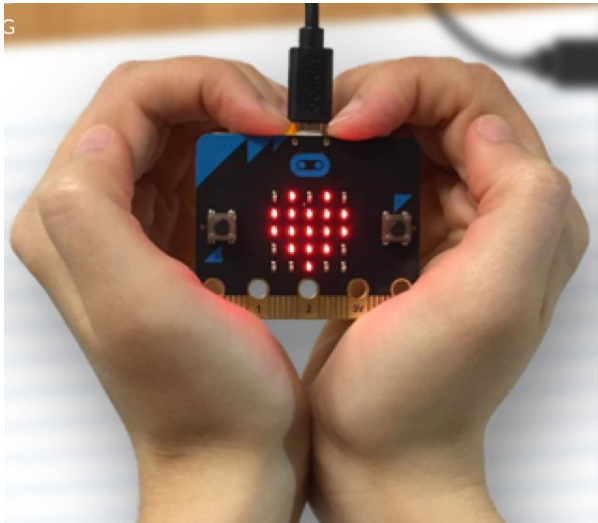
and the player has to do them as quickly as possible.

Today you're going to make your own Bop-It game using a Micro:Bit.



Micro:Bits - IRL

Sadly 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/>

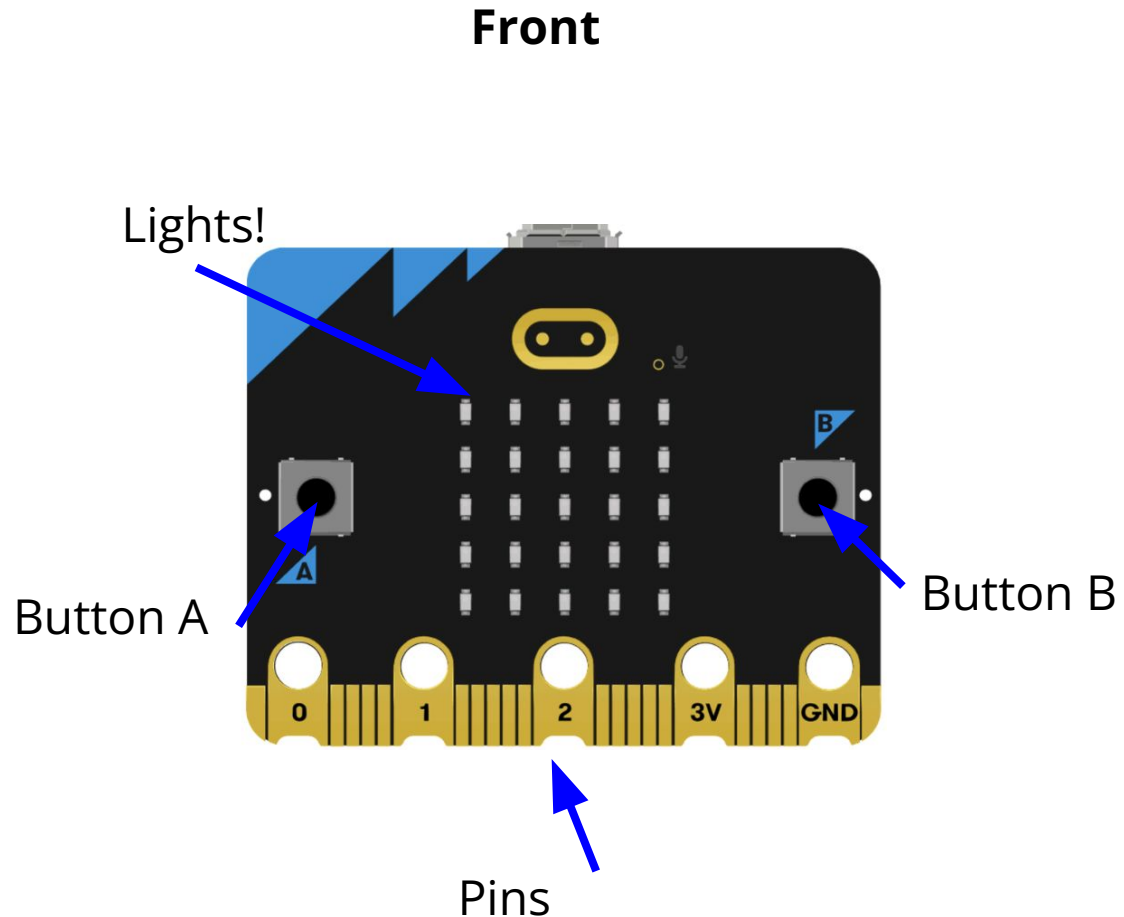
Intro to Micro:Bit

What is a Micro:Bit?

