Welcome to the labs!

Tamagotchi! - Micro:bits

Thank you to our Sponsors!

Platinum Sponsor:

ATLASSIAN

amazon

Gold Sponsor:

Commonwealth Bank

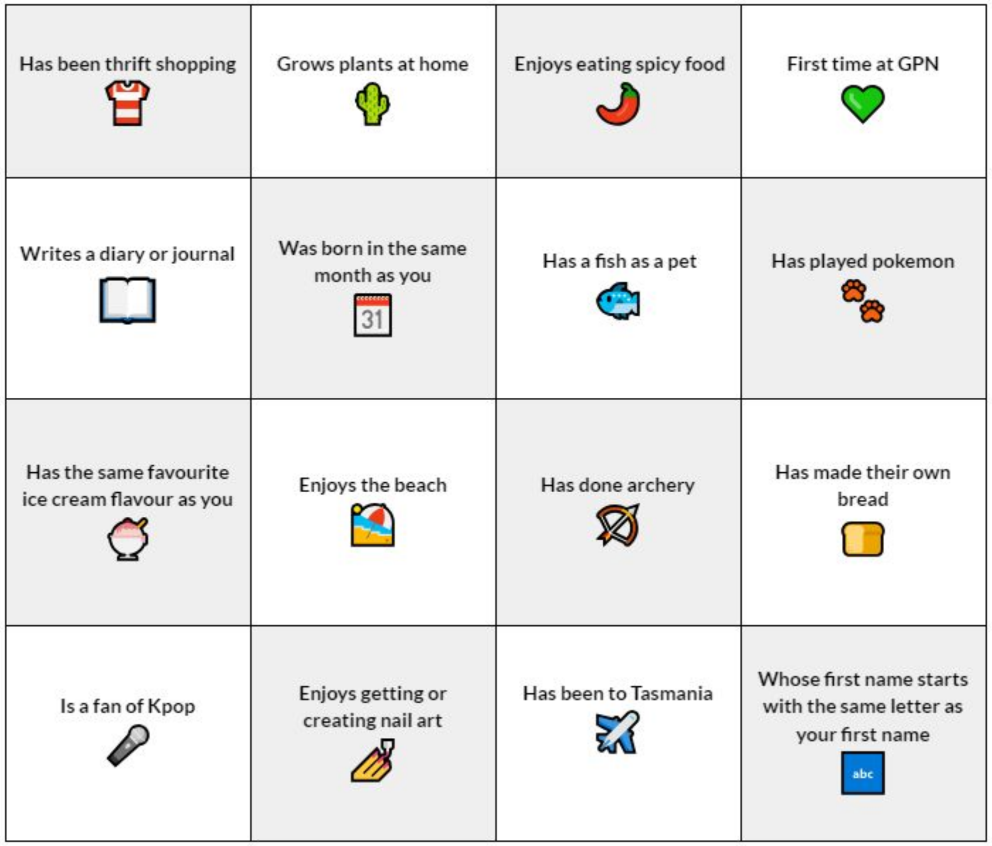
Who are the tutors?

Who are you?

Get to know you BINGO

Grab a printed BINGO sheet & pen

- Read each square
- Find a new friend who can complete any of the squares
- Write their name in the square - you can only put their name in ONE box!
- TUTORS TOO!



Has been thrift shopping	Grows plants at home	Enjoys eating spicy food	First time at GPN
Writes a diary or journal	Was born in the same month as you	Has a fish as a pet	Has played pokemon
Has the same favourite ice cream flavour as you	Enjoys the beach	Has done archery	Has made their own bread
Is a fan of Kpop	Enjoys getting or creating nail art	Has been to Tasmania	Whose first name starts with the same letter as your first name

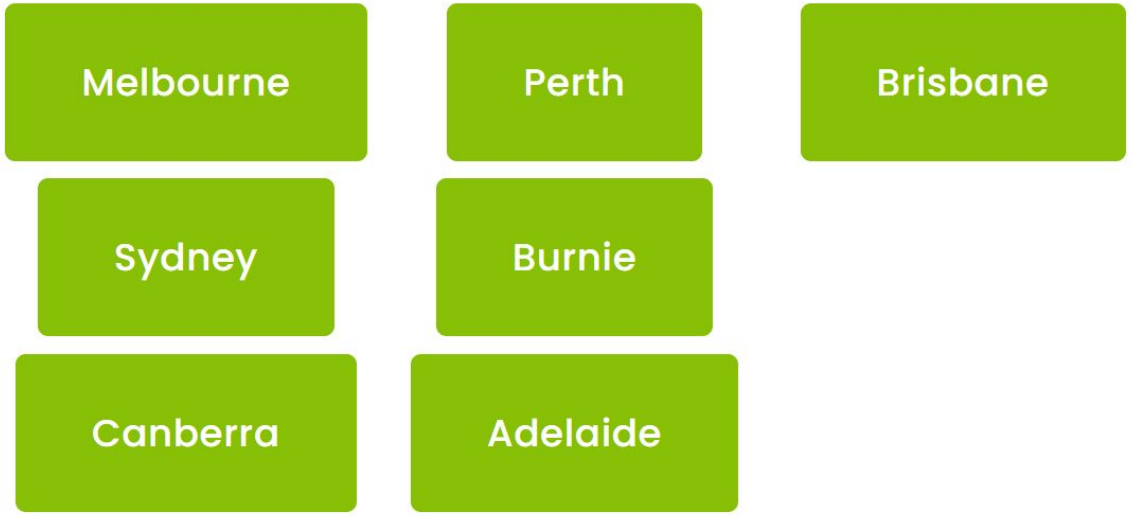
[Link for printing BINGO sheet]

Log on

Log on and jump on the GPN website

girlsprogramming.network/workshop

Click on your location

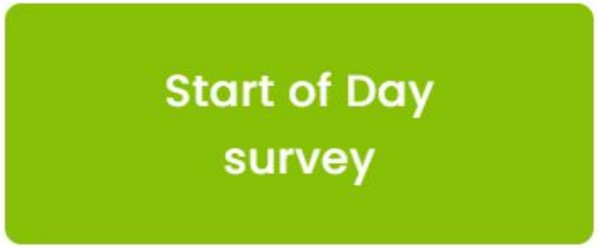


Tell us you’re here!

Click on the

Start of Day Survey

and fill it in now!



Log on

Click on your Room picture

You can see:

- A link to the **Workbook**
- These **Slides** (to take a look back on or go on ahead)
- Other helpful bits like a Cheatsheet to help you code

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 something extra!**

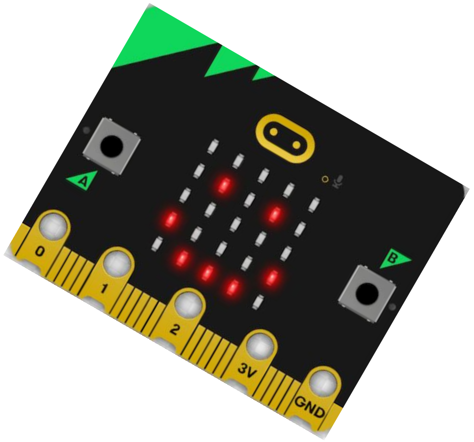
Something to try if you have spare time before the next lecture!

Today’s project!

Tamagotchi- Micro:Bit

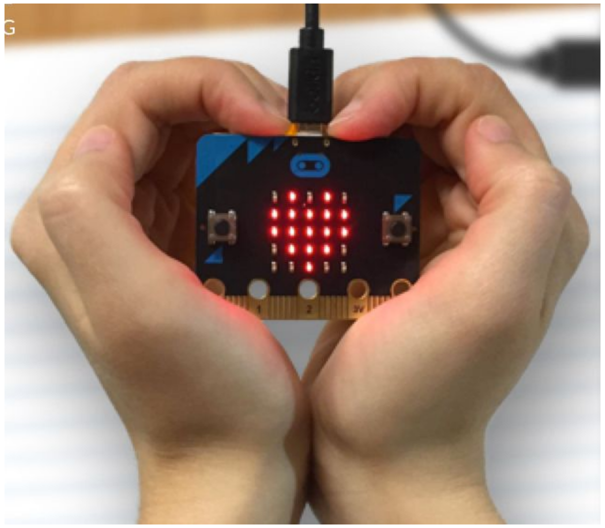
Tamagotchi

- You're going to make your own Tamagotchi electronic pet using a micro:bit
- Tamagotchi pets were a worldwide fad created in Japan in 1996
- Give your pet a name and write some code to feed it, play with it and let it sleep
- **Don't let it get hungry, bored or sleepy!**
- **Keep it alive, watch it grow and change**



Tamagotchi

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

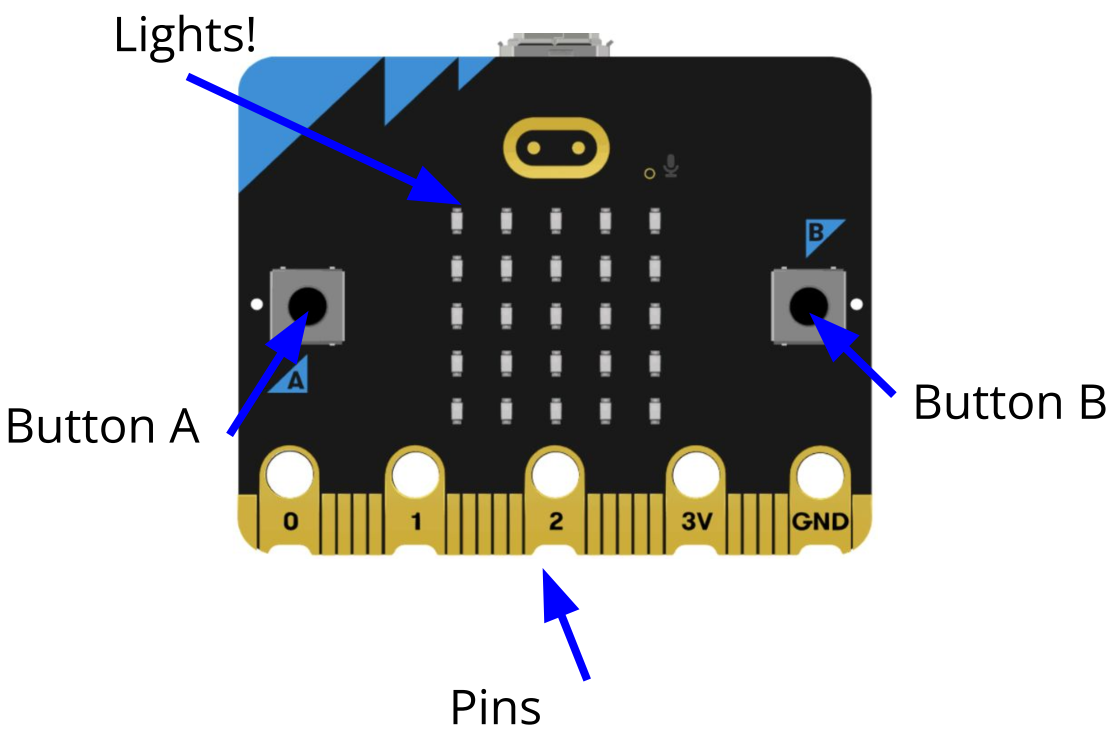
What is a Micro:Bit?

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

Front



What is a Micro:Bit?

