Welcome to the Labs!

Tic Tac Toe

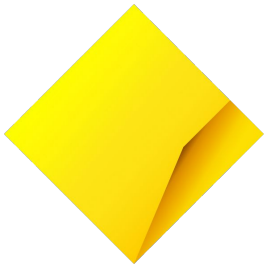
Thank you to our Sponsors!

Platinum Sponsor:

ATLASSIAN

amazon

Gold Sponsor:



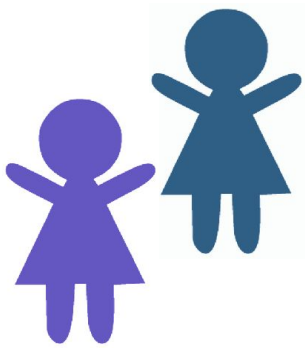
Commonwealth Bank

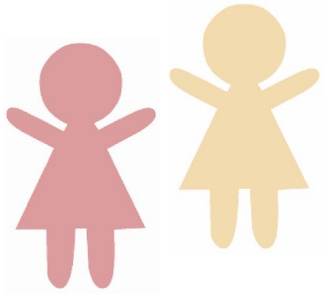
Who are the tutors?

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

Choose your location

You can see:

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

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

Tell us you're here!

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

Today's project!

Tic Tac Toe

What are we going to code today?

You're going to build a game of Tic Tac Toe that you can play with a friend

Each person will take turns to enter your symbol and a spot of the grid where you want to go

Your game will print the grid and tell you who won or if it was a draw

X		O
	X	O
		X

X wins!

It should look like this!

Players take turns to enter their symbol and the spot where they want to go

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 link: https://edstem.org/au/join/MhpTNF
2. Type 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.
7. Click on the one we will be using for this project: →

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

Getting to the lessons

1. Once you are in the course, you'll be taken to a discussion page.
2. Click the button for the lessons page (top right - looks like a book)

The set up of the workbook

The main page:

1. Heading at the top that tells you the project you are in

2. List of “Chapters” called something like **1:Welcome to Tic Tac Toe** They have an icon that looks like this:

3. To complete your project, we will work through the chapters one at a time starting with 1 and continuing on.

Inside a Chapter

Inside a chapter there are two main types of pages:

1. Lesson pages

   The lessons are where you will do your coding. They are called something like **1.1 Welcome** and have this icon: 

2. Checkpoints

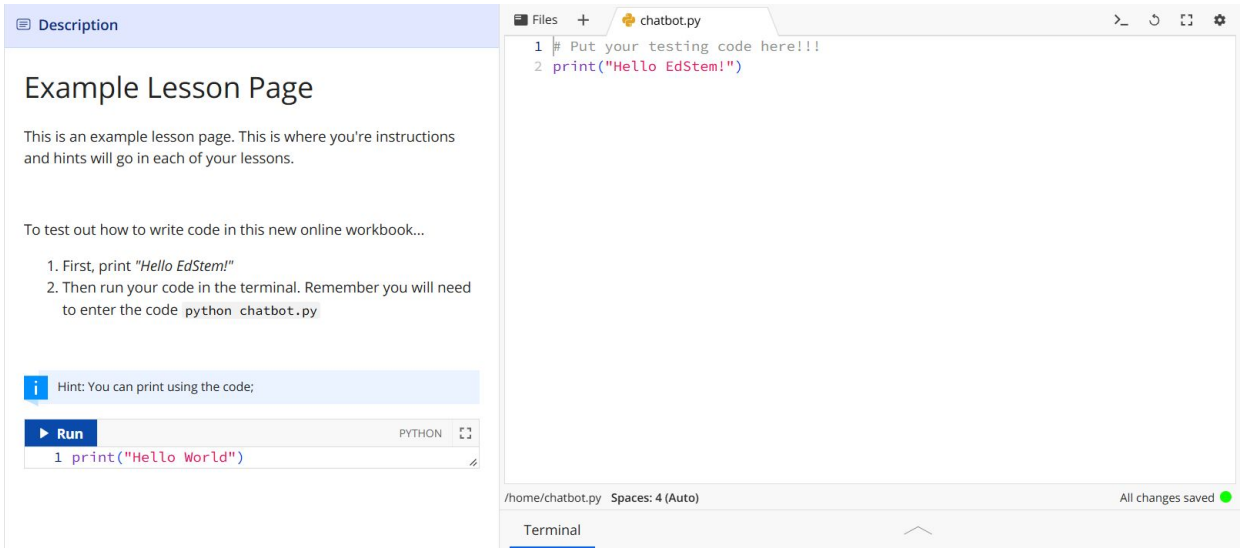
   Each chapter has a checkpoint. Complete the checkpoint to move to the next chapter. Make sure you scroll down to see all the questions listed. Checkpoints look like this: 

How to do the work

In each lesson there is:

1. A section on left with instructions for that lesson
2. 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 so that you can work towards finishing the project.