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

Lights: We can turn each light on or off to make different images

Pins: These let us connect the Micro:Bit to other devices using wires



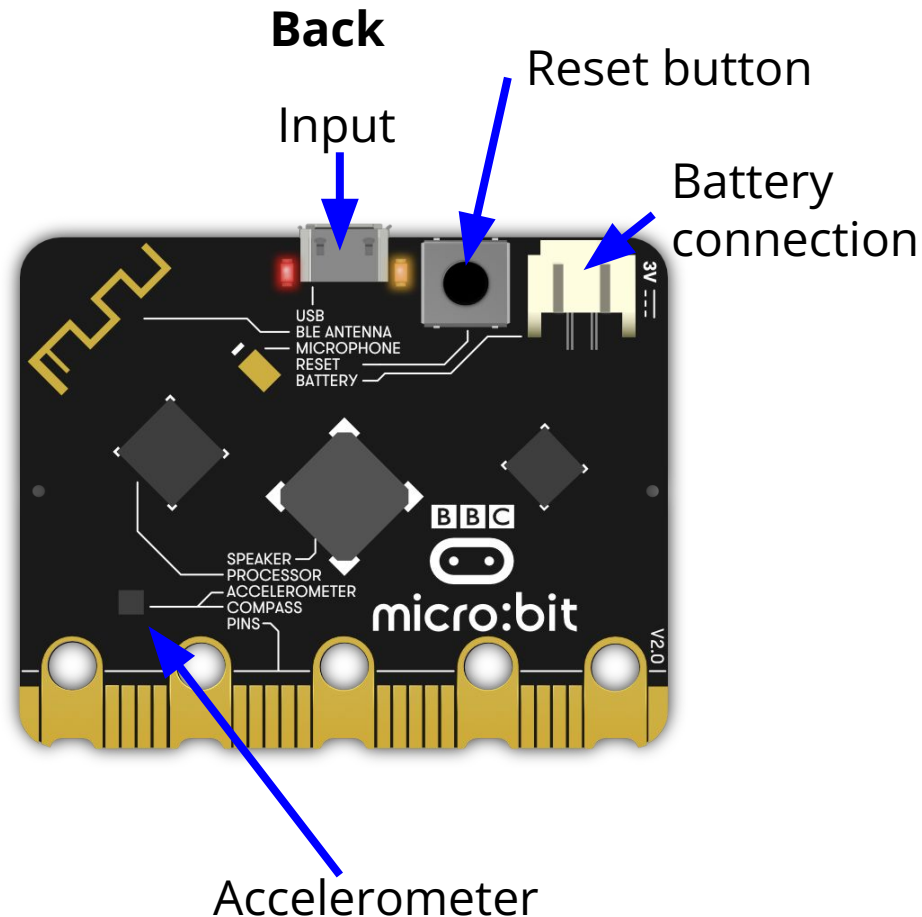
What is a Micro:Bit?

Input: Where we connect the cable from the computer to transfer our code/power to our Micro:Bit

Reset button: Let's you stop your code and starts it again

Battery connection: You can use your micro:bit even when it is not plugged into your computer! Ask your tutor for a battery pack if you need one.

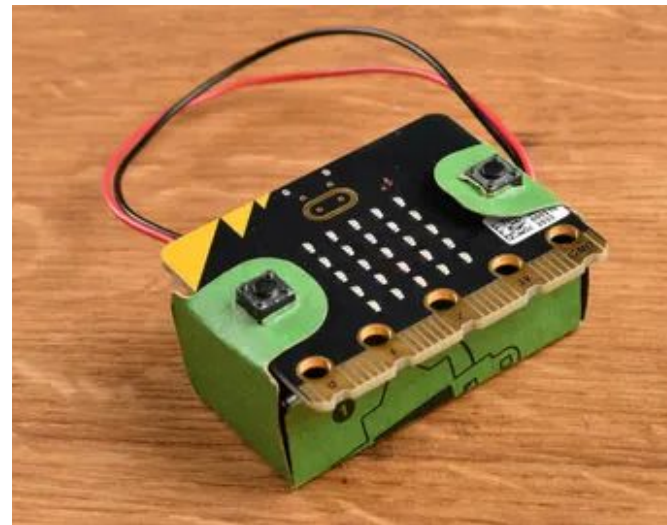
Accelerometer: The Micro:bit can tell us when it is **accelerated** - so it knows when we shake it!



Battery pack!

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

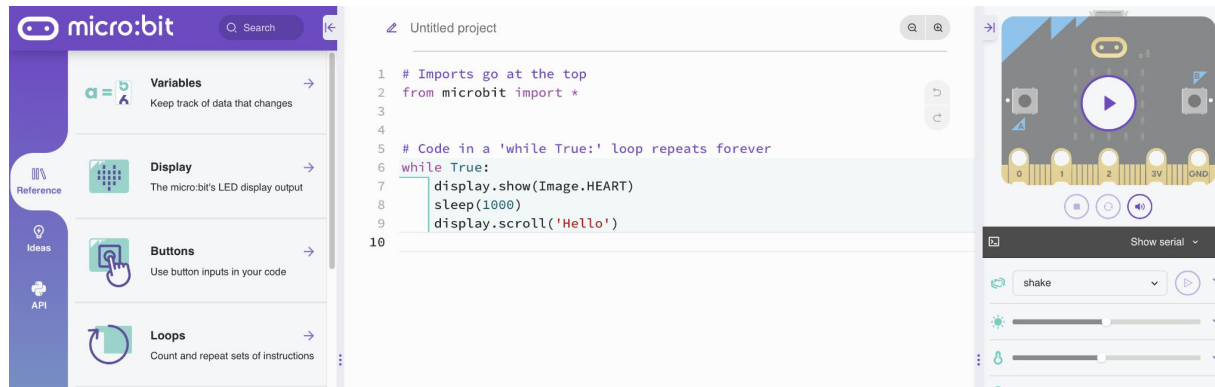
You can ask your tutor for a battery pack if you need one.