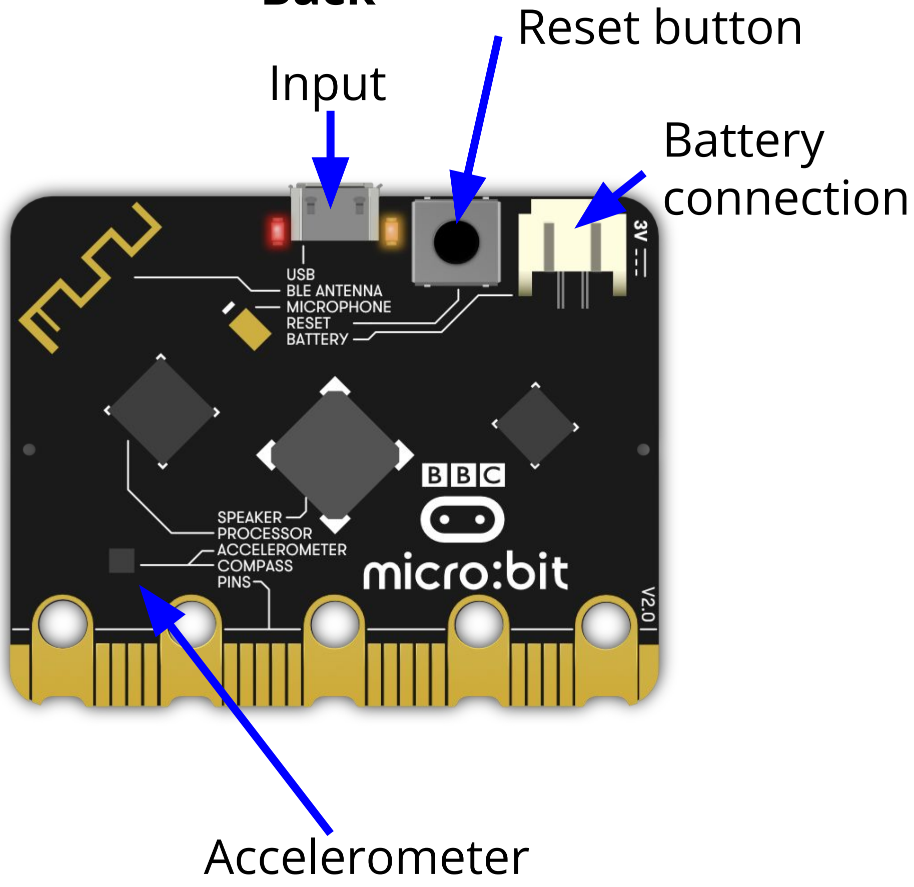
Input: Where we connect the cable from the computer to transfer our code/power to our Micro:Bit

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 you tutor for a battery pack if you need one.

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

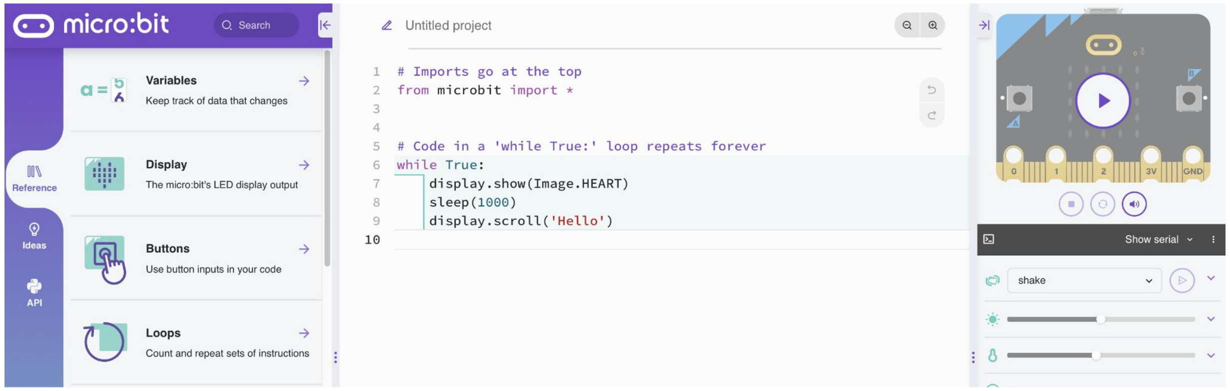
Back



Using python.microbit.org

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

Go to python.microbit.org


micro:bit
Search
Variables
Keep track of data that changes
Display
The micro:bit's LED display output
Buttons
Use button inputs in your code
Loops
Count and repeat sets of instructions
Reference
Ideas
API
Untitled project
Show serial
shake

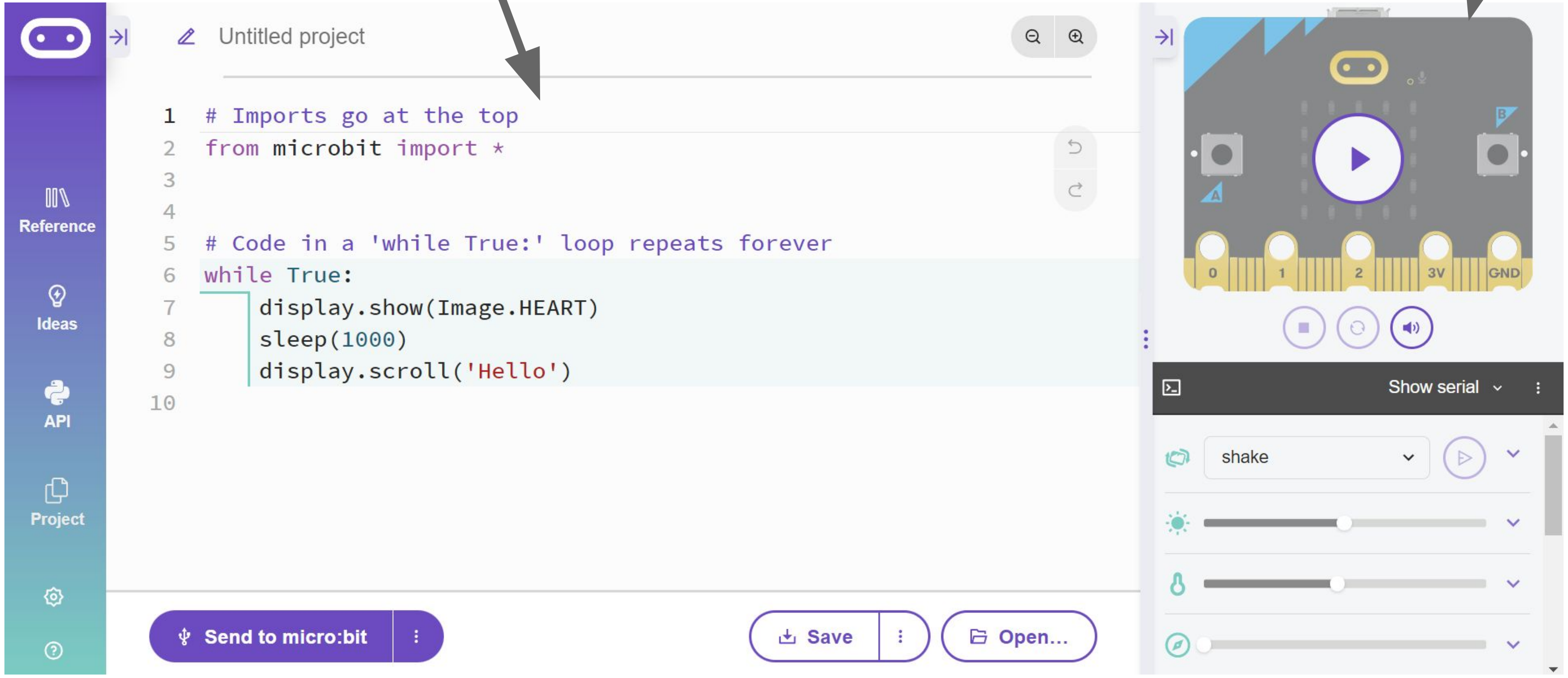
```
# Imports go at the top
from microbit import *


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

You should see this page pop up!

python.microbit.org

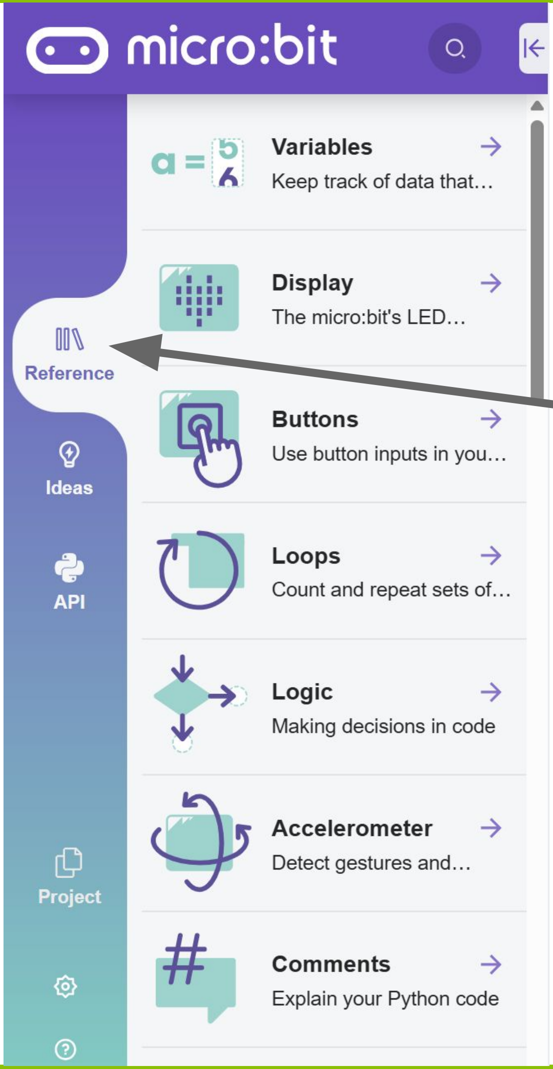
This is where we code

This is the simulator where we test our code



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


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

python.microbit.org



Variables
Keep track of data that…

Display
The micro:bit's LED…

Buttons
Use button inputs in you…

Loops
Count and repeat sets of…

Logic
Making decisions in code

Accelerometer
Detect gestures and…

Comments
Explain your Python code

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 *
```

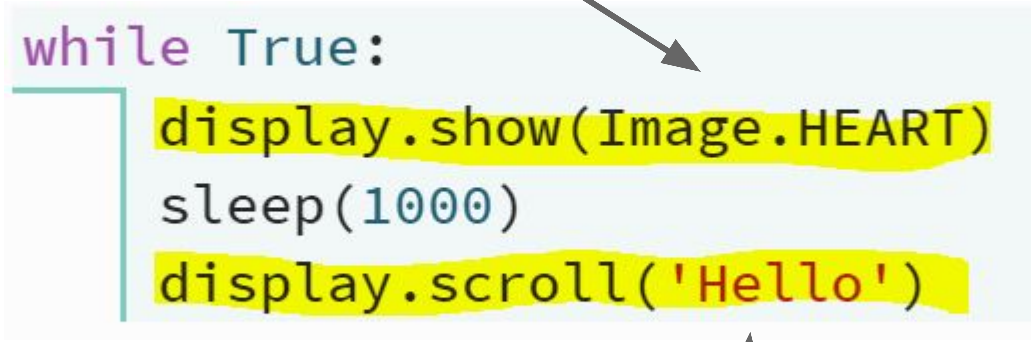
This lets us use lights, sounds, buttons and lots of other cool in our Python code for the Micro:Bit