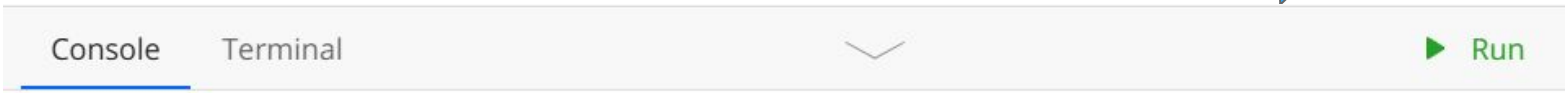


Running your code...

To run your code, you will need to click the button that says Run. It should run automatically and any outputs should be in the "Console" page. You can click the button again to rerun your code.

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

It should look like this;



Some shortcuts...

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

- **Ctrl + A** Pressing these keys together will select all the text on a page
- **Ctrl + C** Pressing these keys together will copy anything that's selected
- **Ctrl + V** Pressing these keys together will paste anything you've copied

On Macs use Command (⌘) instead of Ctrl

Project time!

You now know all about the EdStem!

You should now sign up and join our EdStem class. You should also have a look at part 0 of your workbook

Remember the tutors will be around to help!

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!

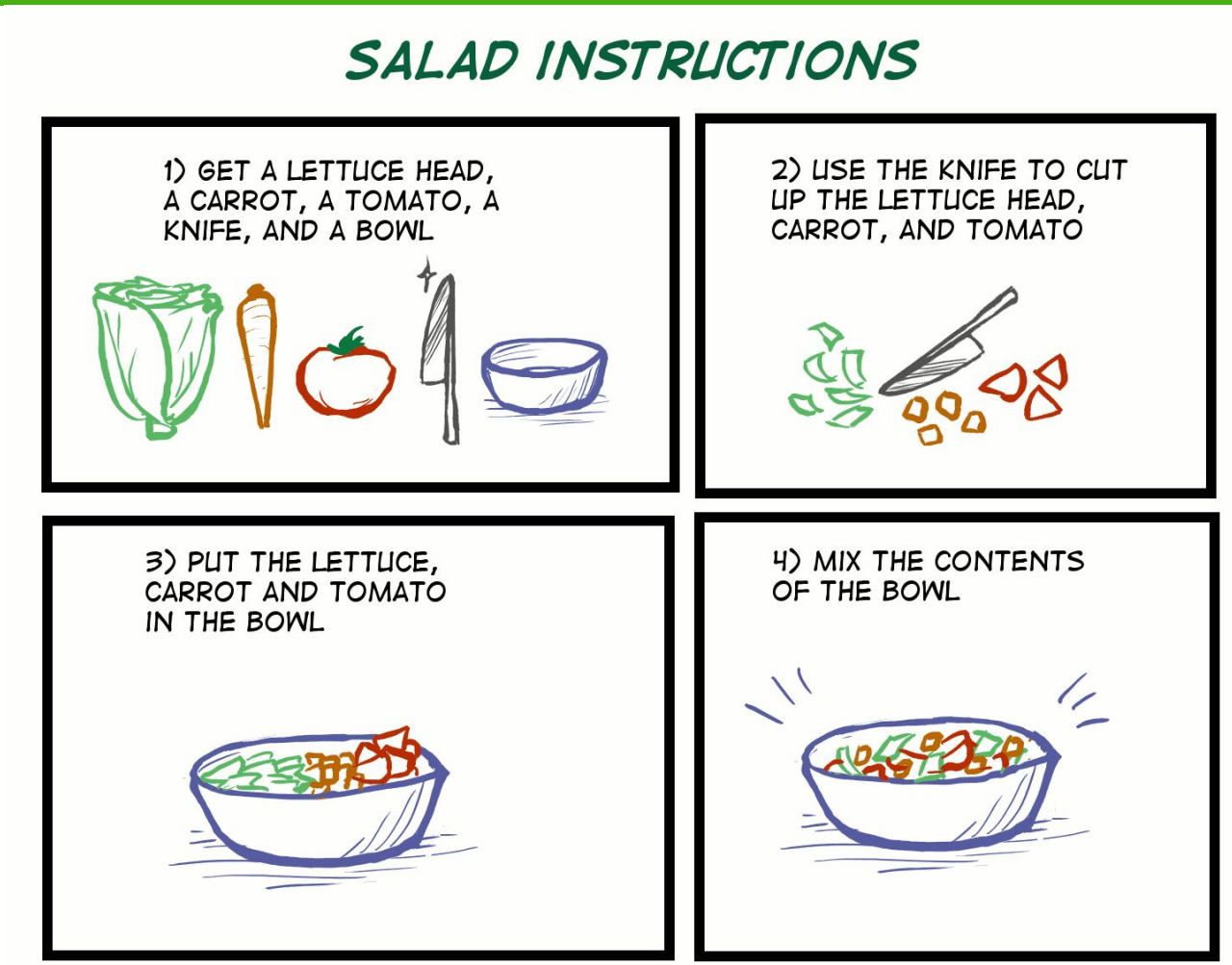
Programming is a language to talk to computers

