Welcome to the Labs!

Secret Diary Chatbot!

Thank you to our Sponsors!

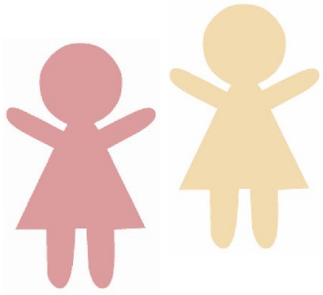
Platinum Sponsor:

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!

Introduction to Edstem

Signing up to Edstem

We are shifting all our courses to a new website called “Edstem” so here’s an overview of how to sign up and how to use it.

First let’s go through how to create an account.

1. Follow this join link: https://edstem.org/au/join/wFAAsK
2. Put in your name and your personal email address
3. Click Create Account
4. Go to your email to verify your account
5. Create a password
6. It should then take you to the courses home page. Click on the one we will be using for this project; ChatbotP

If you don’t have access to your email account, ask a tutor for a GPN edStem login

Getting to the lessons

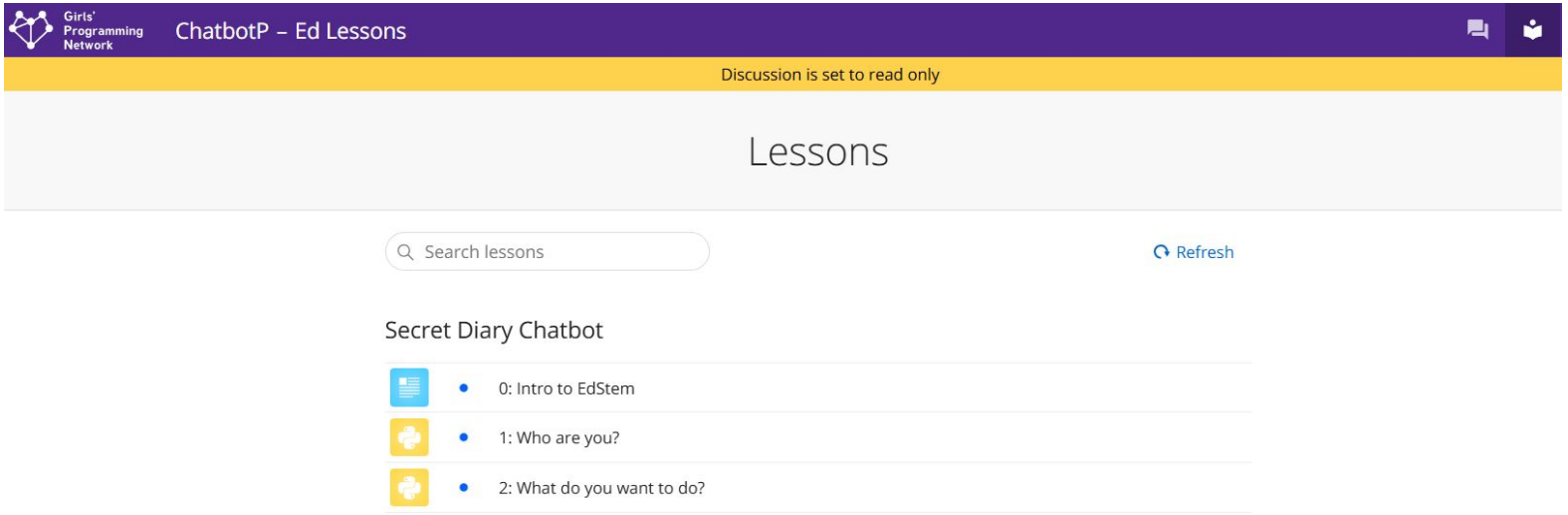
Once you are in the course, you'll be taken to a discussion page.

Click the button for the lessons page (top right - looks like a book)

The Anatomy of the workbook

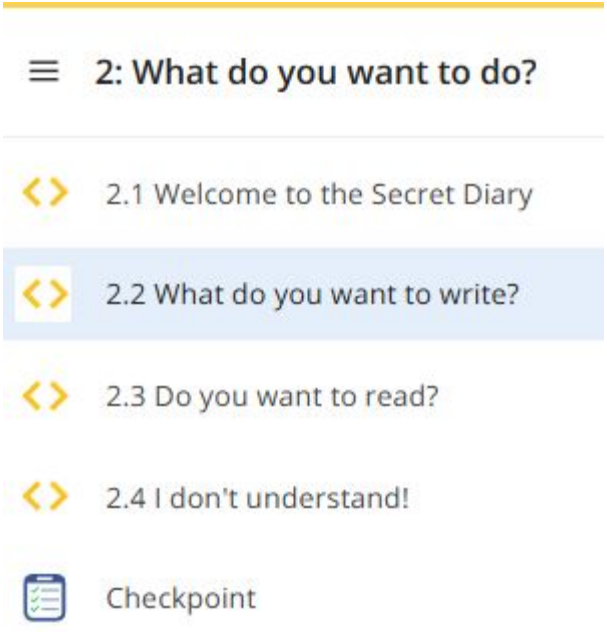
The main page:

- Heading at the top that tells you the project (ChatbotP)
- List of “Chapters” - they have icons that looks like this:
- To complete your project, work through the chapters one at a time



Inside a Chapter

Inside a chapter there are two main types of pages:



1. **Lessons** - where you will do your coding.

   They have this icon: 

2. **Checkpoints**  Checkpoint

   Each chapter has a checkpoint to complete to move to the next chapter. Make sure you scroll down to see all the questions in a checkpoint.