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')
```

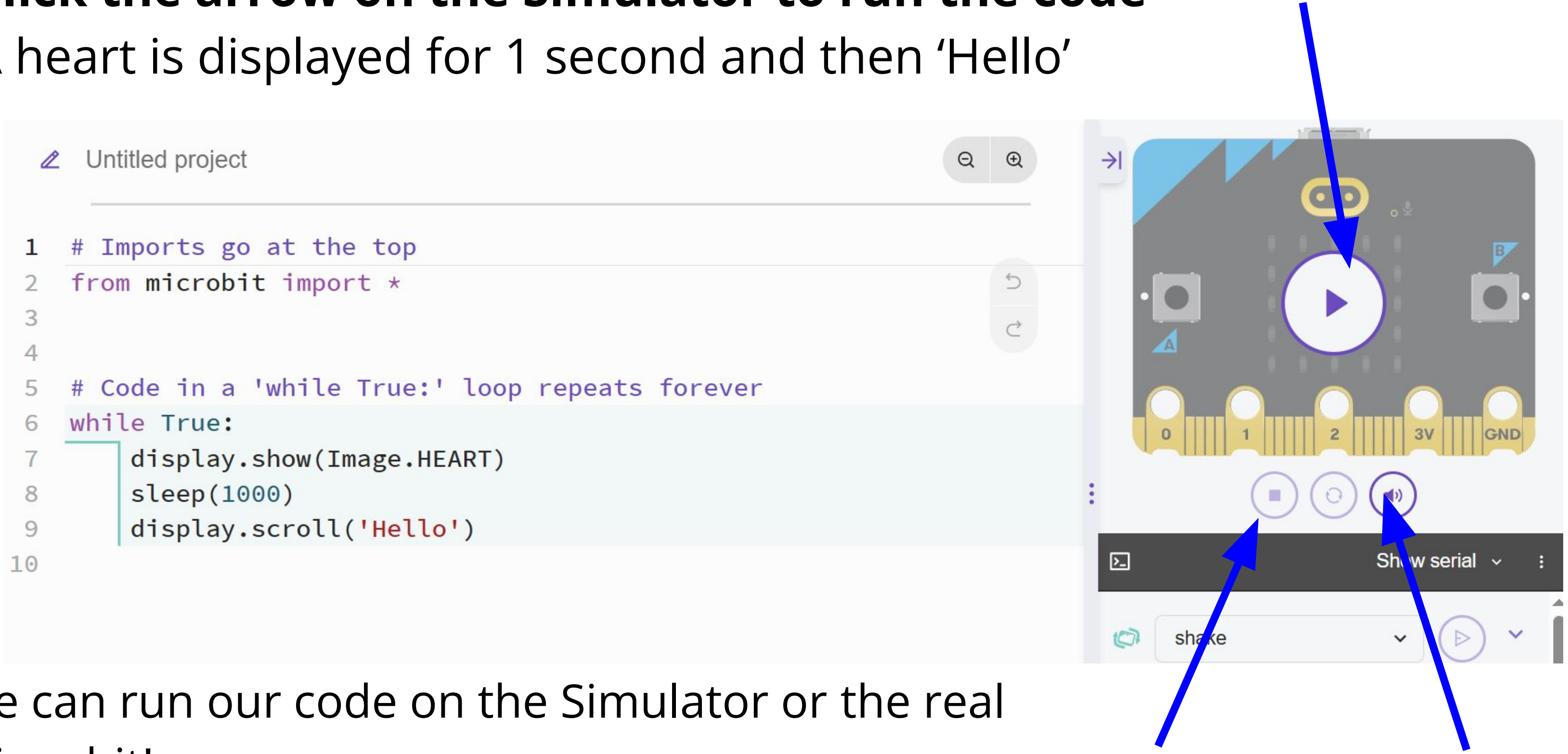
Scroll a word across the display, like 'Hello'

This code is in your **python.makecode.org** coding space - have a look

It's indented in a while loop - so it will repeat forever

Using the Simulator

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



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


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

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

Stop, Restart, Simulator settings are underneath

Stop **Restart**

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

A
B
0
1
2
3V
GND

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!

Chrome or Edge

Run your code on your Micro:Bit like this

1. Make sure your Micro:Bit is plugged into your computer
2. Click **Send to micro:bit** ⋮ 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!

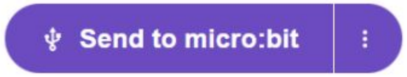
You should see a HEART displayed for 1 second and then HELLO

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
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
```

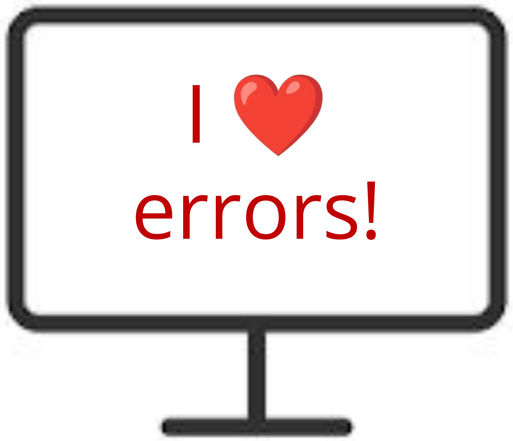
- Programmers use comments to explain what their code does
- You can 'comment out' code to stop it from running

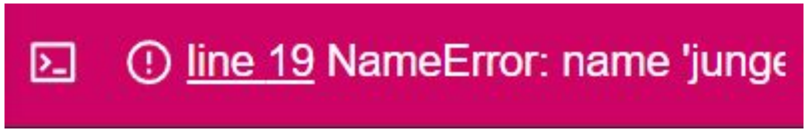
Have a look at the code in the coding space - can you see the purple comments lines starting with the #

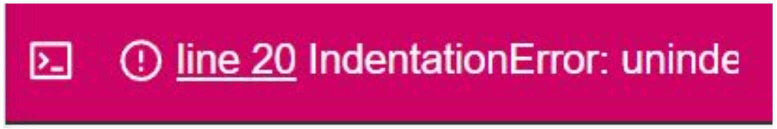
Mistakes are Great! Errors on the Micro:bit!

- Programmers make A LOT of errors!
- Error messages give us hints on how to fix the problem
- Mistakes don't break computers!

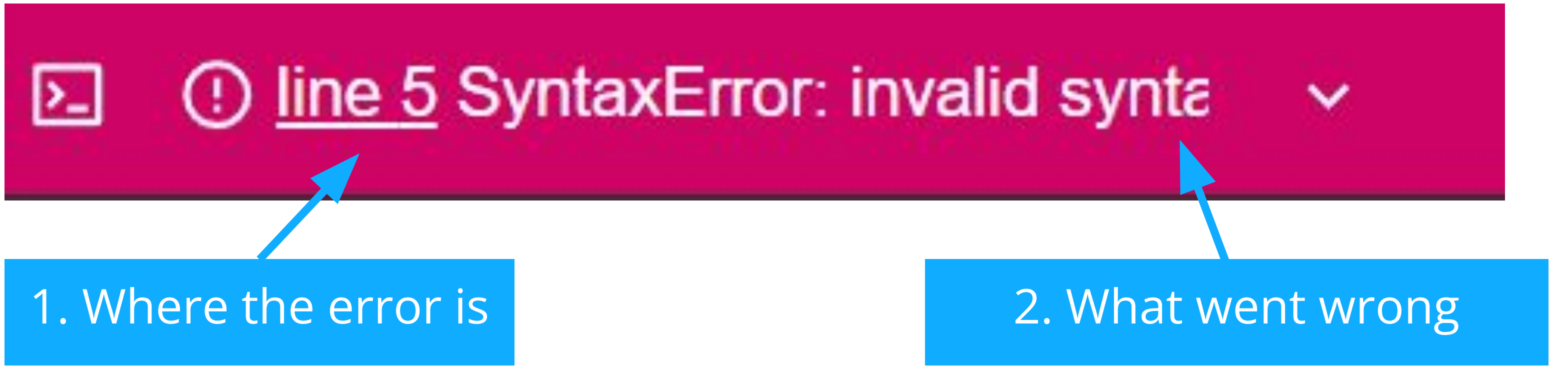
- Lots of unexpected words on the micro:bit is an error message
- Run on the simulator to see it better




line 19 NameError: name 'junge

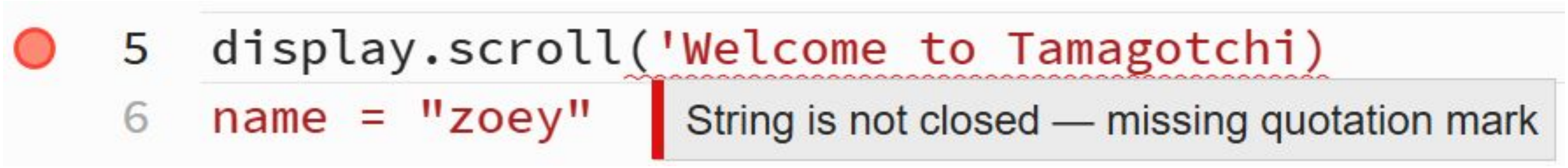

line 20 IndentationError: uninde

We can learn from our mistakes!



```
line 5 SyntaxError: invalid synta
```

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



```
5   display.scroll('Welcome to Tamagotchi)
6   name = "zoey"
```

String is not closed — missing quotation mark

Project Time!

Let's use our MicroBit!

Try Parts 0 & 1 of your Workbook!

The tutors will be around to help!

You've already done the first task!

Intro to Programming

What is programming?

Programming is not a bunch of crazy numbers!

It's giving computers a set of instructions!

A Special Language

A language to talk to dogs!

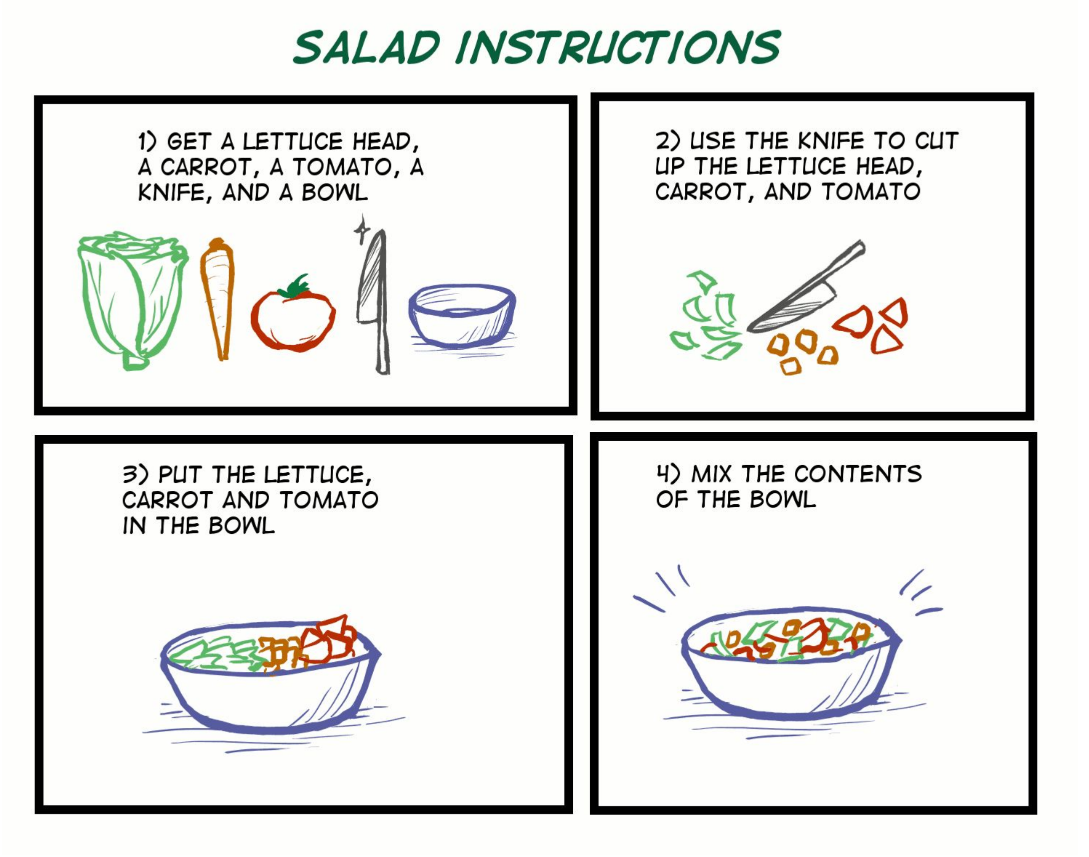
Programming is a language to talk to computers

People are smart! Computers are dumb!