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!

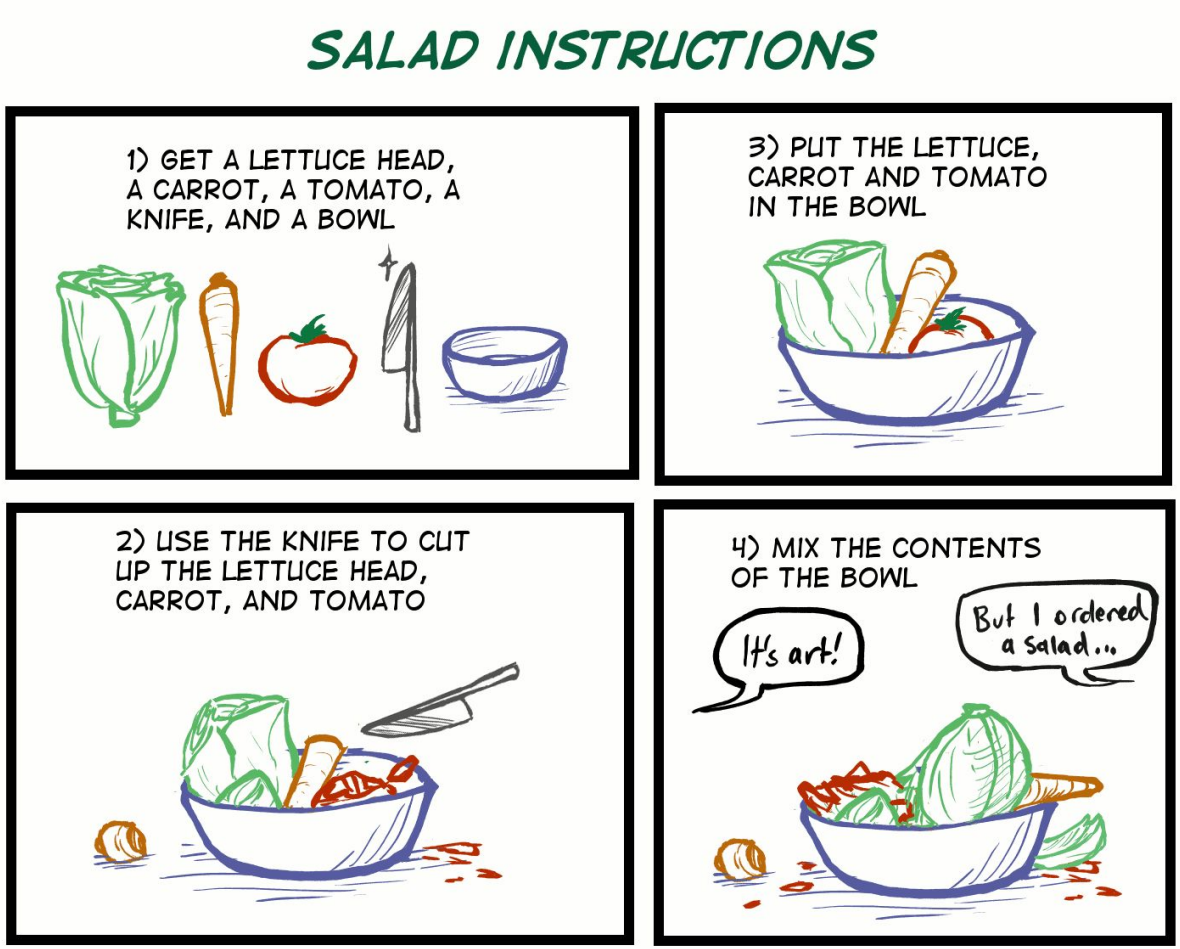


People are smart! Computers are dumb!