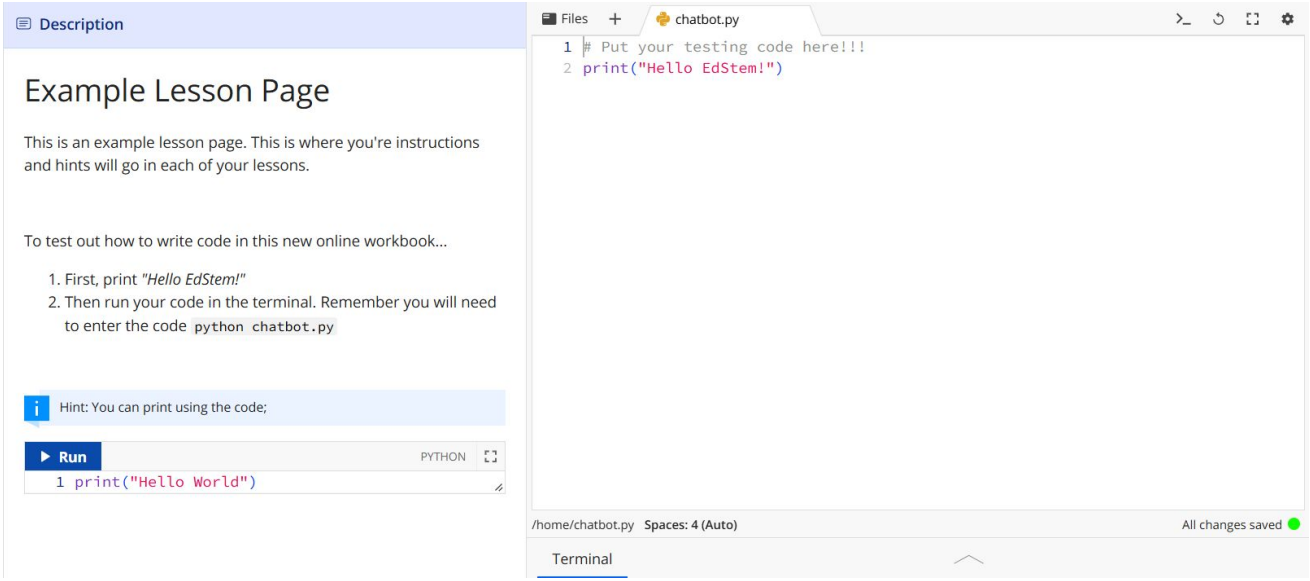
How to do the work

In each lesson there is:

- A section on left with instructions
- A section on right for your code

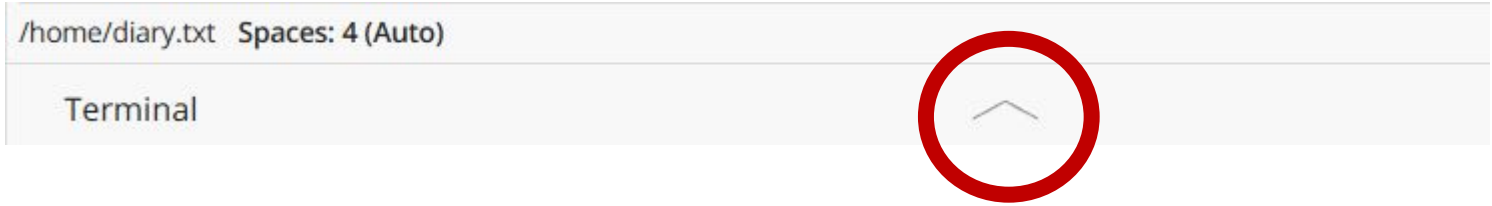
You will need to **copy your code from the last lesson**, then follow the instructions to change your code

There are also Hints and Code Blocks to help you



Running your code...

1. Open the Terminal window below your code



2. Click button that says “Click here to activate the terminal”.

3. Your code should run automatically.
4. Click the button again to rerun your code.
5. You can resize the Terminal window.

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

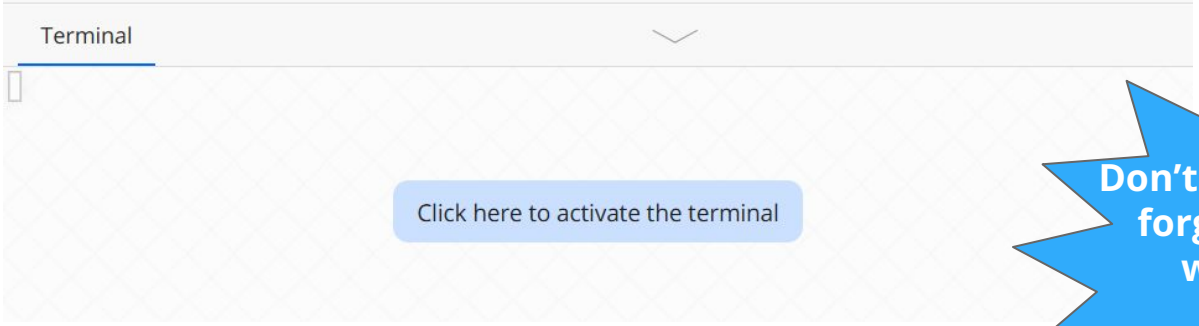
Running your code...

To run your code, click the button in the bottom left that says “Click here to activate the terminal”.

It should run automatically.

You can click the button again to rerun your code.

It should look like this;



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

Some shortcuts...

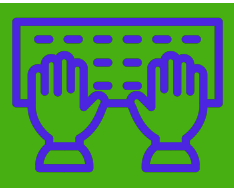
There are a couple things you can do to make copying your code from one page to another easier.