Using python.microbit.org

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

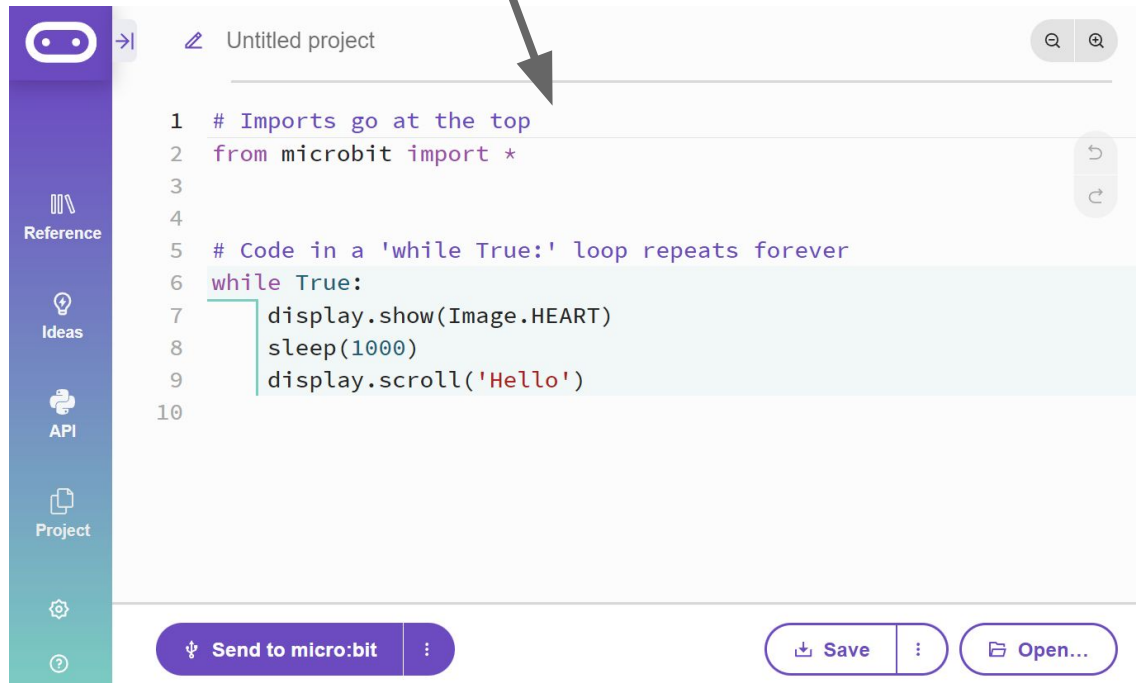
Go to python.microbit.org



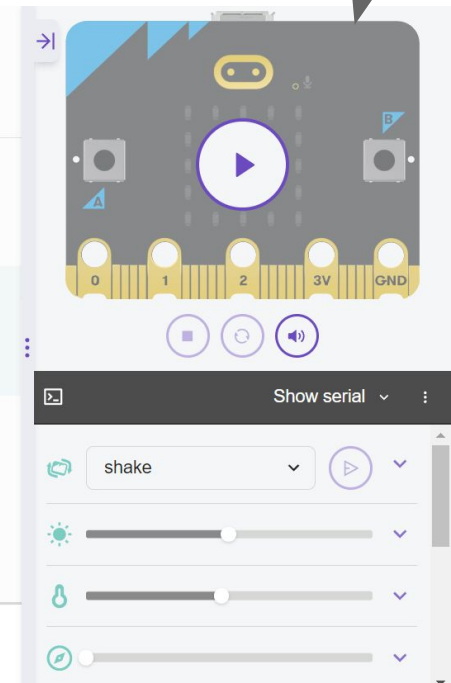
You should see this page pop up!

python.microbit.org

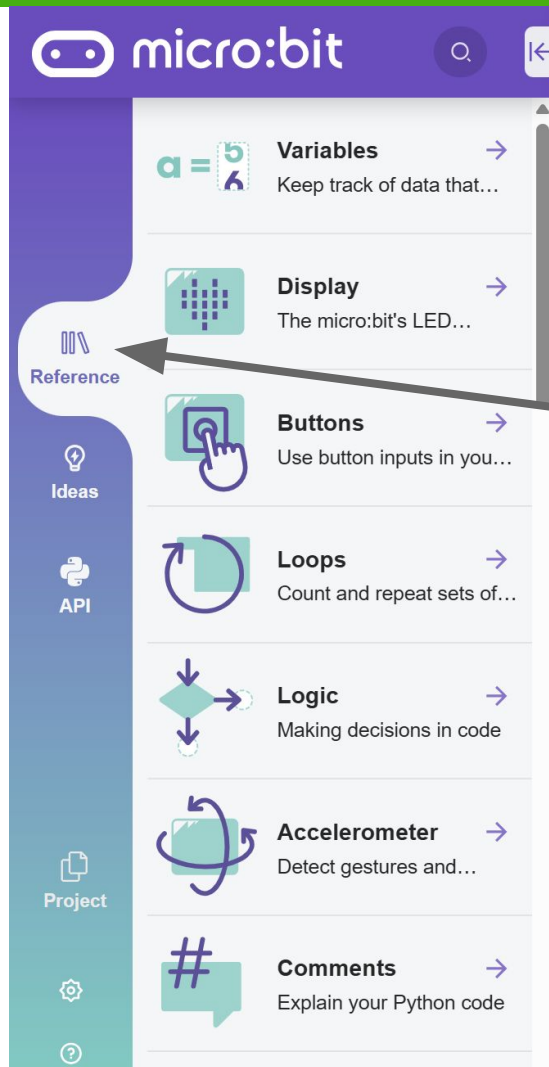
This is where we code



This is the simulator where we test our code



python.microbit.org



This is the reference. Click on the reference button to help you find the syntax for different instructions.

Click here

How do we write code for it?