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

A computer wouldn't know this recipe was wrong!

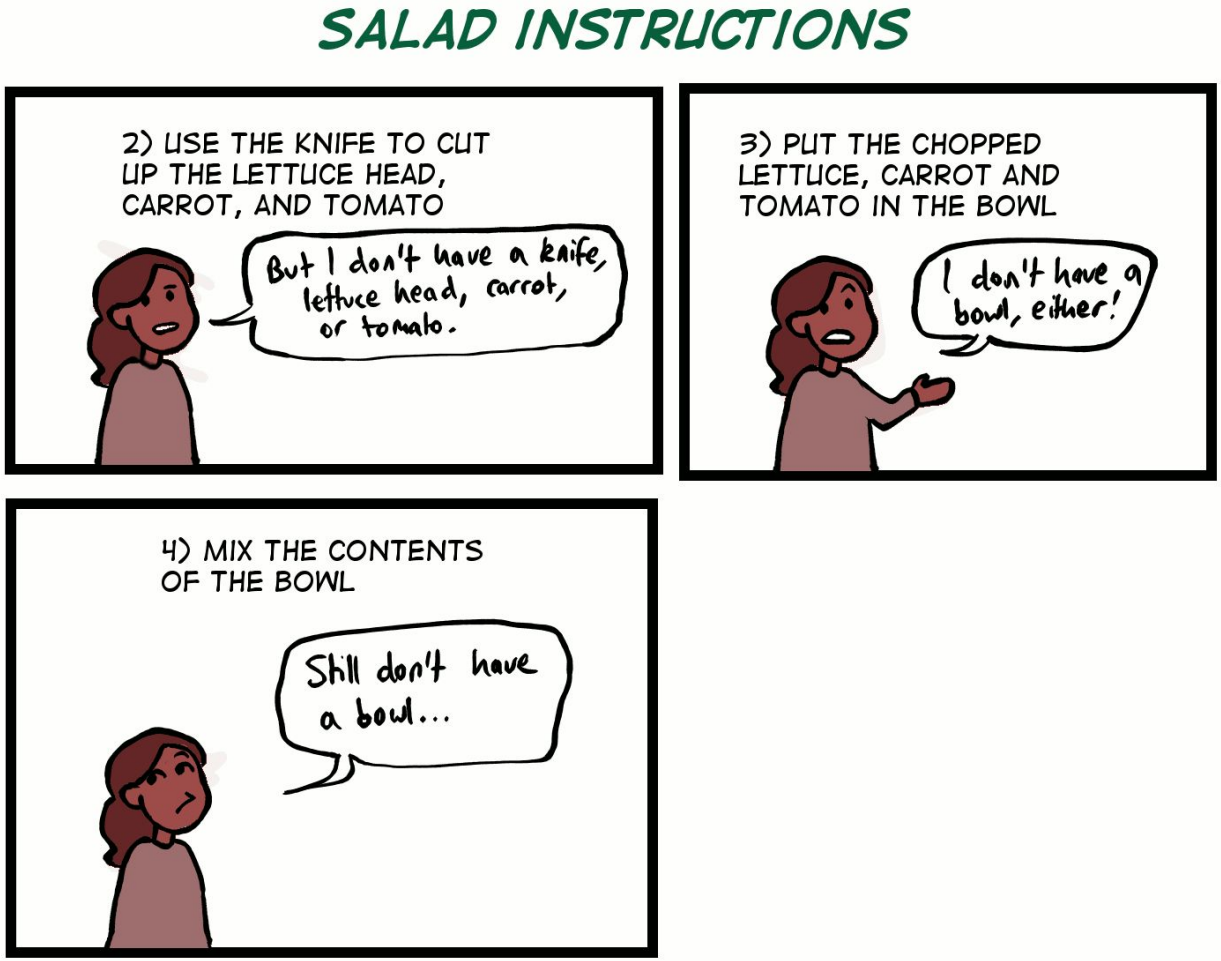
It would still try to make it anyway.



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!



Everyone & Everything has strengths!