1) **Ctrl + A** Pressing these keys together will select all the text on a page

2) **Ctrl + C** Pressing these keys together will copy anything that's selected

3) **Ctrl + V** Pressing these keys together will paste anything you've copied

Need help with EdStem?



There is a section at the top of your workbook that explains how to use EdStem if you get stuck and need a reminder!

It's called 0: Intro to EdStem

Go to Part 0 and have a look!

Intro to Python

Let’s get coding!

Make a mistake!

In your first coding area. Type by **button mashing** the keyboard!

Then click the terminal to run it!

```
asdf asdjlkj;pa j;k4uroei
```

Did you get a big ugly error message?

Mistakes are great!

SyntaxError: Invalid Syntax

Good work you made an error!

ImportError: No module named humour

- Programmers make A LOT of errors!
- Errors give us hints to find mistakes
- Run your code often to get the hints!!
- Mistakes won't break computers!

TypeError: Can't convert 'int' object to str implicitly

```
AttributeError:
'NoneType' object
has no attribute
'foo'
```

KeyError: 'Hairy Potter'

We can learn from our mistakes!

Error messages help us fix our mistakes!

We read error messages from bottom to top

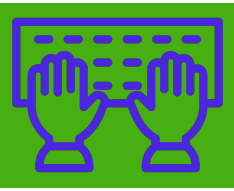
```
Traceback (most recent call last):
  File "C:/Users/Madeleine/Desktop/tmp.py", line 9, in <module>
    print("I have " + 5 + " apples")
TypeError: can only concatenate str (not "int") to str
```

3. Where that code is

1. What went wrong

2. What code didn't work

Write some code!!



1. Type the following into the “Playground” code window in chapter 1
2. Then run the code by clicking in the Terminal window

```
print('hello world')
```

Did it print:

```
hello world
```

???

Tell me more!

We can `print` on many lines at once!

```
>>> print("""Hello world.
This is me!
Life should be fun for everyone""")
```

Tell me more!

We can `print` on many lines at once!

```
>>> print("""Hello world.
This is me!
Life should be fun for everyone""")
Hello world.
This is me!
Life should be fun for everyone
```

Python the calculator!

Try writing some maths into python!

```
>>> 1 + 5
```

```
>>> 2 - 7
```

```
>>> 2 * 8
```

```
>>> 12/3
```

Python the calculator!

Try writing some maths into python!

```
>>> 1 + 5
6
>>> 2 - 7


>>> 2 * 8


>>> 12/3
```

Python the calculator!

Try writing some maths into python!

```
>>> 1 + 5
6
>>> 2 - 7
-5
>>> 2 * 8

>>> 12/3
```

Python the calculator!

Try writing some maths into python!

```
>>> 1 + 5
6
>>> 2 - 7
-5
>>> 2 * 8
16
>>> 12/3
```

Python the calculator!

Try writing some maths into python!

```
>>> 1 + 5
6
>>> 2 - 7
-5
>>> 2 * 8
16
>>> 12/3
4
```

A calculator for words!

What do you think these bits of code do?

Try them and see!

```
>>> "cat" + "dog"
```

```
>>> "tortoise" * 3
```

A calculator for words!

What do you think these bits of code do?
Try them and see!

```
>>> "cat" + "dog"
catdog

>>> "tortoise" * 3
```

A calculator for words!

What do you think these bits of code do?
Try them and see!

```
>>> "cat" + "dog"
catdog
```

```
>>> "tortoise" * 3
tortoisetortoisetortoise
```

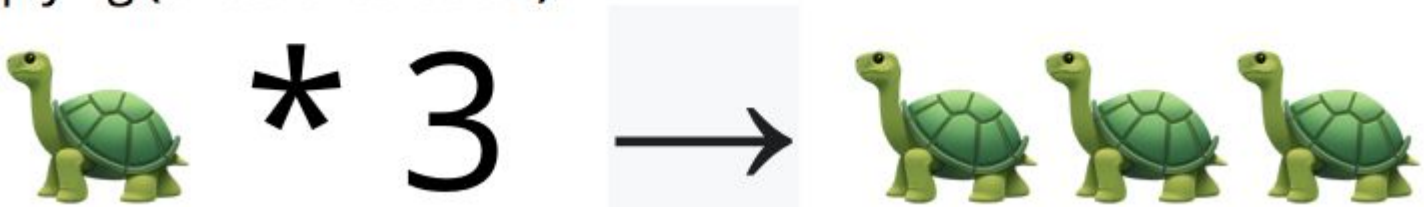
Strings!

`Strings` are things with `"quotes"`

To python they are essentially just a bunch of pictures!

Adding :

Multiplying (3 lots of tortoise!):



Strings and Ints!

Integers are numbers in python.

We can do maths with integers but not strings

```
>>> 5 + "5"
```

We can turn a string into an integer using int()

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

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

```
>>> 5 + "5"
TypeError: unsupported operand type(s) for +: 'int' and
'str'
```

We can turn a string into an integer using int()

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

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

```
>>> 5 + "5"
TypeError: unsupported operand type(s) for +: 'int' and
'str'
```

We can turn a string into an integer using int()

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

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

```
>>> 5 + "5"
TypeError: unsupported operand type(s) for +: 'int' and
'str'
```

We can turn a string into an integer using int()

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

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

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

A special string...

There are some combinations of characters in python that do some interesting things...

One of these character combinations is `"\n"`

Let's see what it does?

```
>> print("\n")
```

A special string...