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

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

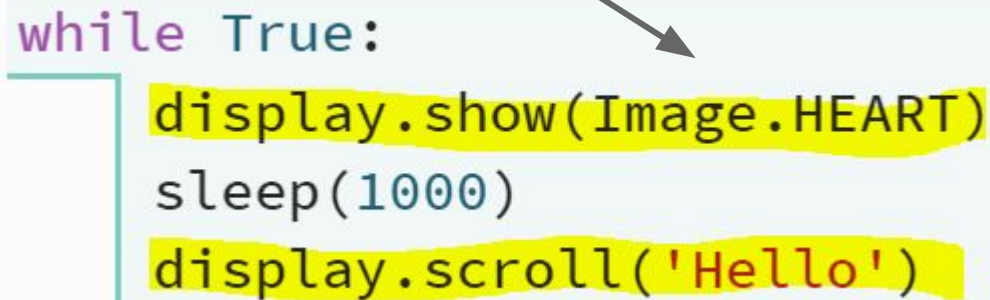
```
from microbit import *
```

This lets us use lights, sounds, buttons and lots of other cool things in our Python code for the Micro:Bit

The Display

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

Show an image, like a heart!



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

The code is shown in a light blue box. The first line is 'while True:'. The next three lines are indented and highlighted in yellow: 'display.show(Image.HEART)', 'sleep(1000)', and 'display.scroll('Hello')'. An arrow points from the text 'Show an image, like a heart!' to the 'Image.HEART' part of the code. Another arrow points from the text 'Scroll a word across the display, like 'Hello'' to the 'Hello' part of the code.

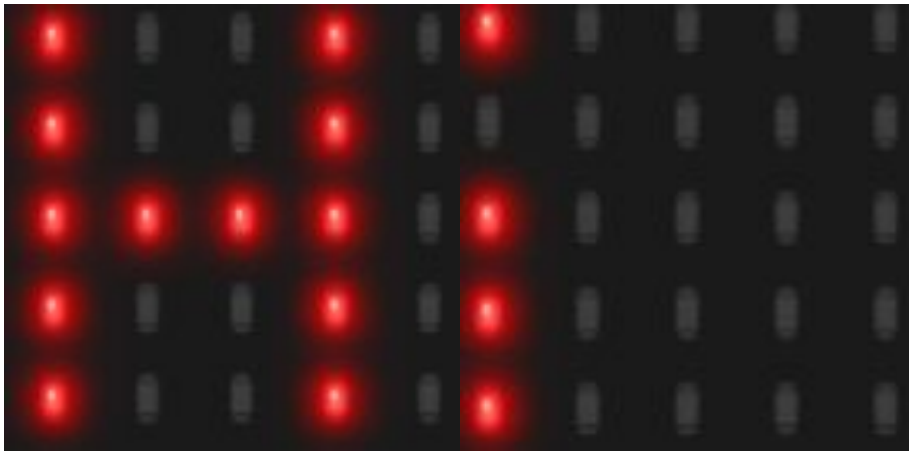
Scroll a word across the display, like 'Hello'

This code is in your **python.microbit.org** coding space - have a look
It's indented in a while loop - so it will repeat forever

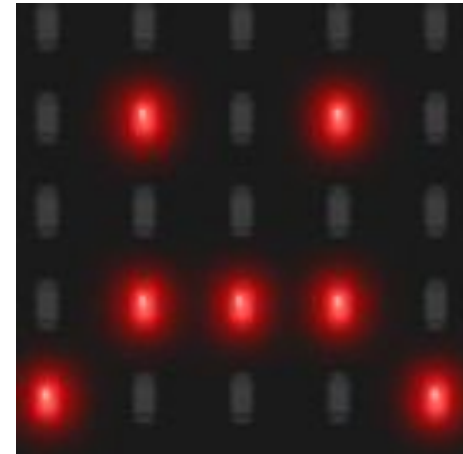
To show or to scroll....

Remember our microbit has a 5x5 grid of lights.

```
display.scroll('Hi!')
```



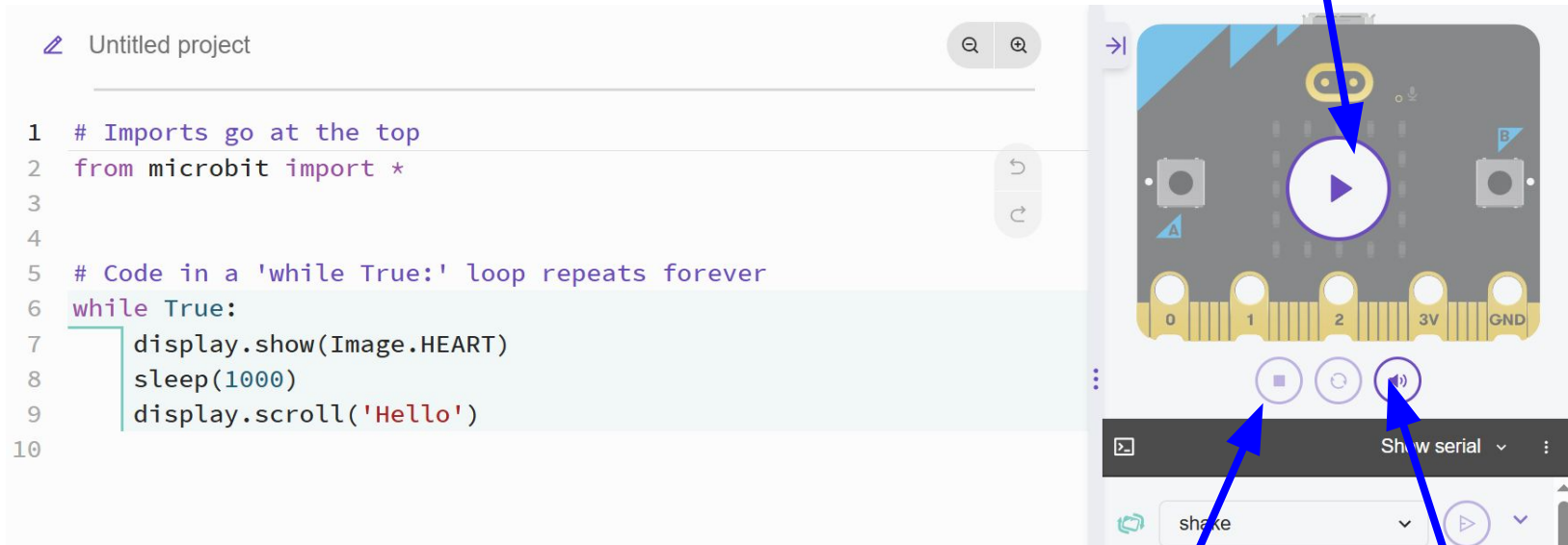
```
display.show(Image.SAD)
```



So if you want to display long words or messages, use scroll. If you want to display something that fits in a 5x5 grid, use show!