Programming is creating a set of instructions, like a recipe.

Computers do EXACTLY what you say, every time.

Which is great if you give them a good recipe!

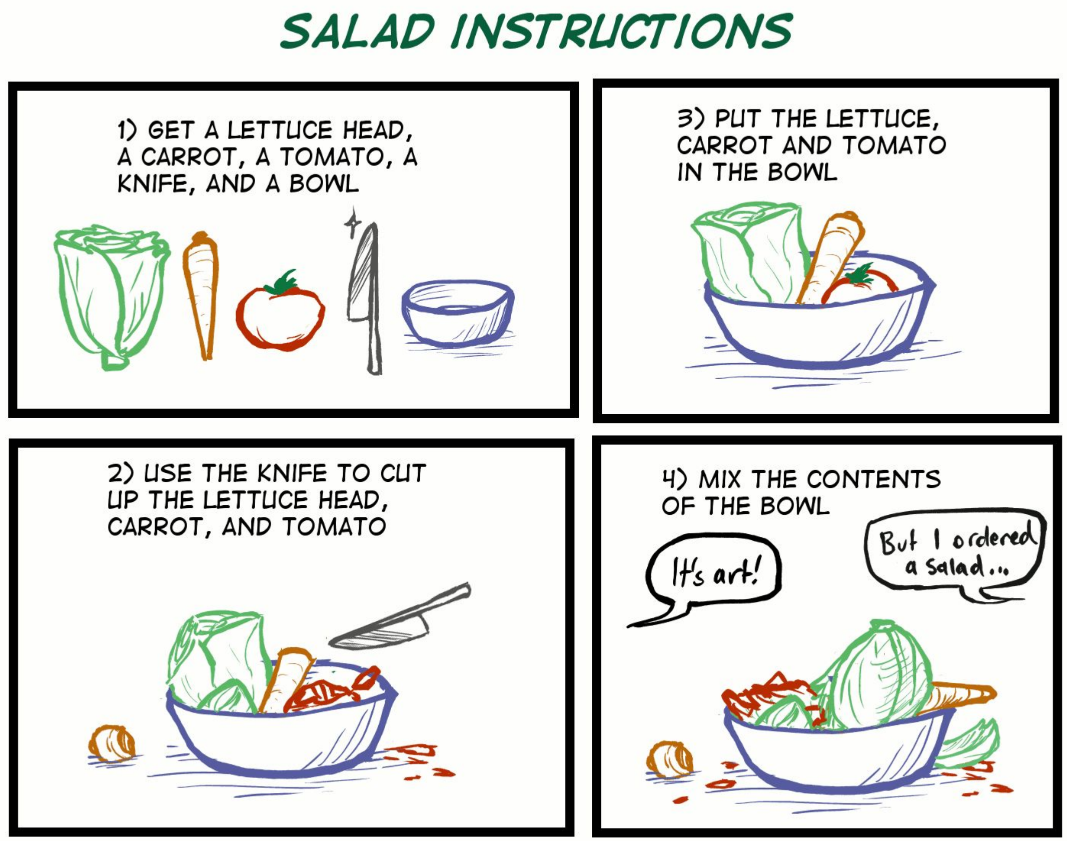

SALAD INSTRUCTIONS
1) GET A LETTUCE HEAD, A CARROT, A TOMATO, A KNIFE, AND A BOWL
2) USE THE KNIFE TO CUT UP THE LETTUCE HEAD, CARROT, AND TOMATO
3) PUT THE LETTUCE, CARROT AND TOMATO IN THE BOWL
4) MIX THE CONTENTS OF THE BOWL

People are smart! Computers are dumb!

But if your recipe is wrong e.g. get it out of order….

A computer wouldn't know this recipe was wrong.

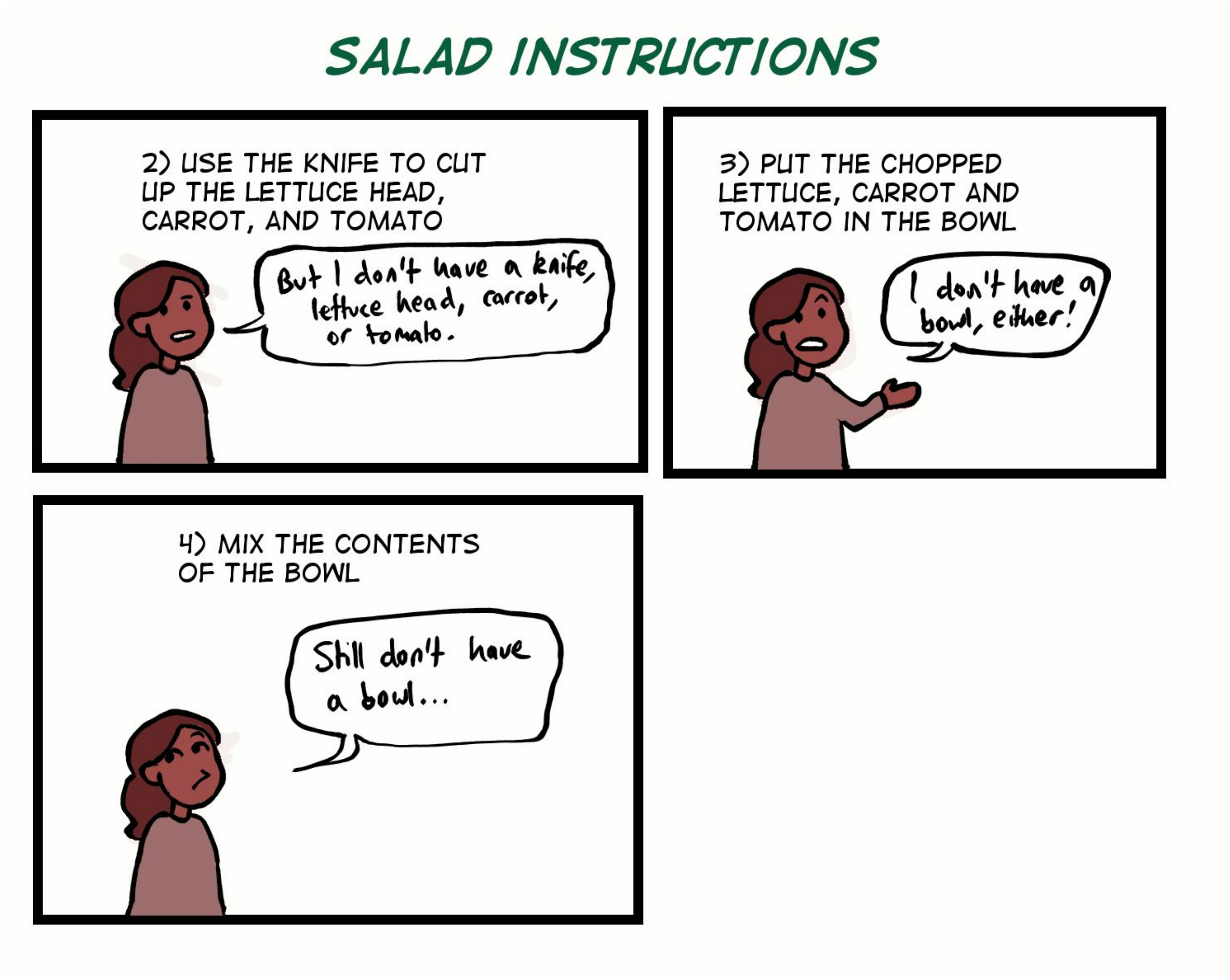
It would still try to make it anyway!


SALAD INSTRUCTIONS
1) GET A LETTUCE HEAD, A CARROT, A TOMATO, A KNIFE, AND A BOWL
3) PUT THE LETTUCE, CARROT AND TOMATO IN THE BOWL
2) USE THE KNIFE TO CUT UP THE LETTUCE HEAD, CARROT, AND TOMATO
4) MIX THE CONTENTS OF THE BOWL
It's art!
But I ordered a salad...

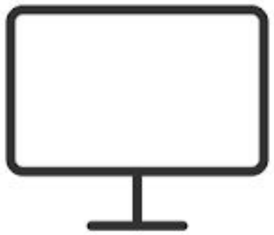
People are smart! Computers are dumb!

Computers are bad at filling in the gaps!

A computer wouldn't know something was missing, it would just freak out!


SALAD INSTRUCTIONS
2) USE THE KNIFE TO CUT UP THE LETTUCE HEAD, CARROT, AND TOMATO
But I don't have a knife, lettuce head, carrot, or tomato.
3) PUT THE CHOPPED LETTUCE, CARROT AND TOMATO IN THE BOWL
I don't have a bowl, either!
4) MIX THE CONTENTS OF THE BOWL
Still don't have a bowl...

Everyone & everything has strengths!



How is the human brain different from a computer's brain?

Everyone & everything has strengths!

- Understand instructions very well despite spelling mistakes or typos
- Solve hard problems
- Invent computers and tell them what to do!
- Get smarter by learning

- Only does exactly what humans tell it
- Does it the same way every time
- Will work endlessly
- Really good at being repetitive
- REALLY fast
- Get smarter when humans tell it how

Variables

No Storing is Boring!

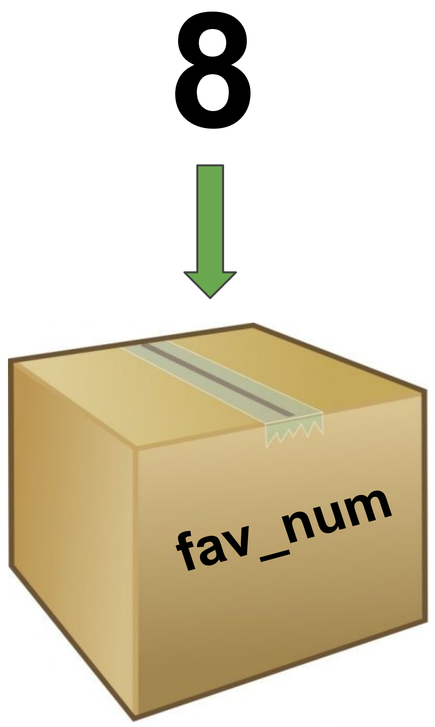
It's useful to be able to remember things for later!
Computers remember things in **"variables"**

Variables are like putting things into a labelled cardboard box.

Let's make our favourite number 8!