There are some combinations of characters in python that do some interesting things...

One of these character combinations is `"\n"`

Let's see what it does?

```
>> print("\n")



>>
```

A special string...

There are some combinations of characters in python that do some interesting things...

One of these character combinations is `"\n"`

Let's see what it does?

```
>> print("\n")

>>
```

But nothing happened?

A special string...

Let's try something that will be easier to notice...

Let's see what it does?

```
>> print("Hello\nWorld")
```

A special string...

Let's try something that will be easier to notice...

Let's see what it does?

```
>> print("Hello\nWorld")
Hello
World
```

Typing \n in a string results in a new line!

Keeping organized with Comments!

Sometimes we want to write things in our file that the computer doesn't look at so we can write notes for later. We can use **comments** for that!

Sometimes we want to write a note for a people to read

```
# This code was written by Vivian
```

And sometimes we want to not run some code (but don't want to delete it!)

```
# print("Goodbye world!")
```

Try it!

1. Add a comment to your chatbot.py file in 1.1
2. Run your code to make sure it doesn't do anything extra!

Variables and Input

No Storing is Boring!

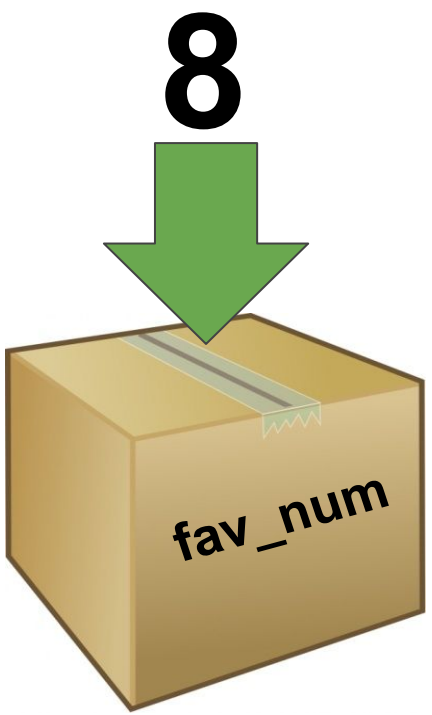
It's useful to be able to remember things for later!

Computers remember things in **"variables"**

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

Let's make our favourite number 8 today!

fav_num = 8



Variables

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


```
fav_num - 6
```

```
fav_num + 21
```

```
fav_num * 2
```

```
fav_num / 2
```

Variables

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


```
fav_num - 6
```

=> 2

```
fav_num + 21
```

```
fav_num * 2
```

```
fav_num / 2
```

Variables

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


```
fav_num - 6
```

=> 2

```
fav_num + 21
```

=> 29

```
fav_num * 2
```

```
fav_num / 2
```

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
```

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
```

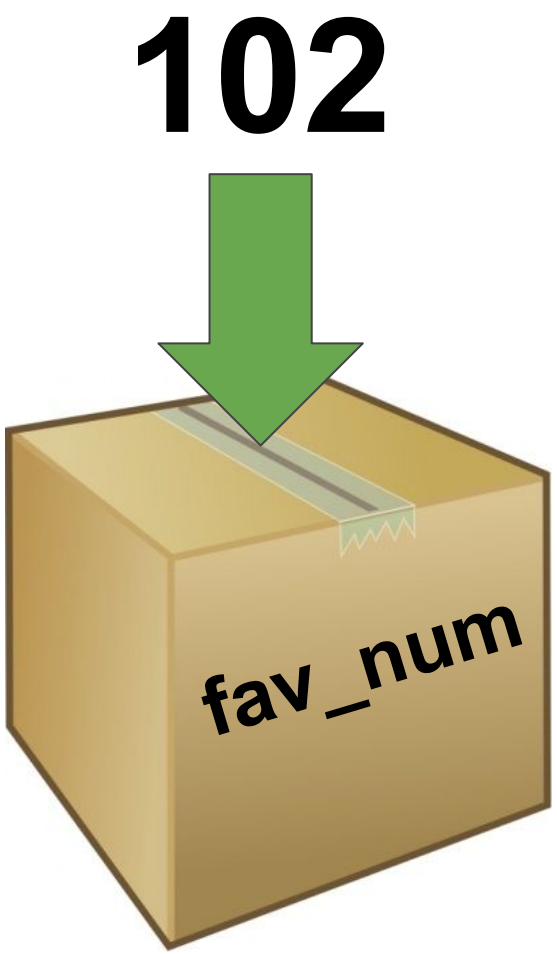
But writing 8 is much shorter than writing `fav_num`???

Variables

Variables are useful for storing things that change

(i.e. things that "vary" - hence the word "variable")

Try changing `fav_num` to **102**.



Variables

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


```
fav_num - 6
```

=> 96

```
fav_num + 21
```

=> 123

```
fav_num * 2?
```

=> 204

```
fav_num / 2?
```

=> 51

No variables VS using variables

4
Changes

1
Change

8 - 6	**102** - 6
8 * 2	**102** * 2
8 + 21	**102** + 21
8 / 2	**102** / 2

fav_num = **8**	fav_num = **102**
fav_num - 6	fav_num - 6
fav_num * 2	fav_num * 2
fav_num + 21	fav_num + 21
fav_num / 2	fav_num / 2

Reusing variables

We can replace values in variables:

```
animal = "dog"
print("My favourite animal is a " + animal)
animal = "cat"
print("My favourite animal is a " + animal)
animal = animal + "dog"
print("My favourite animal is a " + animal)
```

What will this output?