Using the Simulator

- **Click the arrow on the Simulator to run the code**
- A heart is displayed for 1 second and then 'Hello'



We can run our code on the Simulator or the real micro:bit!

Stop, Restart, Simulator settings are underneath

Connect the Micro:Bit

- Tutors will hand out the micro:bits & cables
- Connect the small end of the cable to the top of micro:bit
- Connect the other end to computer USB port
- New micro:bits will play a “Meet the Microbit” program for you to follow:
 - Push the buttons
 - Shake
 - Tilt to catch flashing LED
 - Clap a few times
- The tutors will help you

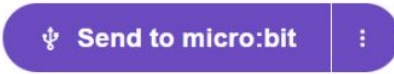


Run the code on the Micro:Bit (Chrome/Edge)

It's fun to mess around with the Micro:Bit on the simulator.
Now let's see your code on a Micro:Bit in real life!



Run your code on your Micro:Bit like this

1. Make sure your Micro:Bit is plugged into your computer
2. Click  bottom left
3. Follow the prompts
4. Choose your micro:bit and click CONNECT
5. **Wait for the red light** on the back of your micro:bit to stop flashing
6. Your code should be running on the micro:bit!

You should see a HEART displayed for 1 second and then HELLO (repeating)
Want your code to start again? Press black **“reset”** button on the back

Run the code on the Micro:Bit (other browser)

This is for if you don't have the Chrome or Edge browser (eg Safari)

Run your code on your Micro:Bit like this

1. Make sure your Micro:Bit is plugged into your computer
2. Click  bottom left
3. Click Close when you get a popup
4. Name your project and click Confirm and Save
5. Follow the instructions on the popup (drag the file from your downloads folder to the MICROBIT device)
6. **Wait for the red light** on the back of your micro:bit to stop flashing
7. Your code should be running on the micro:bit!

You should see a HEART displayed for 1 second and then HELLO (repeating)
Want your code to start again? Press black **“reset”** button on the back

Comments

- We use **comments** to write things in our code for humans!
- The computer ignores comments
- Comments start with a **#**

This code was written by Alex

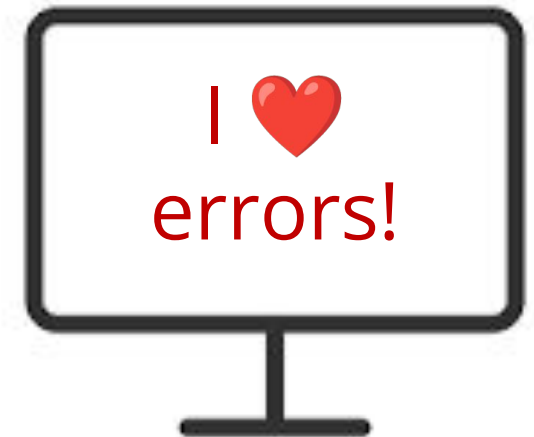
- Programmers use comments to explain what their code does
- You can 'comment out' code to stop it from running

Have a look at the code in the coding space - can you see the purple comments lines starting with the #

```
# Code in a 'while True:' loop repeats forever
```

Mistakes are Great! Errors on the Micro:bit!

- Programmers make A LOT of errors!
- Error messages give us hints on how to fix the problem
- Mistakes don't break computers!
- Unexpected words scrolling across the micro:bit is an error message
- Run on the simulator to see it better



```
❏ ! line 19 NameError: name 'junge'
```



```
❏ ! line 20 IndentationError: unde
```

We can learn from our mistakes!



1. Where the error is

2. What went wrong

- In your code - red dot at the start of the line
- Put the cursor over than line of code to get a hint

```
6 while True:
7     display.show(Image.HEARTx)
```

Project Time!

Let's get started!

Try to do Part 0 - 1

You've already done the first task!
The tutors will be around to help!

Lists and Random!

Strings and Ints - Data Types!

Strings can have any characters in them, even just spaces!

Strings are surrounded by quotes (" or ')

```
"Hello, world!"      "bla bla bla"      "abcd1234"
```

```
":)"      "'I can use single quotes too!'"
```

Integers are whole numbers in python - no quotes (")

We can do maths with **integers** but not **strings**.

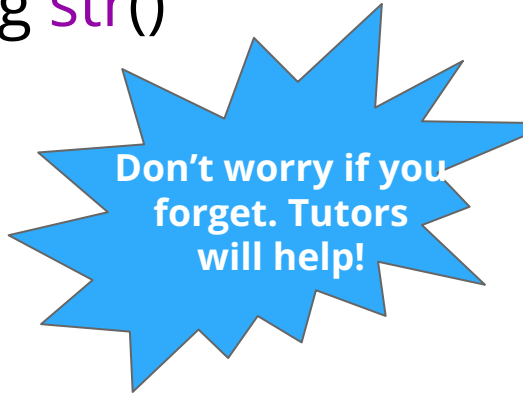
```
1      0      22      954865746
```

Strings and Ints!