In our code we make a variable and set it to a value like this:

fav_num = 8



Variables

Instead of writing the number 8, we can write fav_num. The computer will substitute the fav_num's current value.

```
1. fav_num - 6
```

```
3. fav_num + 21
```

```
2. fav_num * 2
```

```
4. fav_num / 2
```

Variables

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

1. `fav_num - 6`

   2

2. `fav_num * 2`

3. `fav_num + 21`

4. `fav_num / 2`

Variables

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

1. `fav_num - 6`

   2

2. `fav_num * 2`

   16

3. `fav_num + 21`

4. `fav_num / 2`

Variables

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

1. `fav_num - 6`

   2

2. `fav_num * 2`

   16

3. `fav_num + 21`

   29

4. `fav_num / 2`

Variables

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

1. `fav_num - 6`

   2

2. `fav_num * 2`

   16

3. `fav_num + 21`

   29

4. `fav_num / 2`

   4

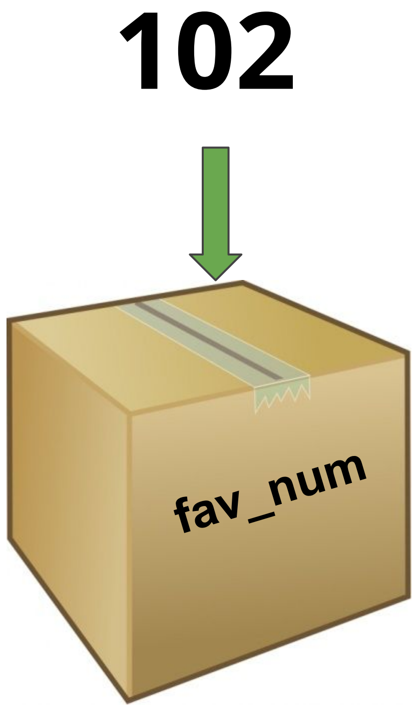
Variables

Variables are useful for storing things that change

Variables contain data that "vary" - hence the word "variable".

Let's change `fav_num` to **102**.

fav_num = 102



Variables

We're able to use our code for a new purpose, without rewriting everything:

1. `fav_num - 6`
```
96
```
2. `fav_num * 2`
```
204
```
3. `fav_num + 21`
```
123
```
4. `fav_num / 2`
```
51
```

Reusing variables

We can replace values in variables and print it with text:

```
animal = "dog"
display.scroll("My favourite animal is a " + animal)
animal = "cat"
display.scroll("My favourite animal is a " + animal)
animal = animal + "dog"
display.scroll("My favourite animal is a " + animal)
```

What will this output?

Reusing variables

We can replace values in variables:

```
animal = "dog"
display.scroll("My favourite animal is a " + animal)
animal = "cat"
display.scroll("My favourite animal is a " + animal)
animal = animal + "dog"
display.scroll("My favourite animal is a " + animal)
```

```
My favourite animal is a dog
My favourite animal is a cat
My favourite animal is a catdog
```

Variables

Your turn!

Can you guess what each `display.scroll` will do?

```
>>> x = 3
>>> display.scroll(x)

>>> display.scroll(x + x)

>>> y = x
>>> display.scroll(y)

>>> y = y + 1
>>> display.scroll(y)
```

Variables

Your turn!

Can you guess what each `display.scroll` will do?

```
>>> x = 3
>>> display.scroll(x)
3
>>> display.scroll(x + x)


>>> y = x
>>> display.scroll(y)


>>> y = y + 1
>>> display.scroll(y)
```

Variables

Your turn!

Can you guess what each `display.scroll` will do?

```
>>> x = 3
>>> display.scroll(x)
3
>>> display.scroll(x + x)
6
>>> y = x
>>> display.scroll(y)


>>> y = y + 1
>>> display.scroll(y)
```

Variables

Your turn!

Can you guess what each `display.scroll` will do?

```
>>> x = 3
>>> display.scroll(x)
3
>>> display.scroll(x + x)
6
>>> y = x
>>> display.scroll(y)
3
>>> y = y + 1
>>> display.scroll(y)
```

Variables

Your turn!

Can you guess what each `display.scroll` will do?

```
>>> x = 3
>>> display.scroll(x)
3
>>> display.scroll(x +
x)
6
>>> y = x
>>> display.scroll(y)
3
>>> y = y + 1
>>> display.scroll(y)
4
```

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!

Scroll... Scroll... Scroll... on the micro:bit

Words are too big to display within a 5x5 grid of lights.

Remember we can display words with **`display.scroll()`**.

```
display.scroll('Hello World')
```

Sometimes the text scrolls across too slowly - you can speed it up with **delay.**

```
display.scroll('Hello World', delay=100)
```

A smaller delay (eg 100 results in faster scrolling).

The default speed is 150!

Multiple Instructions

What happens if we want to change the speed **AND** join variables with strings?

This is how you would do it! :)

```
win_count = 3
display.scroll('Wins: '+ str(win_count), delay=75)
```

Message

delay

See that we need to use **str( )** to convert the number win_count to a string before we can join it (+) with the the other string!

Sleep... zzz! ... on the micro:bit

Computers are really fast, sometimes our program moves too quickly to enjoy it!

For example:

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

Without a sleep, the computer will run through the code so quickly, and we will only see a CONFUSED face.

We can slow it down by using **sleep()**

Sleep is done in milliseconds (so the number of seconds x 1000)

Different data!

There are lots of types of data! Our main 4 ones are these:

Strings

Things in quotes used for storing text

```
a = “This is a string”
display.scroll(a)
```

Different data!

There are lots of types of data! Our main 4 ones are these:

Strings

Things in quotes used for storing text

```
a = “This is a string”
display.scroll(a)
```

Ints

Whole numbers we can do maths with

```
a = 1
b = 2
display.scroll(a + b)
```

Different data!

There are lots of types of data! Our main 4 ones are these:

Strings

Things in quotes used for storing text

```
a = “This is a string”
display.scroll(a)
```

Ints

Whole numbers we can do maths with

```
a = 1
b = 2
display.scroll(a + b)
```

Floats

Decimal numbers for maths

```
a = 1.5
b = 2.0
display.scroll(a / b)
```

Different data!

There are lots of types of data! Our main 4 ones are these:

Strings

Things in quotes used for storing text

```
a = “This is a string”
display.scroll(a)
```

Ints

Whole numbers we can do maths with

```
a = 1
b = 2
display.scroll(a + b)
```

Floats

Decimal numbers for maths

```
a = 1.5
b = 2.0
display.scroll(a / b)
```

Booleans

For `True` and `False`

```
boring = False
a = 5 > 3
display.scroll(a)
```

Different data!

There are lots of types of data! Our main 4 ones are these:

Strings

Things in quotes used for storing text

```
a = “This is a string”
display.scroll(a)
```

```
This is a string
```

Ints

Whole numbers we can do maths with

```
a = 1
b = 2
display.scroll(a + b)
```

```
3
```

Floats

Decimal numbers for maths

```
a = 1.5
b = 2.0
display.scroll(a / b)
```

```
0.75
```

Booleans

For `True` and `False`

```
boring = False
a = 5 > 3
display.scroll(a)
```

```
True
```

Project Time!

Let’s use our MicroBit!

Try Part 2 of your Workbook!

The tutors will be around to help!

If Statements

Conditions!

Conditions let us make a 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

```
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

5 < 10	True	"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

5 < 10	True	"Dog" == "dog"
3 + 2 == 5	True	"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

```
5 < 10        True      "Dog" == "dog"
3 + 2 == 5    True      "D" in "Dog"
5 != 5        False     "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

5 < 10	True	"Dog" == "dog"	False
3 + 2 == 5	True	"D" in "Dog"	
5 != 5	False	"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

5 < 10	True	"Dog" == "dog"	False
3 + 2 == 5	True	"D" in "Dog"	True
5 != 5	False	"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

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

Looping

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")
```

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
- <u>So it is `True`!</u>

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
- <u>So it is `True`!</u>

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:
    display.scroll("that’s a small
    number")
```

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

```
>>>
```

Conditions

How about a different number???

```
fave_num = 9000

if fave_num < 10:
    display.scroll("that’s a small number")
```

What do you think happens?

```
>>>
```

Nothing!

If statements

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

This line ...

... controls this line

If statements

Actually .....

This line ...

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

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

If statements

```
fave_num = 5
if fave_num < 10:
    display.scroll("that's a small number")
    display.scroll("and I like that")
    display.scroll("A LOT!!")
```

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

```
>>>
```

If statements

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

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

If statements

```
word = "GPN"
if word == "GPN":
  display.scroll("GPN is awesome!")
```

What happens?

If statements

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

What happens?

```
>>> GPN is awesom
```

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!
```

Don’t forget : and ==

:

Colon at the end of each if, elif or else line

==

Two equals signs if you’re checking for equals in the condition line

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

OR

What if you wanted to check if the weather was raining or snowing, and take an umbrella for either possibility.

```
weather = "raining"
if weather == "raining":
  print("Take an umbrella!")
if weather == "snowing":
  print("Take an umbrella!")
```

This also doesn’t look very nice. If you change what you’re printing you have to change it twice.

OR

Instead, you can use OR

```
weather = "snowing"
if weather == "raining" or weather == "snowing":
  print("Take an umbrella!")
```

What happens?

```
>>>
```

OR

Instead, you can use OR

```
weather = "snowing"
if weather == "raining" or weather == "snowing":
  print("Take an umbrella!")
```

What happens?

```
>>> Take an umbrella!
```

While Loops

Loops

Are we there yet?
No
Are we there yet?
NO! Are we there yet?
NO! Are we there yet?
NO!
How long do we have to do this????
Until we get there....
Are we there yet?
YES....
... HECTIC!!

We know how to do things on repeat!

Sometimes we want to do some code on repeat!

Introducing … `while` loops!

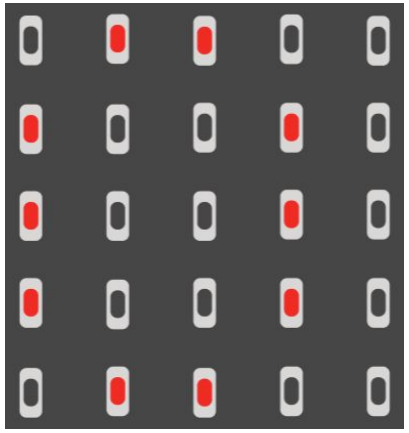
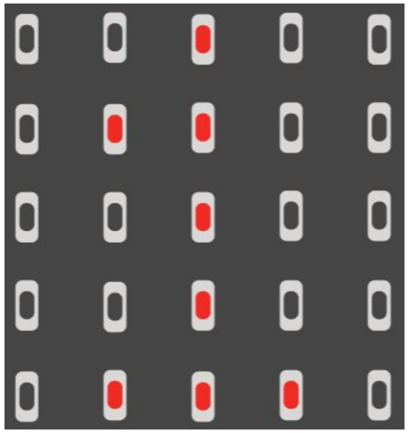
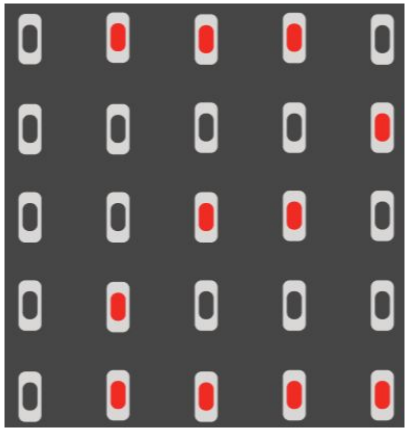
What do you think this does?