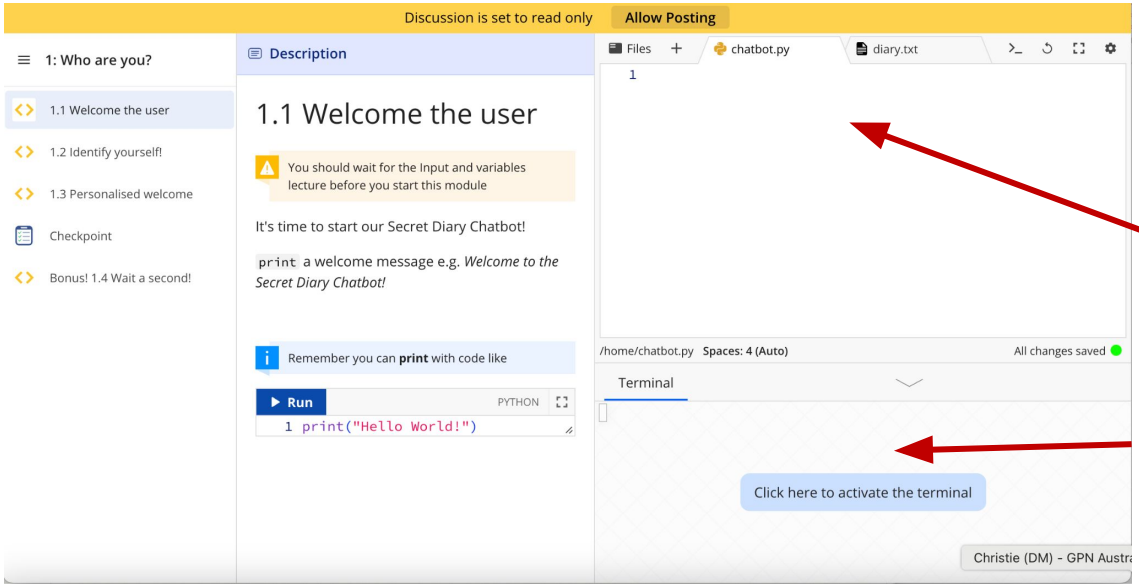
- Understand instructions despite:
  - Spelling mistakes
  - Typos
  - Confusing parts
- Solve problems
- Tell computers what to do
- Get smarter every day

- Does exactly what you tell it
- Does it the same every time
- Doesn't need to sleep
- Will work for hours on end
- Doesn't get bored
- Really really fast
- Get smarter when you tell it how

Intro to Python

Let’s get coding!

Let’s make a mistake!



1. Type by **button mashing** the keyboard here e.g.

   `ks@674dbkjSDfkl`

2. **Click** in the Terminal panel here to run your code!

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!

KeyError: 'Hairy Potter'

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

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

We can learn from our mistakes!

```
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
```

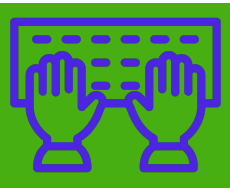
1. What went wrong

2. Which bit of code didn't work

3. Where that code is

We read error messages from bottom to top.

Write some code!!



1. Type the following into the code window
2. Then run the code by clicking in the Terminal window

```
print("hello world")
```

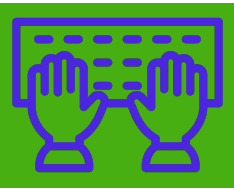
Did it print:

```
hello world
```

???

Print another message - remember quotes at the start & end

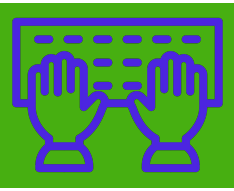
Python the calculator!



Try writing some maths into python! After typing each line, test it out by clicking in the Terminal window.

- `print(1 + 5)`
- `print(2 - 7)`
- `print(2 * 8)`
- `print(12 / 3)`

Python the calculator!

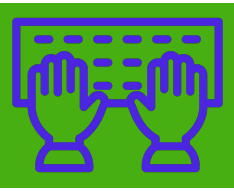


Try writing some maths into python! After typing each line, test it out by clicking in the Terminal window.

- `print(1 + 5)`

  `6`
- `print(2 - 7)`
- `print(2 * 8)`
- `print(12 / 3)`

Python the calculator!



Try writing some maths into python! After typing each line, test it out by clicking in the Terminal window.

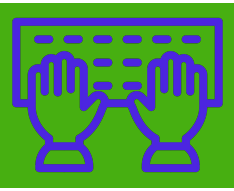
- `print(1 + 5)`

  `6`
- `print(2 - 7)`

  `-5`
- `print(2 * 8)`

- `print(12 / 3)`

Python the calculator!



Try writing some maths into python! After typing each line, test it out by clicking in the Terminal window.