Reusing variables

We can replace values in variables:

```
animal = "dog"
print("My favourite animal is a " + animal)
animal = "cat"
print("My favourite animal is a " + animal)
animal = animal + "dog"
print("My favourite animal is a " + animal)
```

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

What can we store?

We can put any value in a variable:

```
apples = 5 + 5
print(apples)
apples = apples - 1
print(apples)
apples = "Delicious"
print(apples)
```

What will this output?

What can we store?

We can put any value in a variable:

```
apples = 5 + 5
print(apples)
apples = apples - 1
print(apples)
apples = "Delicious"
print(apples)
```

```
10
9
Delicious
```

Variables

Your turn!

Can you guess what each `print` will do?

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

>>> print(x + x)

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

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

Variables

Your turn!

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)
```

Variables

Your turn!

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)
```

Variables

Your turn!

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)
```

Variables

Your turn!

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
```

Switcharoo - Making copies!

Set some variables!

```
>>> x = 3
>>> y = x
>>> x = 5
```

What do x and y contain now?

Let's find out together!

Switcharoo - Making copies!

Set some variables!

```
>>> x = 3
>>> y = x
>>> x = 5
```

What do x and y contain now?

```
>>> x
5
>>> y
3
```

y hasn't changed because it has a copy of x in it!

Different data types!

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

Strings

Things in quotes used for storing text

```
"This is a string"
```

Ints

Whole numbers we can do maths with

```
a = 1
b = 2
print(a + b)
```

Floats

Decimal numbers for maths

```
a = 1.5
b = 2.0
print(a / b)
```

Booleans

For `True` and `False`

```
a = 5 > 3
boring = False
```

Asking a question!

It's more fun when we get to interact with the computer!

Try out this code to get the computer to ask you a question!

```
my_name = input('What is your name? ')
print('Hello ' + my_name)
```

What do you think happens?

Asking a question!

It's more fun when we get to interact with the computer!

Try out this code to get the computer to ask you a question!

```
my_name = input('What is your name? ')
print('Hello ' + my_name)
```

What do you think happens?

```
What is your name? Maddie
Hello Maddie
```

Asking a question!

Store the answer in the variable my_name

Writing input tells the computer to wait for a response

This is the question you want printed to the screen

```
my_name = input('What is your name? ')
print('Hello ' + my_name)
```

What do you think happens?

```
What is your name? Maddie
Hello Maddie
```

We can use the answer the user wrote that we then stored later!

Asking a question!

How would we ask somebody for their favourite type of cake?

How would we print their answer?

```
What cake do you like? chocolate
chocolate cake for you!
```

Asking a question!

How would we ask somebody for their favourite type of cake?

How would we print their answer?

```
flavour = input('What cake do you like? ')
```

```
What cake do you like? chocolate
chocolate cake for you!
```

Asking a question!

How would we ask somebody for their favourite type of cake?

How would we print their answer?

```
flavour = input('What cake do you like? ')
print(flavour + ' cake for you!')
```

```
What cake do you like? chocolate
chocolate cake for you!
```

Project time!

You now know all about variables & input!

Let's put what we learnt into our project

Try to do Part 1

The tutors will be around to help!

If Statements

Conditions!

Conditions let us make decisions.

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

Can you guess what these are?	
5 < 10	"Dog" == "dog"
3 + 2 == 5	"D" in "Dog"
5 != 5	"Q" not in "Cat"

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"
```

```
>>> 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"]
False "banana" in animals
True  "cat" in animals
```

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???

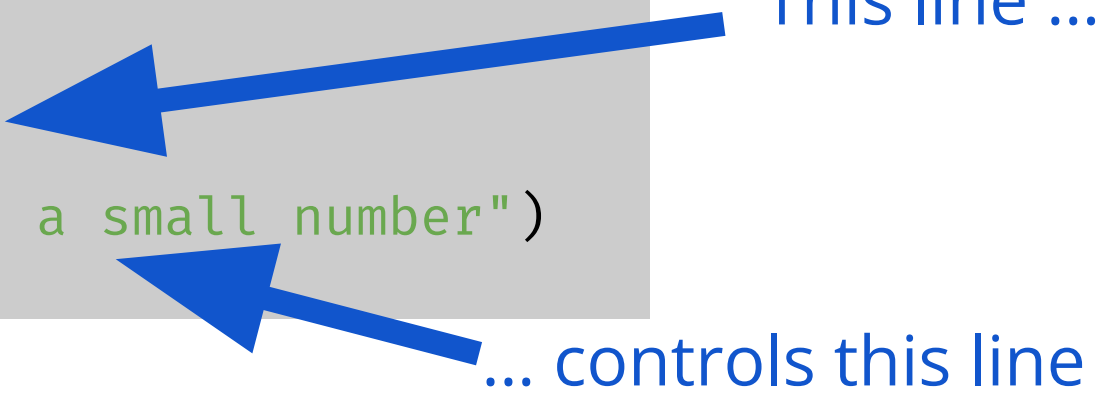
```
fave_num = 9000
if fave_num < 10:
    print("that's a small number")
```

What do you think happens?