```
i = 0
while i < 3:
    display.scroll(i)
    i = i + 1
```

Introducing ... `while` loops!

What do you think this does?

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

Introducing … `while` loops!

Stepping through a while loop…

Introducing … while loops!

One step at a time!

```
i = 0
while i < 3:
    display.scroll(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:
    display.scroll(i)
    i = i + 1
```

MY VARIABLES

```
i = 0
```

Introducing ... `while` loops!

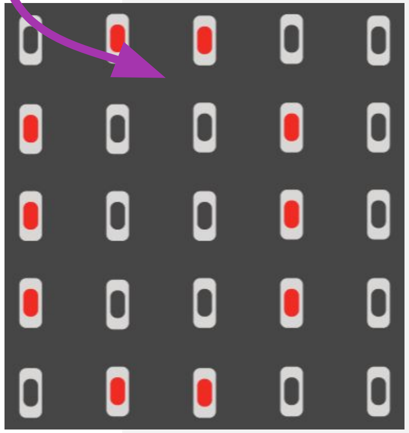
One step at a time!

Print !

```
i = 0
while i < 3:
    display.scroll(i)
    i = i + 1
```

MY VARIABLES

```
i = 0
```



Introducing ... while loops!

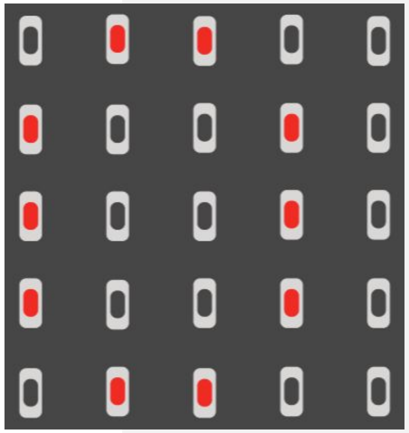
One step at a time!

```
i = 0
while i < 3:
    display.scroll(i)
◆   i = i + 1
```

MY VARIABLES

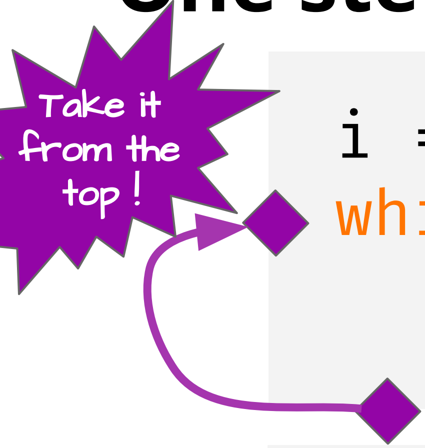
~~i = 0~~
i = 1

UPDATE TIME !



Introducing ... while loops!

One step at a time!

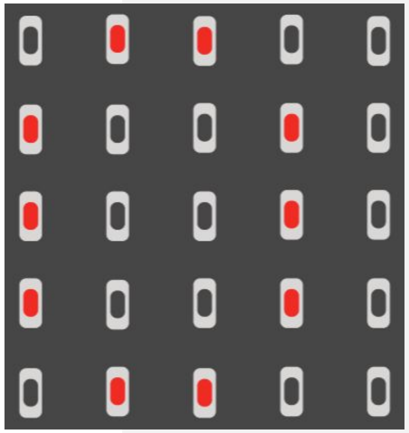


```
i = 0
while i < 3:
    display.scroll(i)
    i = i + 1
```

MY VARIABLES

~~i = 0~~

i = 1



Introducing ... while loops!

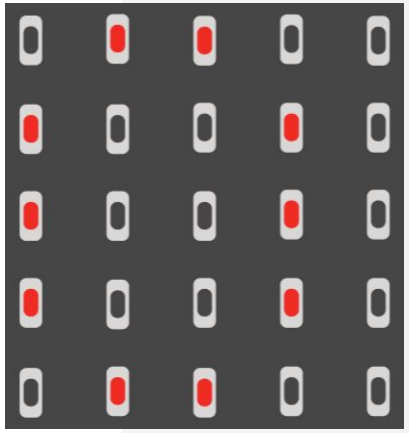
One step at a time!

I is less than 3 !

```
i = 0
while i < 3:
    display.scroll(i)
    i = i + 1
```

MY VARIABLES

~~i = 0~~
i = 1



Introducing ... while loops!

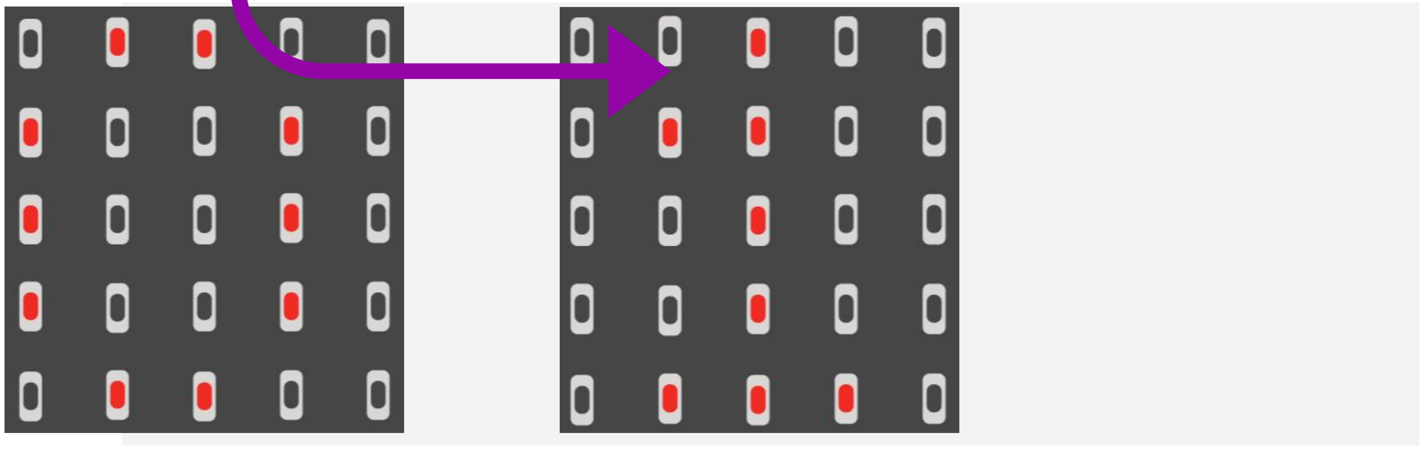
One step at a time!

Print !

```
i = 0
while i < 3:
    display.scroll(i)
    i = i + 1
```

MY VARIABLES

~~i = 0~~
i = 1



Introducing ... `while` loops!

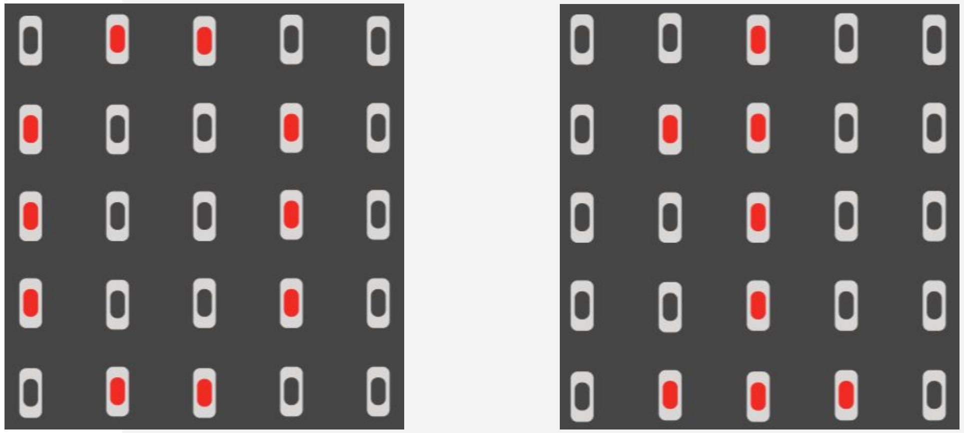
One step at a time!

```
i = 0
while i < 3:
    display.scroll(i)
◆   i = i + 1
```

MY VARIABLES

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

UPDATE TIME !



Introducing ... while loops!

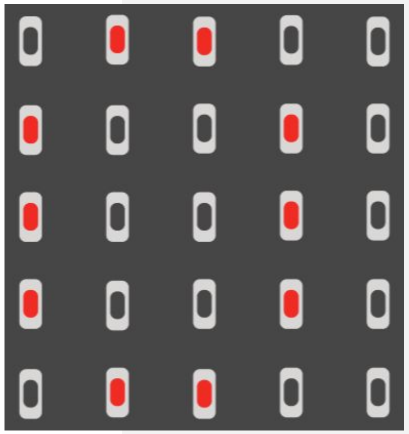
One step at a time!

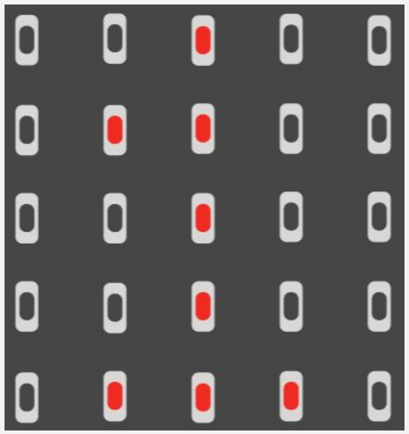
Take it from the top !

```
i = 0
while i < 3:
    display.scroll(i)
    i = i + 1
```

MY VARIABLES

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





Introducing ... while loops!

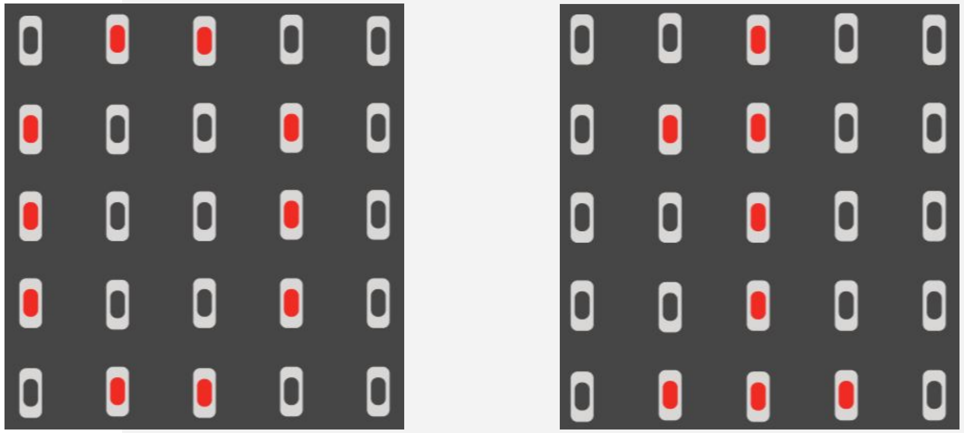
One step at a time!

2 is less than 3 !