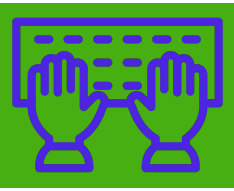
- `print(1 + 5)`

  `6`
- `print(2 - 7)`

  `-5`
- `print(2 * 8)`

  `16`
- `print(12 / 3)`

Python the calculator!



Try writing some maths into python! After typing each example, run by clicking in the Terminal window.

- `print(1 + 5)`

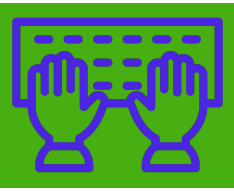
  `6`
- `print(2 - 7)`

  `-5`
- `print(2 * 8)`

  `16`
- `print(12 / 3)`

  `4`

A calculator for words!

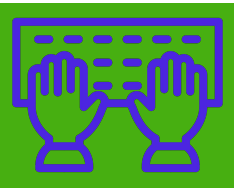


What do you think these bits of code do?
Try them! Run the code after typing in each example.

- `print("cat" + "dog")`

- `print("tortoise" * 3)`

A calculator for words!

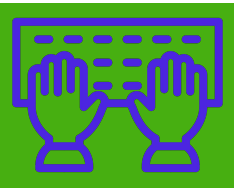


What do you think these bits of code do?
Try them! Run the code after typing in each example.

- ```
  print("cat" + "dog")
  ```

  ```
  catdog
  ```
- ```
  print("tortoise" * 3)
  ```

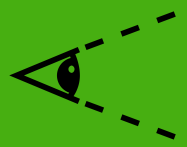
A calculator for words!



What do you think these bits of code do?
Try them! Run the code after typing in each example.

- ```
  print("cat" + "dog")
  ```
  ```
  catdog
  ```
- ```
  print("tortoise" * 3)
  ```
  ```
  tortoisetortoisetortoise
  ```

Strings!



Strings are things with **"quotes"**

Strings can have any letters in them, even spaces!

```
"Hello, world!"
                                "bla bla bla"
":)"        " "
                          'I can use single quotes too!'
          "¯\_(ツ)_/¯"
                                  "asdfghjklqwertyuiopzxcvbnm"
"DOGS ARE AWESOME!"
                        "!@#$%^&*()_+-=[]|\:;'<>,./?"
```

Adding a comment!

Sometimes we want to write things in our file that the computer doesn't look at. 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!")
```

Variables

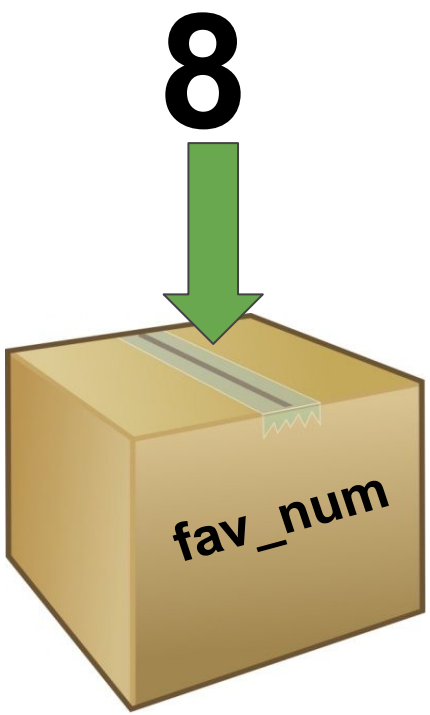
No Storing is Boring!

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

Variables are like putting things into a labeled cardboard box.

Let's make our favourite number 8!

fav_num = 8



Variables

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

1. `fav_num - 6`
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`

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`

   4

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`
   `4`

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`
   `4`

3. `fav_num + 21`
   `29`

4. `fav_num / 2`
   `4`

Variables

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

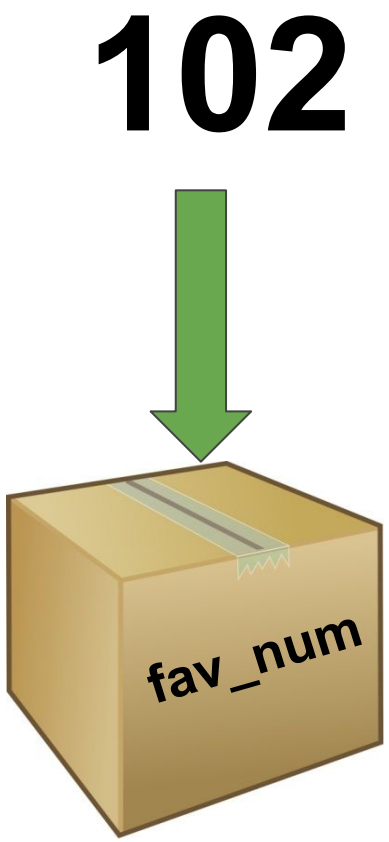
But writing 8 is so much easier than writing fav_num! 🤔

Let's talk about why variables are SO much better!

Variables

Variables are useful for storing things that change
(i.e. things that "vary" - hence the word "variable")

What would happen if we changed `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?
```