Sometimes we need to turn a string into an integer and vice versa so we can use them as we want to in the code

- we can turn a **string** into an **integer** using **int()**
- we can turn an **integer** into a **string** using **str()**

You'll be doing this in your code!



Don't worry if you
forget. Tutors
will help!

Strings and Variable names

The difference between strings and variable names can be confusing! Let's take a look at an example.

```
animal = "dog"  
display.scroll(animal)  
display.scroll("animal")
```

What do you think is shown?

Strings and Variable names

The difference between strings and variable names can be confusing! Let's take a look at an example.

```
animal = "dog"
```

```
display.scroll(animal)
```

```
display.scroll("animal")
```

No quotes = uses the value of the variable!

Quotes = uses the word in the quotes!

What do you think is shown?

Strings and Variable names

The difference between strings and variable names can be confusing! Let's take a look at an example.

```
animal = "dog"
```

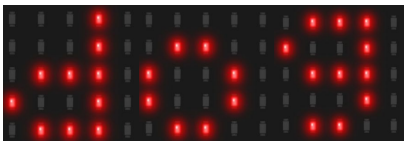
```
display.scroll(animal)
```

```
display.scroll("animal")
```

No quotes = uses the value of the variable!

Quotes = uses the word in the quotes!

What do you think is shown?



Strings and Variable names

The difference between strings and variable names can be confusing! Let's take a look at an example.

```
animal = "dog"
```

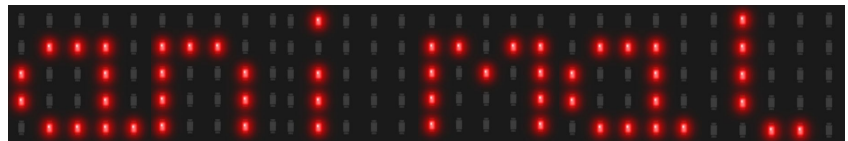
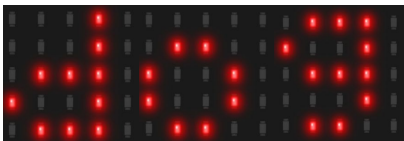
```
display.scroll(animal)
```

```
display.scroll("animal")
```

No quotes = uses the value of the variable!

Quotes = uses the word in the quotes!

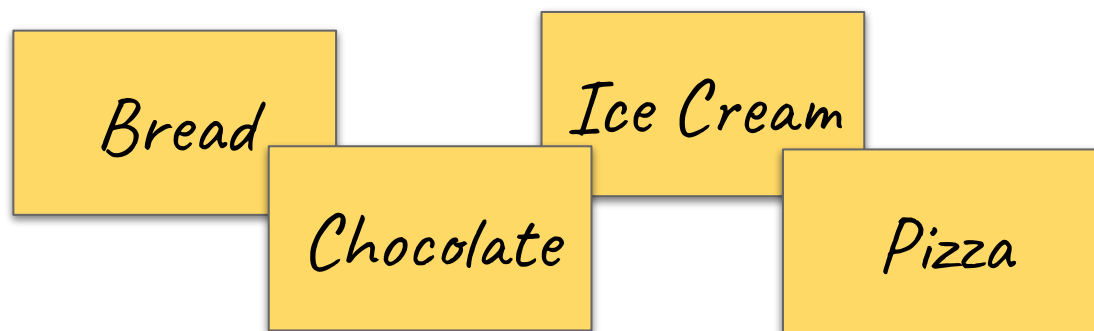
What do you think is shown?