```
i = 0
while i < 3:
    display.scroll(i)
    i = i + 1
```

MY VARIABLES

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



Introducing … while loops!

One step at a time!

Print !

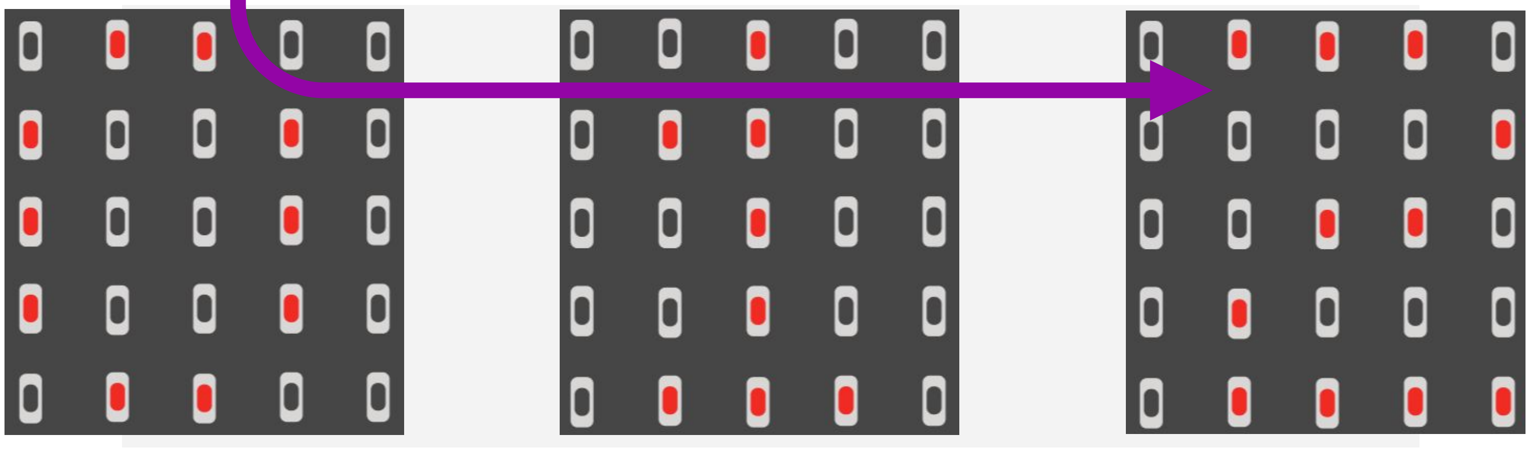
```
i = 0
while i < 3:
    display.scroll(i)
    i = i + 1
```

MY VARIABLES

~~i = 0~~

~~i = 1~~

i = 2



Introducing ... `while` loops!

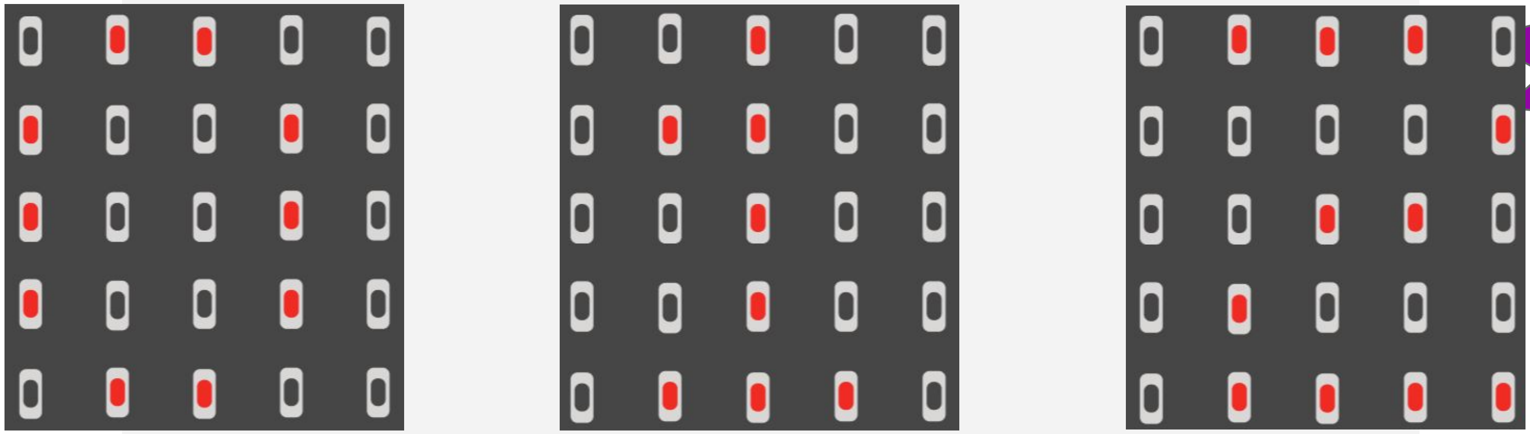
One step at a time!

```
i = 0
while i < 3:
    display.scroll(i)
    i = i + 1
```

MY VARIABLES

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

UPDATE TIME !



Introducing ... while loops!

One step at a time!

Take it from the top!

```
i = 0
while i < 3:
    display.scroll(i)
    i = i + 1
```

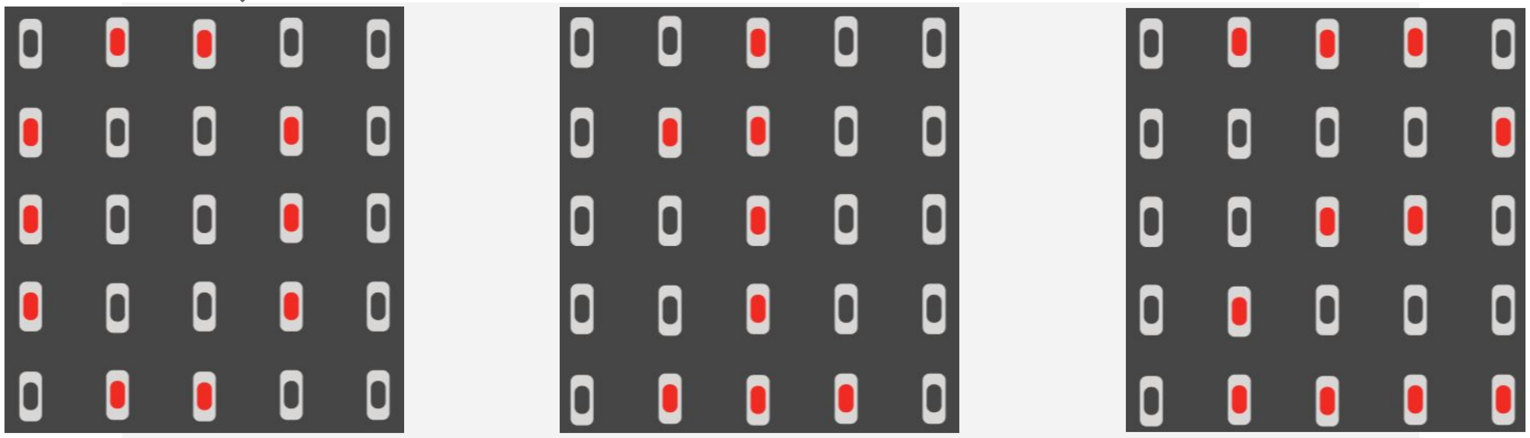
MY VARIABLES

~~i = 0~~

~~i = 1~~

~~i = 2~~

i = 3



Introducing … while loops!

One step at a time!

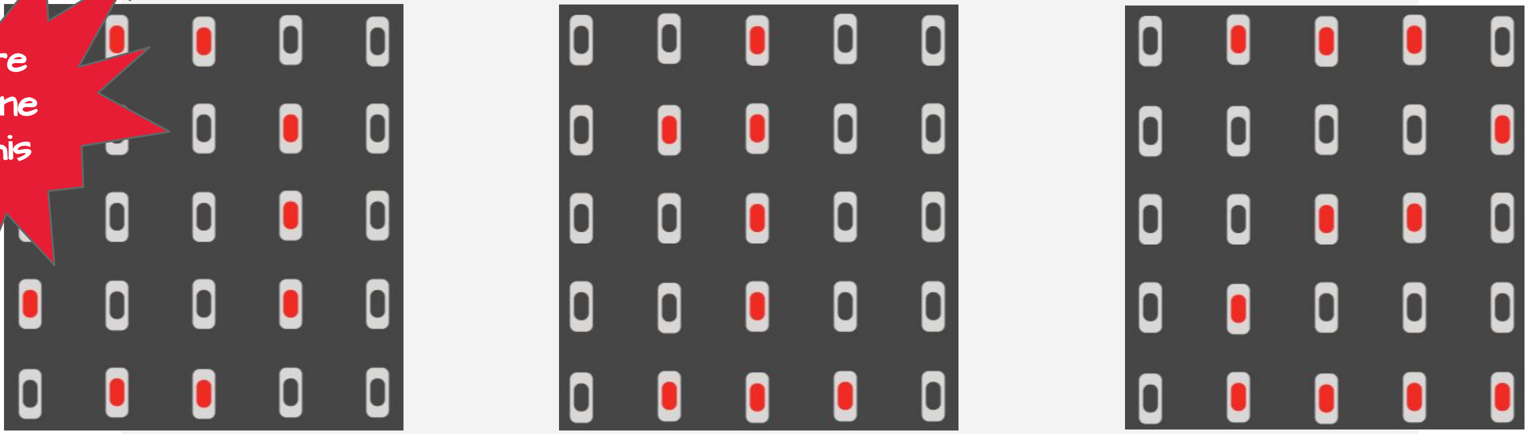
```
i = 0
while i < 3:
    display.scroll(i)
    i = i + 1
```

MY VARIABLES

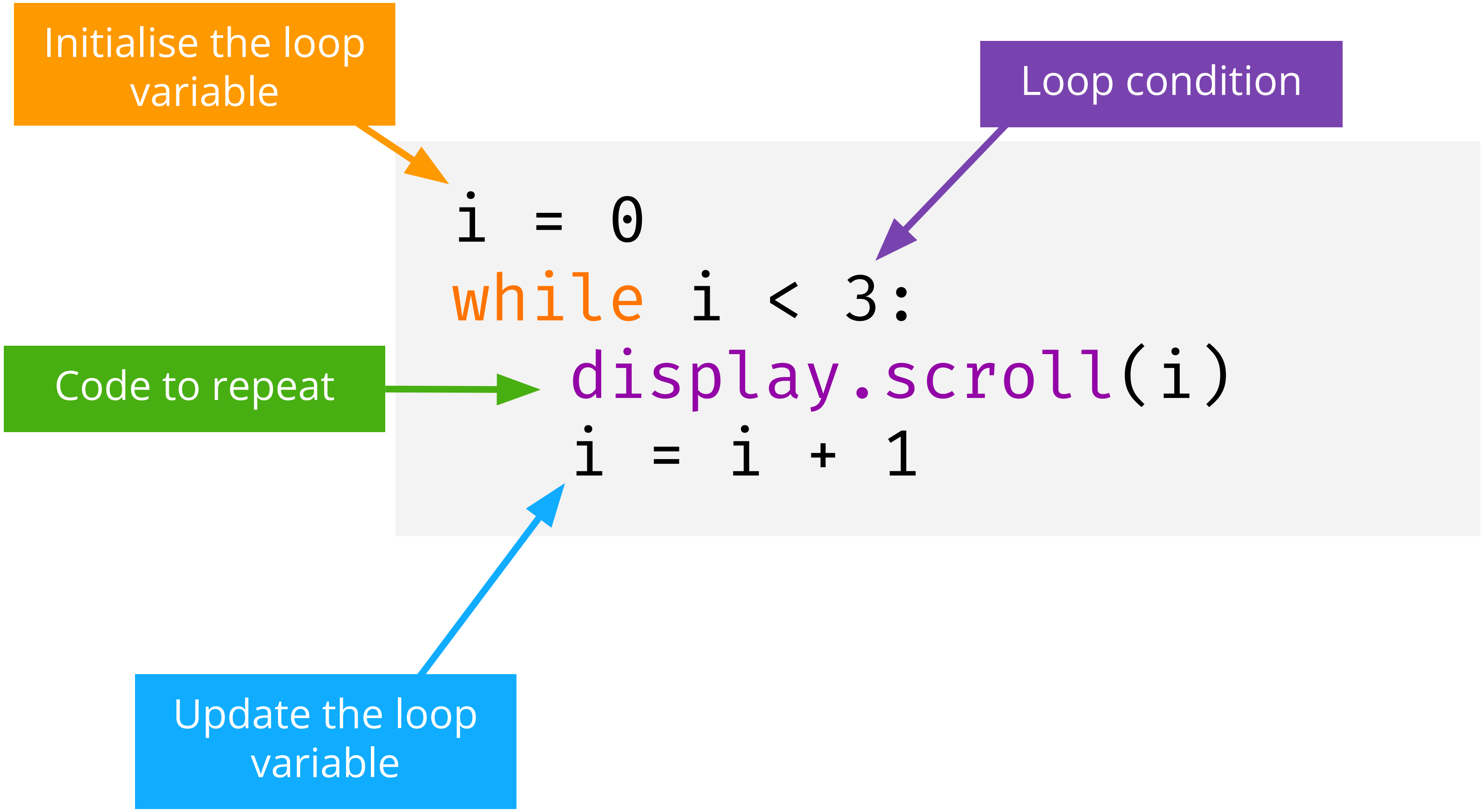
~~i = 0~~
~~i = 1~~
~~i = 2~~
i = 3

3 IS NOT less than 3 !

We are done with this loop!



Introducing … `while` loops!