```
>>>
```

If statements

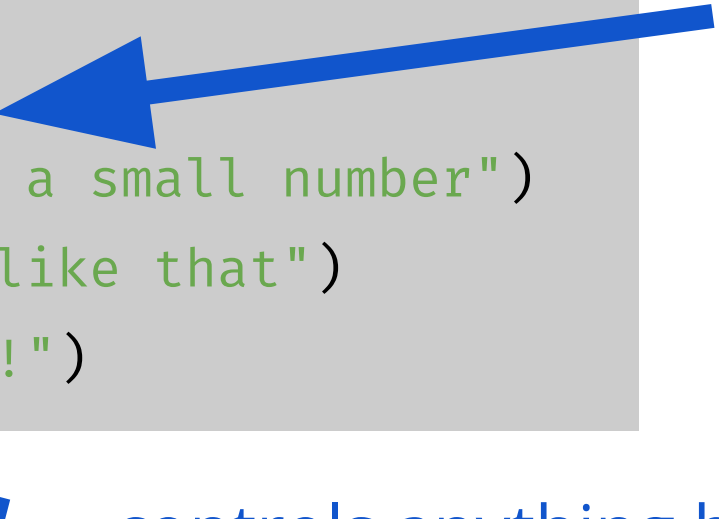
```
fave_num = 5
if fave_num < 10:
    print("that's a small number")
```



If statements

Actually .....

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



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

```
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 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!
```

Project Time!

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

See `if` you can do Part 2

The tutors will be around to help!

Files

Filing it away!

What if that data is too big to write in with the keyboard?

What happens if we want to use different data in our program?

We'd have to change the data inside our code!!

Instead, we can keep our data in a file and pick what file we want to use.

people.txt

```
Aleisha,brown,black,hat
Brittany,blue,red,glasses
Charlie,green,brown,glasses
Dave,blue,red,glasses
Eve,green,brown,glasses
Frankie,hazel,black,hat
George,brown,black,glasses
Hannah,brown,black,glasses
Isla,brown,brown,none
Jackie,hazel,blonde,hat
Kevin,brown,black,hat
Luka,blue,brown,none
```

Opening files!

To get access to the stuff inside a file in python we need to **open** it!

That doesn't mean clicking on the little icon!

```
f = open("test.txt", "r")
```

You'll now be able to read the things in `f`

If your file is in the same location as your code you can just use the name!

A missing file causes an error

Here we try to open a file that doesn't exist:

```
f = open("missing.txt", "r")
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
IOError: [Errno 2] No such file or
directory: 'missing.txt'
```

You can read a whole file into a string

```
>>> f = open("haiku.txt", "r")
>>> my_string = f.read()


>>> print(my_string)
Wanna go outside.
Oh NO! Help! I got outside!
Let me back inside!
```

haiku.txt

```
Wanna go outside.
Oh NO! Help! I got outside!
Let me back inside!
```

Write to files!

You can also write to files!

```
f = open("newfile.txt", "a")
f.write("This is my text!")
```

Notice we used `"a"` instead of `"r"`? We opened it in append mode!

This will create a new file if it doesn't exist, and add the text to the bottom of the file.

Closing Time

Always remember to close your file when you're finished with it:

```
f.close()
```

This will close your file and save it.

Project time!

Don’t `file` that knowledge away

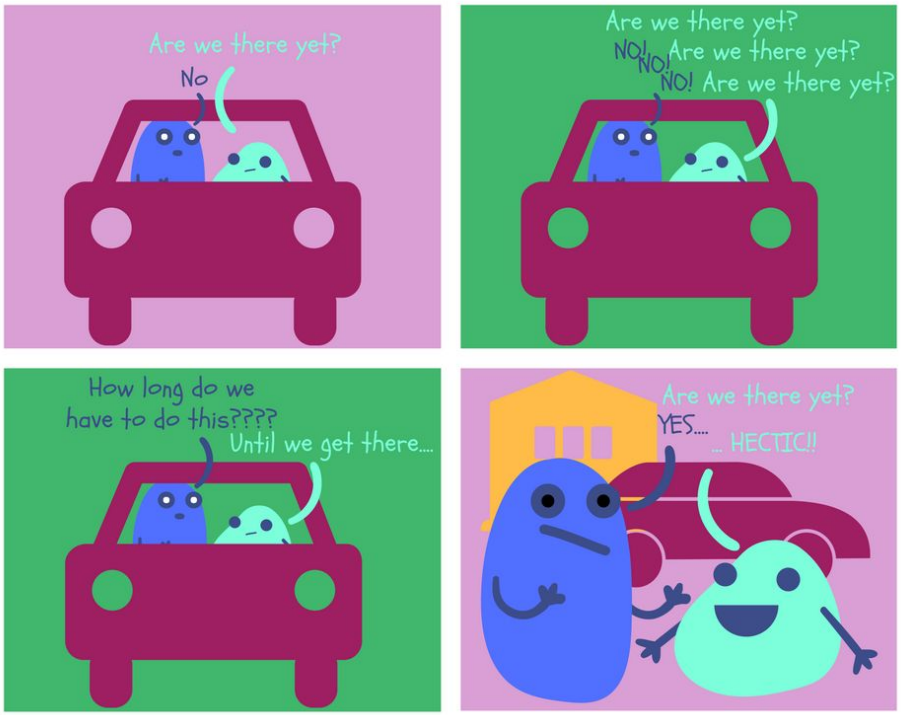
Use it in the next section of the project!

Try to do Part 3 - Part 4

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
>>>
```