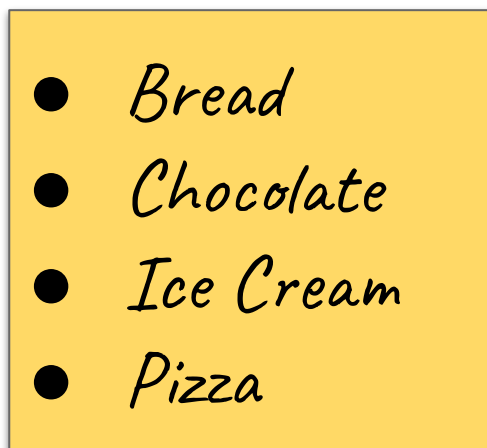
Lists

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

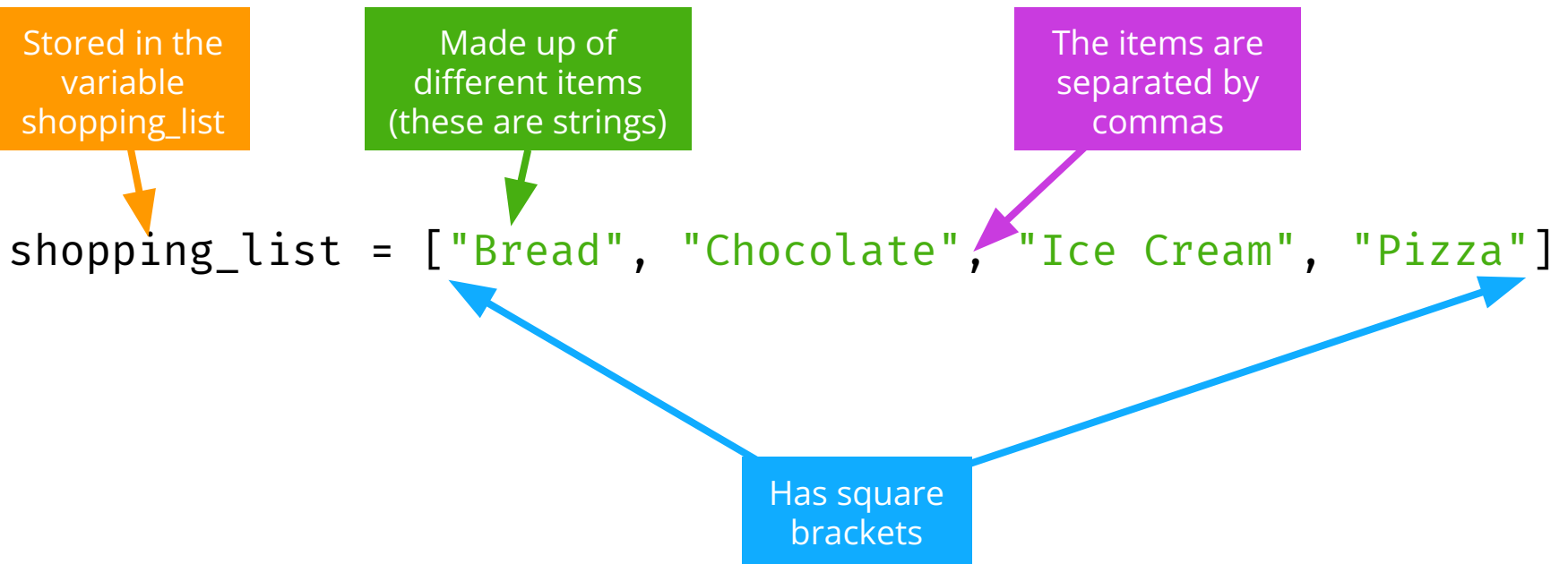
- You can have a list of **integers**

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

- You can have a list of **strings**

```
>>> friends = ['fred', 'mattie', 'adele']
```

List anatomy

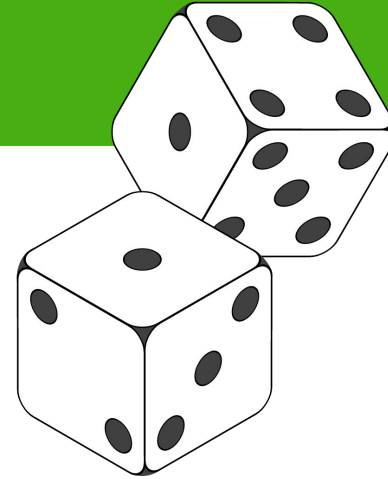


That's so random!

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



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



We want the computer to be random sometimes!



Using the random module

Let's choose something randomly from a list!

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

We can do it like this

1. Import the random module!

```
>>> import random
```

2. Make a list

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

3. Choose randomly from the list and store it in a variable!

```
>>> dinner = random.choice(shopping_list)
```

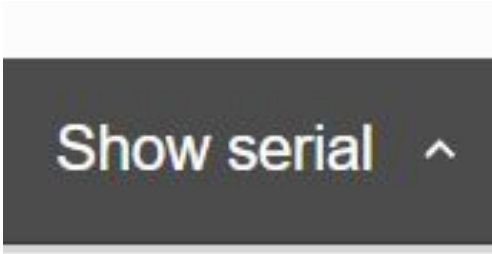


Printing bits

What if we wanted to display something from our program but not on the micro:bit?

We can use `print('put some text here')` to print words and sentences and symbols in the ***serial*** on the screen

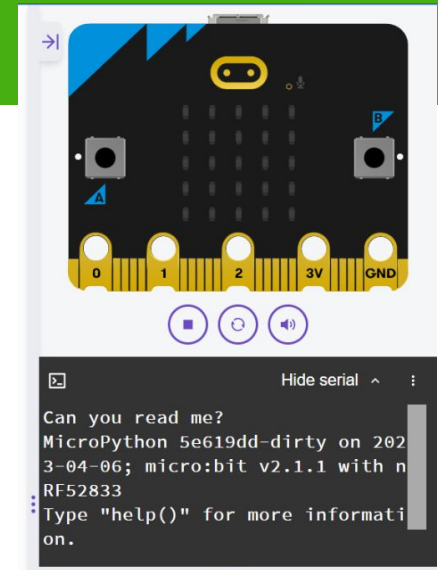
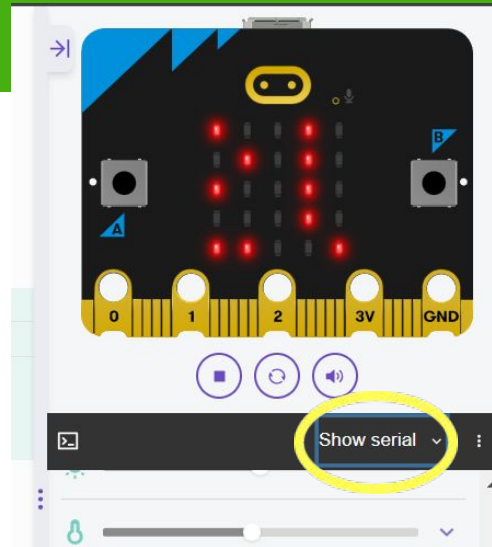
See the `print` by clicking **Show serial** on the screen