```
i = 0
while i < 3:
    display.scroll(i)
    i = i + 1
```

What happens when.....

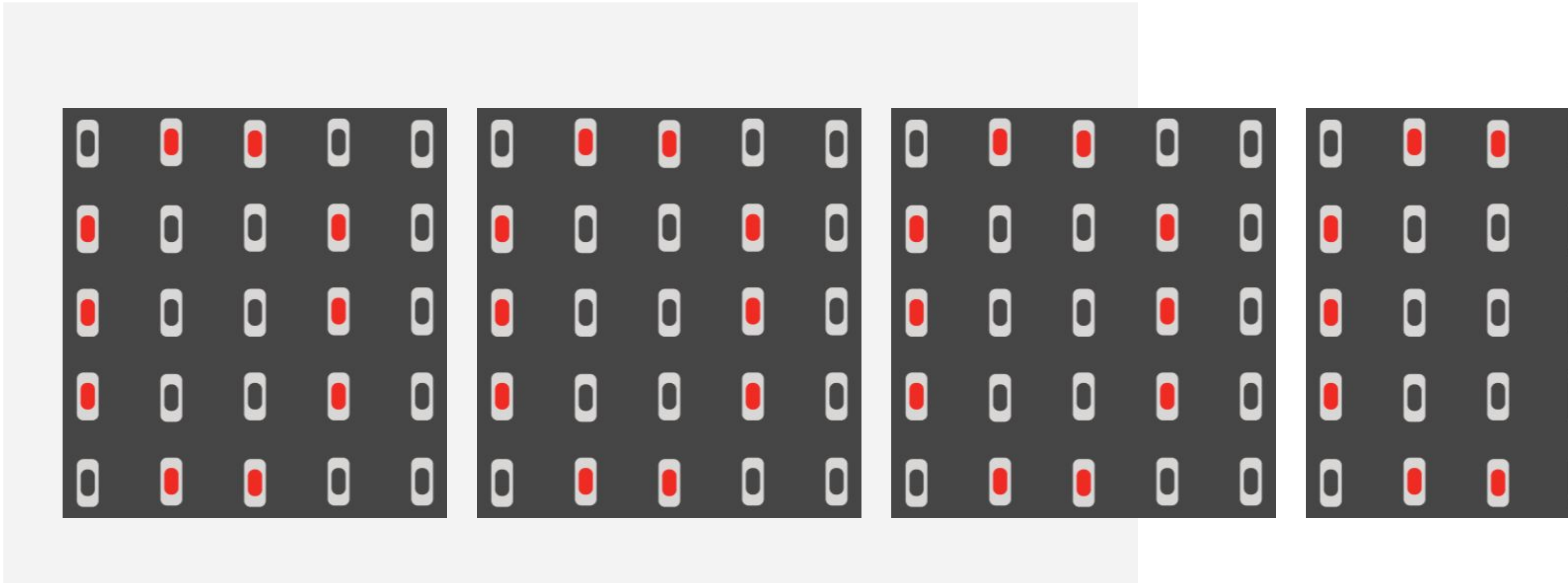
What happens if we forget to update the loop variable?

```
i = 0
while i < 3:
    display.scroll(i)
```

What happens when.....

What happens if we forget to update the loop variable?

```
i = 0
while i < 3:
    display.scroll(i)
```



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:
    display.scroll("Are we there yet?")
```

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:
    display.scroll("Are we there yet?")
```

```
Are we the
```

Give me a break!

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

That's when we use the break keyword!

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

    if button_a.is_pressed():
        break

display.clear()
```

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)
```

Project Time!

while we’re here:

See `if` you can do Part 3

The tutors will be around to help!

Micro:Bit Inputs

Conditions!

Conditions let us make a 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

```
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

```
5 < 10          True          "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

5 < 10	True	"Dog" == "dog"
3 + 2 == 5	True	"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

```
5 < 10        True      "Dog" == "dog"
3 + 2 == 5    True      "D" in "Dog"
5 != 5        False     "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

5 < 10	True	"Dog" == "dog"	False
3 + 2 == 5	True	"D" in "Dog"	
5 != 5	False	"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

5 < 10	True	"Dog" == "dog"	False
3 + 2 == 5	True	"D" in "Dog"	True
5 != 5	False	"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

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

Conditions

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

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

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

```
>>>
```

Conditions

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

```
fave_num = 5
if True
    display.scroll("that's a small number")
```

What do you think happens?

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

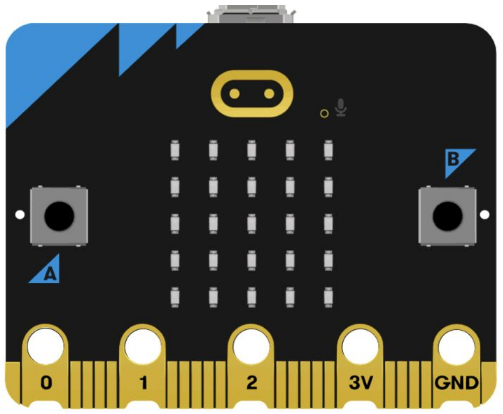
Buttons!

Your Micro:Bit has 2 buttons: Button A and Button B

You can use this code to check if a button is pressed:

```
if button_a.was_pressed():

If button_b.was_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.was_pressed():
   display.show(Image.HAPPY)

if button_b.was_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.was_pressed():
    display.show(Image.HAPPY)

if button_b.was_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.was_pressed():
   display.show(Image.HAPPY)

if button_b.was_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?

Accelerometer!

Your micro:bit has a motion sensor (accelerometer).

This sensor has the ability to detect when you shake it or tilt it left to right, backwards and forwards and up and down.

We can use a **while loop** in yoru code like this to continually check if the micro:bit has been shaken:

Information from the sensor

```
while True:
  if accelerometer.was_gesture('shake'):
```

Accelerometer!

What do you think this code does?

```
while True:
  if accelerometer.was_gesture('shake'):
      display.scroll('I'm getting dizzy')
```

Accelerometer!

What do you think this code does?

```
while True:
  if accelerometer.was_gesture('shake'):
      display.scroll('I’m getting dizzy')
```

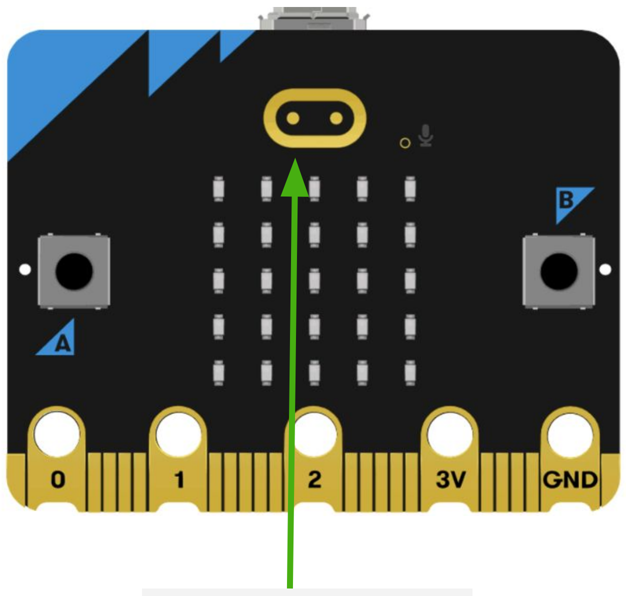
It will display ‘I’m getting dizzy’ every time the micro:bit is shaken

Pin Logo!

Your Micro:Bit has touch sensitive pin logo at the top of the Micro:bit.

You can use this code to check if the pin logo is being touched.

```
if pin_logo.is_touched():
```



Pin Logo

Pin Logo!

What do you think this code does?

```
while True:
    if pin_logo.is_touched():
        display.show(image.DUCK)
    else:
        display.clear()
```

Pin Logo!

What do you think this code does?

```
while True:
    if pin_logo.is_touched():
        display.show(image.DUCK)
    else:
        display.clear()
```

While the pin is being touched, the duck is displayed on the microbit screen.

While the pin is not being touched, the screen is blank.

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** seconds?

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** seconds?

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** seconds?

10,500

Strings and Ints!

Integers are numbers in python.

We can do maths with integers but not strings.

Predict what will happen if we write and run the following code.

1. `5 + "5"`

Strings and Ints!

Integers are numbers in python.

We can do maths with integers but not strings.

Predict what will happen if we write and run the following code.

1. `5 + "5"`

   TypeError: can only concatenate str (“not int”) to str

Strings and Ints!

Integers are numbers in python.

We can do maths with integers but not strings.

Predict what will happen if we write and run the following code.

1. `5 + "5"`

   TypeError: can only concatenate str (“not int”) to str

2. But, we can turn a string into an integer using int()

```
5 + int("5")
```

Strings and Ints!

Integers are numbers in python.

We can do maths with integers but not strings.

Predict what will happen if we write and run the following code.

1.
```
5 + "5"
```

TypeError: can only concatenate str ("not int") to str

2. But, we can turn a string into an integer using int()

```
5 + int("5")
10
```

Strings and Ints!

Integers are numbers in python.

We can do maths with integers but not strings.

Predict what will happen if we write and run the following code.

1. `5 + "5"`

   TypeError: can only concatenate str (“not int”) to str

2. But, we can turn a string into an integer using int()

   `5 + int("5")`

   `10`

3. Similarly, we turn an integer into a string using str()

   `str(5) + "5"`

Strings and Ints!

Integers are numbers in python.

We can do maths with integers but not strings.

Predict what will happen if we write and run the following code.

1. `5 + "5"`

   TypeError: can only concatenate str ("not int") to str

2. But, we can turn a string into an integer using int()

   ```
   5 + int("5")
   10
   ```

3. Similarly, we turn an integer into a string using str()

   ```
   str(5) + "5"
   "55"
   ```

Project Time!

Does that press your buttons?

Try to do Parts 4-9!

The tutors will be around to help!

Tell us what you think!

Click on the

End of Day Form

and fill it in now!