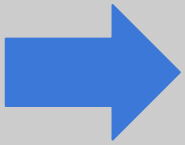
=> 51

No variables VS using variables

4
Changes

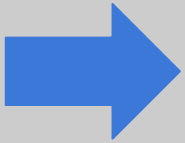
1
Change

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



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

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



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

Reusing variables

We can replace values in variables

We can print variables

We use + to print a “string” combined with a variable

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

What will this output?

Reusing variables

We can replace values in variables

We can print variables

We use + to print a “string” combined with a variable

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

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

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)
```

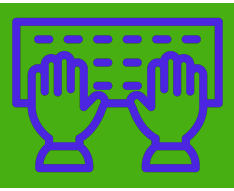
Variables

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
```

Project time!

You now know all about the building blocks of Python!

Let's put what we learnt into our project

Try to do Part 1!

The tutors will be around to help!

Inputs

Asking a question!

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

Let's get the computer to ask us 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!

Let's get the computer to ask us 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 Input!

Let's put what we learnt into our project

Try to do the Part 2!

The tutors will be around to help!

If Statements

Conditions!

Conditions let us make decision.

First we test if the condition is met!

Then maybe we'll do the thing

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

Yep it's raining

..... take an umbrella

Booleans (True and False)

Computers store whether a condition is met in the form of

`True` and `False`

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

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

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???

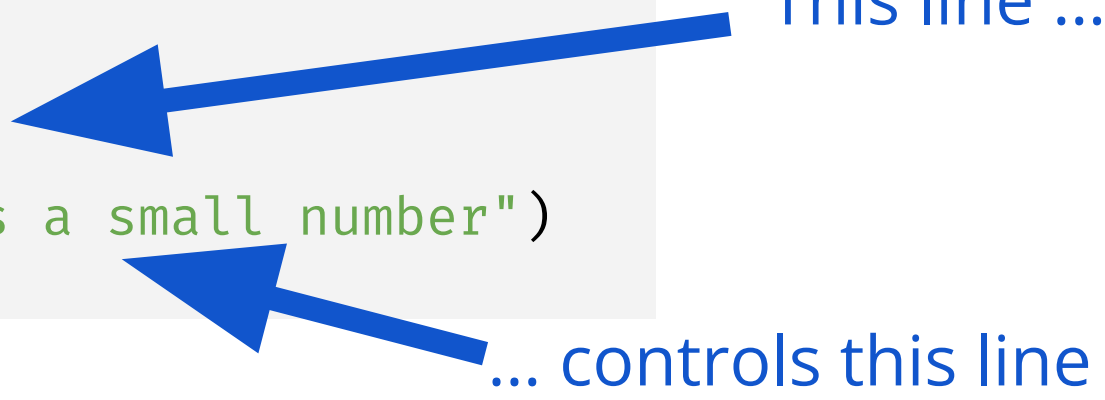
```
fave_num = 9000
if fave_num < 10:
    print("that's a small number")
```