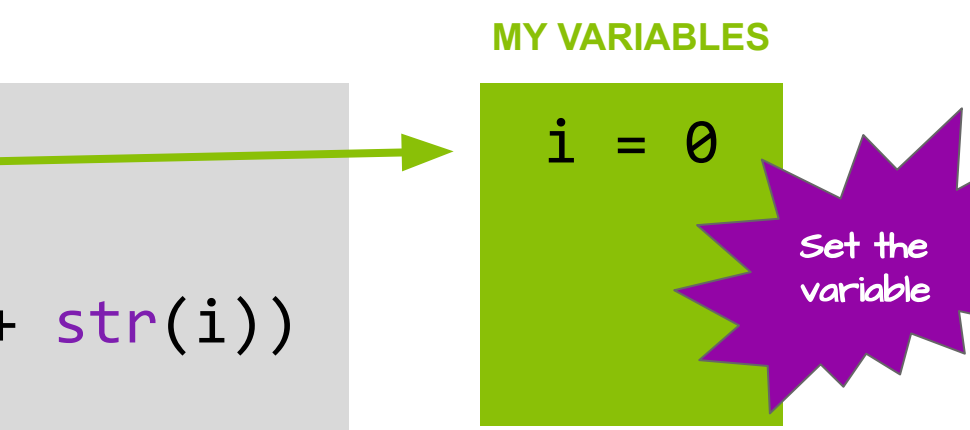
Introducing … `while` loops!

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
```



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
```

Introducing … `while` loops!

One step at a time!


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

MY VARIABLES

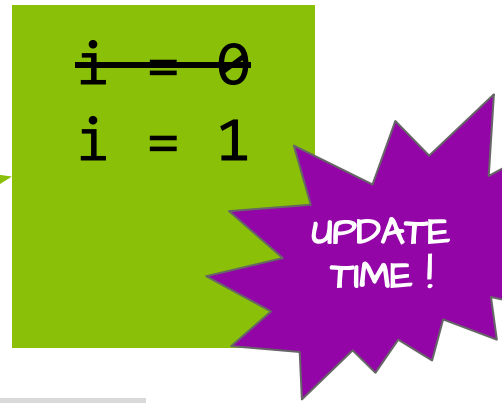
```
i = 0
```

```
i is 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
```

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!


```
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!


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

MY VARIABLES

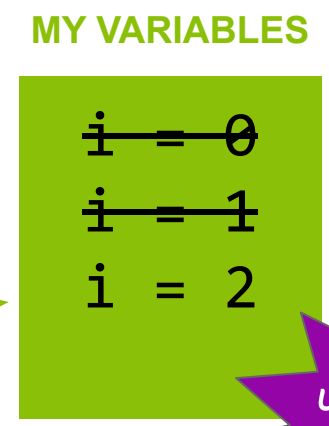
~~i = 0~~
i = 1

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

Introducing … `while` loops!

One step at a time!

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




```
i is 0
i is 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

```
i is 0
i is 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

```
i is 0
i is 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

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


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

MY VARIABLES

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

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

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
```

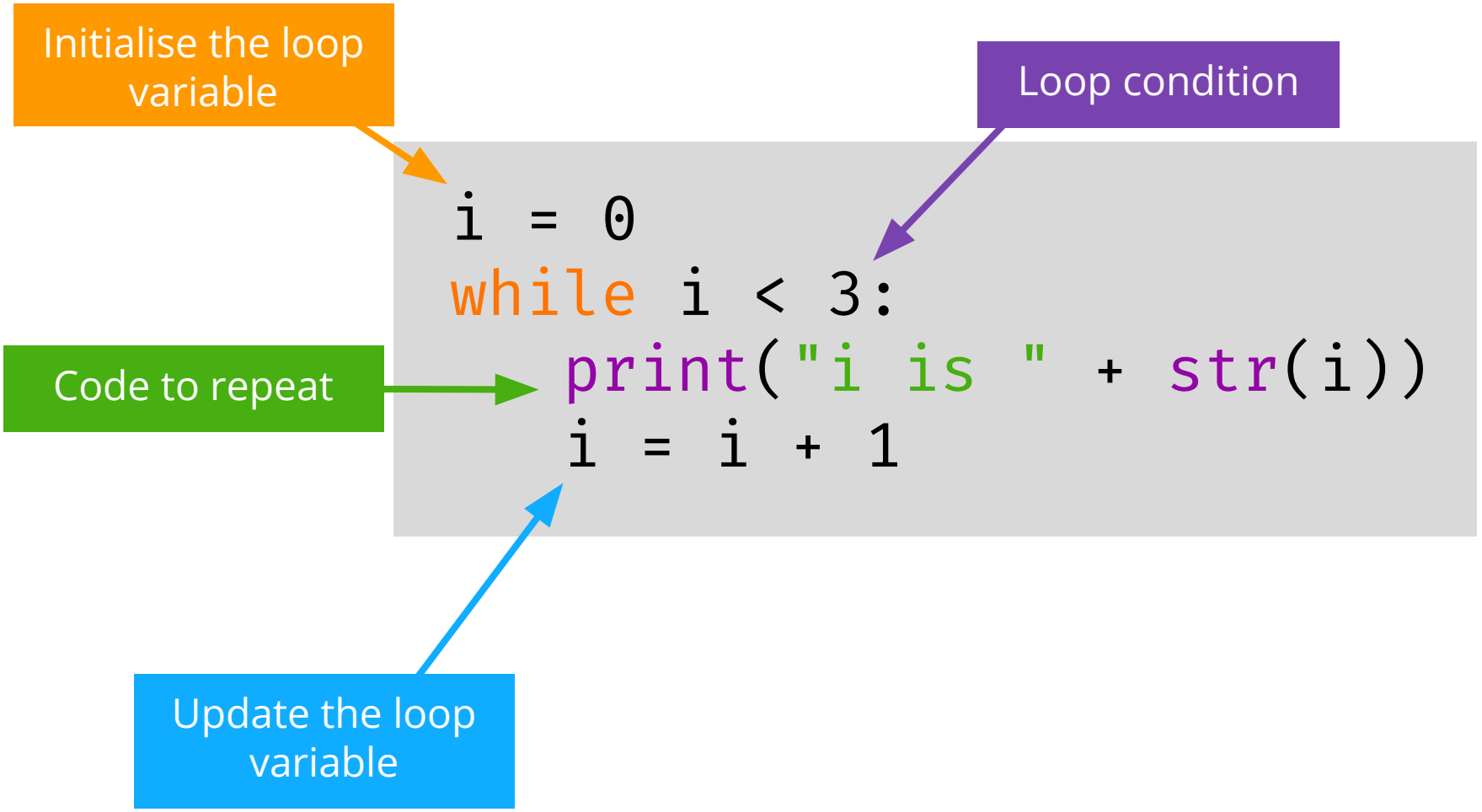
MY VARIABLES

~~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!



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))
```

```
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?")
```