Show serial ^

Printing bits

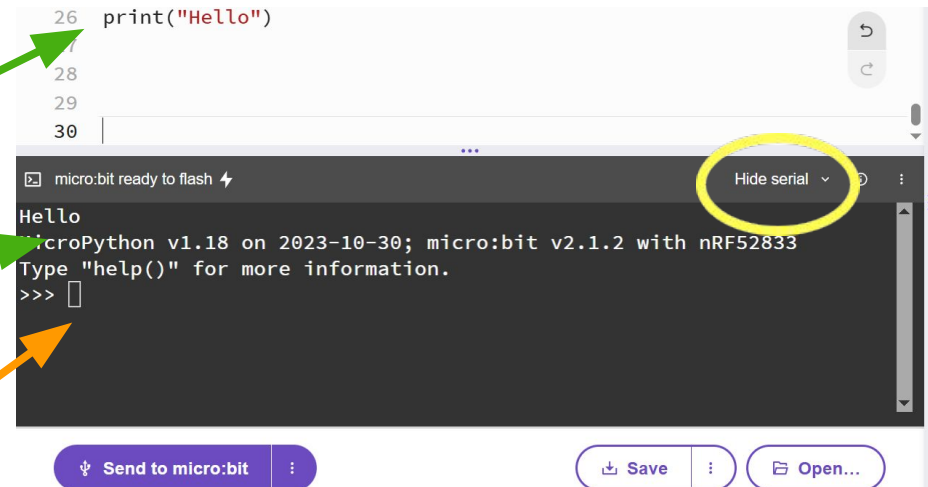
There's a **serial** for the simulator



... and a **serial** for when you are running your code on the micro:bit

The code gets shown in the serial using print()

We can type more code here too!



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



If it's raining 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`

`3 + 2 == 5`

`5 != 5`

`"Dog" == "dog"`

`"D" in "Dog"`

`"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 $5 < 10$

$3 + 2 == 5$

$5 != 5$

"Dog" == "dog"

"D" in "Dog"

"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 these will be?**

"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. **What do you think these will be?**

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. **What do you think these will be?**

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. **What do you think these will be?**

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. **What do you think these will be?**

True "A" in "AEIOU"

False "Z" in "AEIOU"

False "a" in "AEIOU"

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

"banana" in animals

False

"cat" in animals

Booleans (True and False)

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

True "A" in "AEIOU"

False "Z" in "AEIOU"

False "a" in "AEIOU"

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

"banana" in animals

False

"cat" in animals

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")
```

What do you think happens?

```
>>> 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!!")
```

What do you think happens?

```
>>>
```

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?

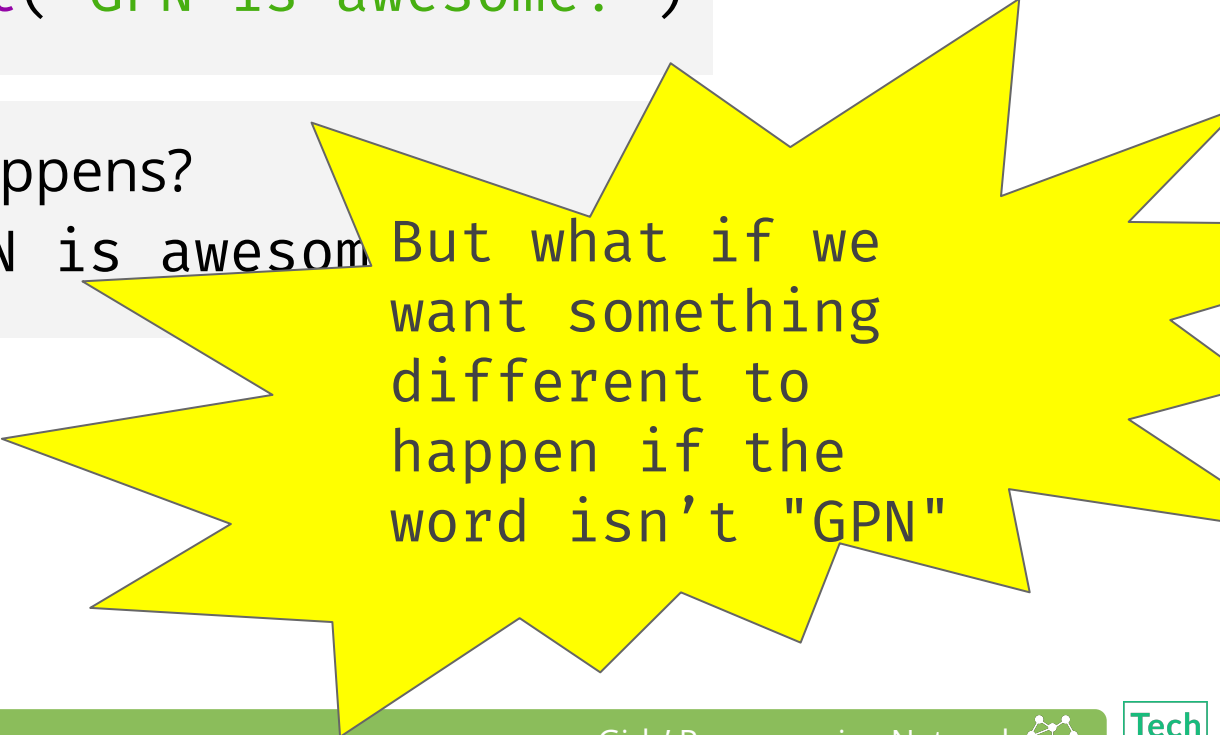
```
>>> GPN is awesome!
```

If statements

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

What happens?

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



But what if we
want something
different to
happen if the
word isn't "GPN"

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!
```

== AND :

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

Remember ...

==

When testing for equals in
your condition

:

At end of each if line to say you have
finished writing your condition

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!

Initialise the loop variable

Loop condition

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

Code to repeat

Update the loop variable

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!

Print!

```
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 is less
than 3!

```
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
```

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

MY VARIABLES

```
i = 0
i = 1
i = 2
```

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
```

We are
done
with this
loop!

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

MY VARIABLES

```
i = 0
i = 1
i = 2
i = 3
```

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?

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

[illegible]

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 Micro:Bit 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?

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?

The Micro:Bit shows a Happy face then a Sad face

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!