Project Time!

while we're here:

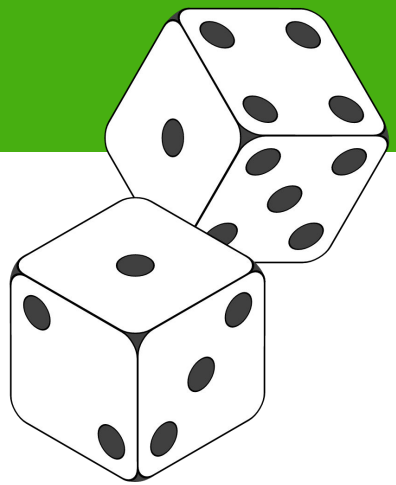
Try to do Part 5 and 6!

And extensions!

The tutors will be around to help!

Random!

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! 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 IDLE

```
>>> 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)
```

Getting a random number

Let's say you don't want to just pick a random thing from a list, but a random number between 0 and 100... how would you do that?

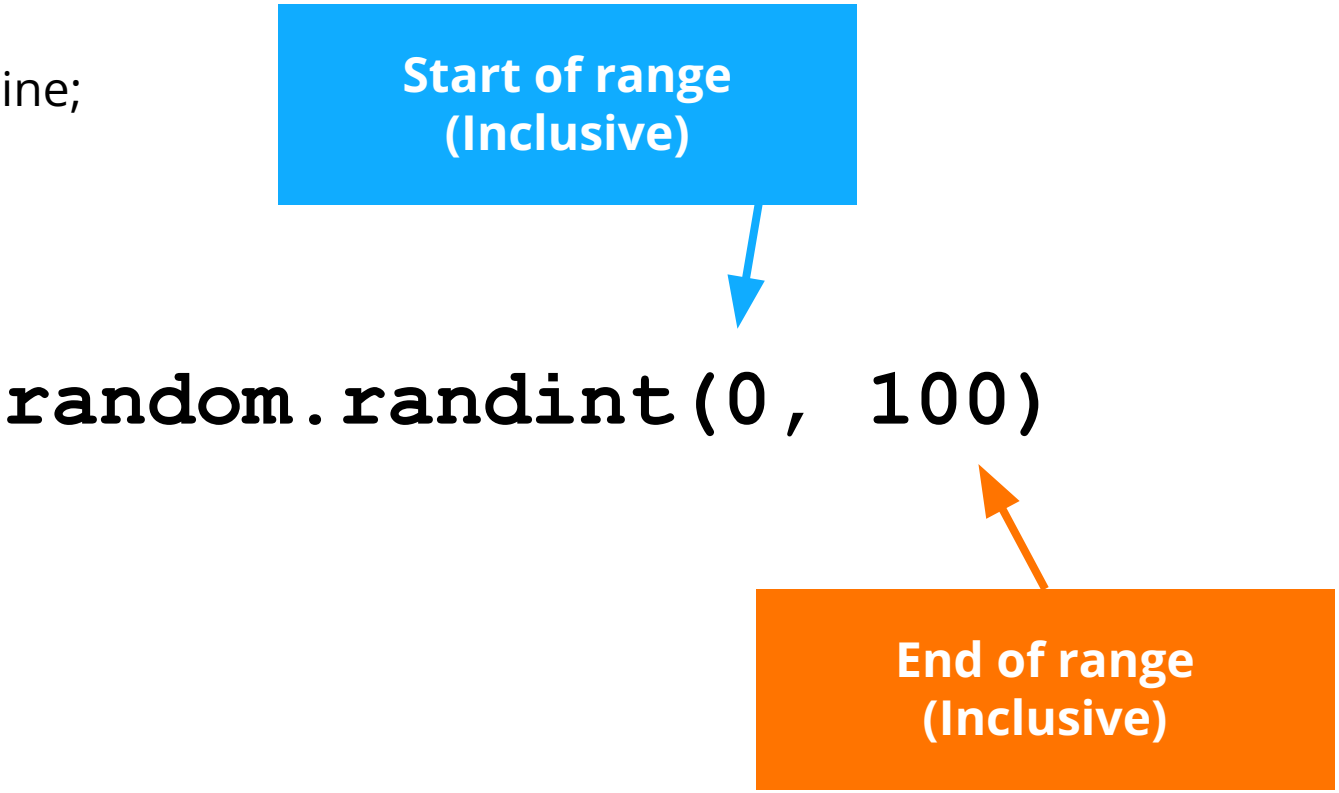
With this line;

```
random.randint(0, 100)
```

Getting a random number

Let's say you don't want to just pick a random thing from a list, but a random number between 0 and 100... how would you do that?

With this line;



Project Time!

Raaaaaaaaaandom! Can you handle that?

Let's try use it in our project!

Try to do the next Part

The tutors will be around to help!