What do you think happens?

```
>>>
```

If statements

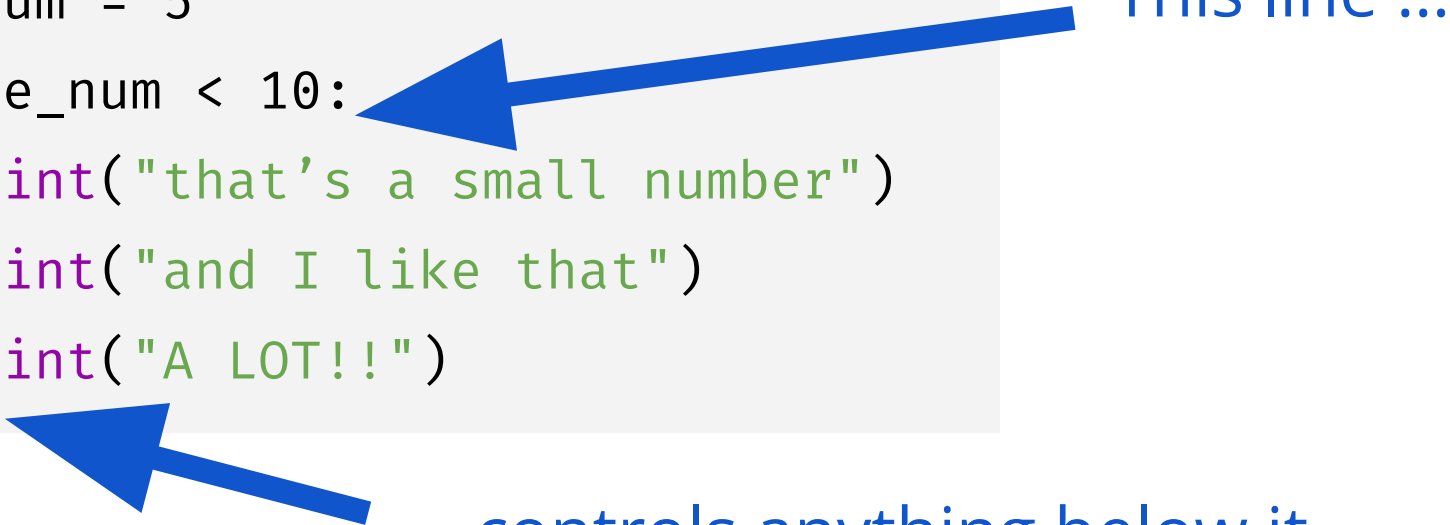
```
fave_num = 5
if fave_num < 10:
    print("that's a small number")
```



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?

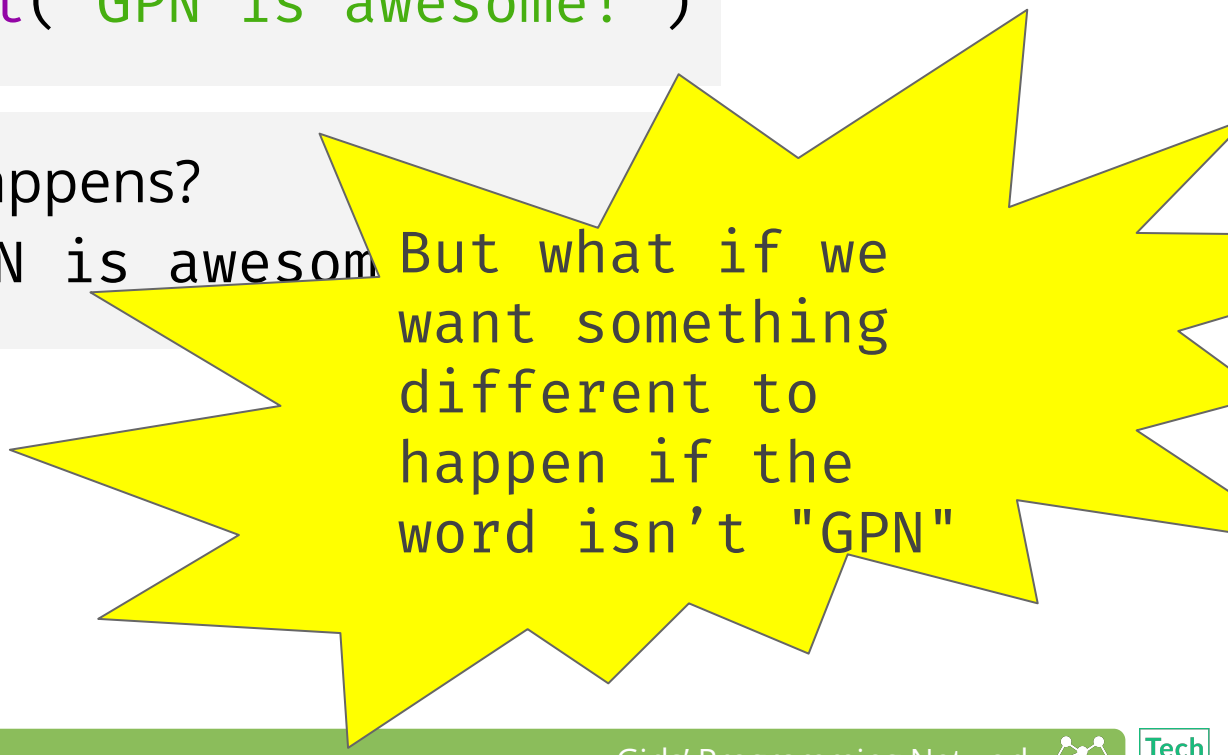
```
>>> GPN is awesome!
```

If statements

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

What happens?

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



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

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

```
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

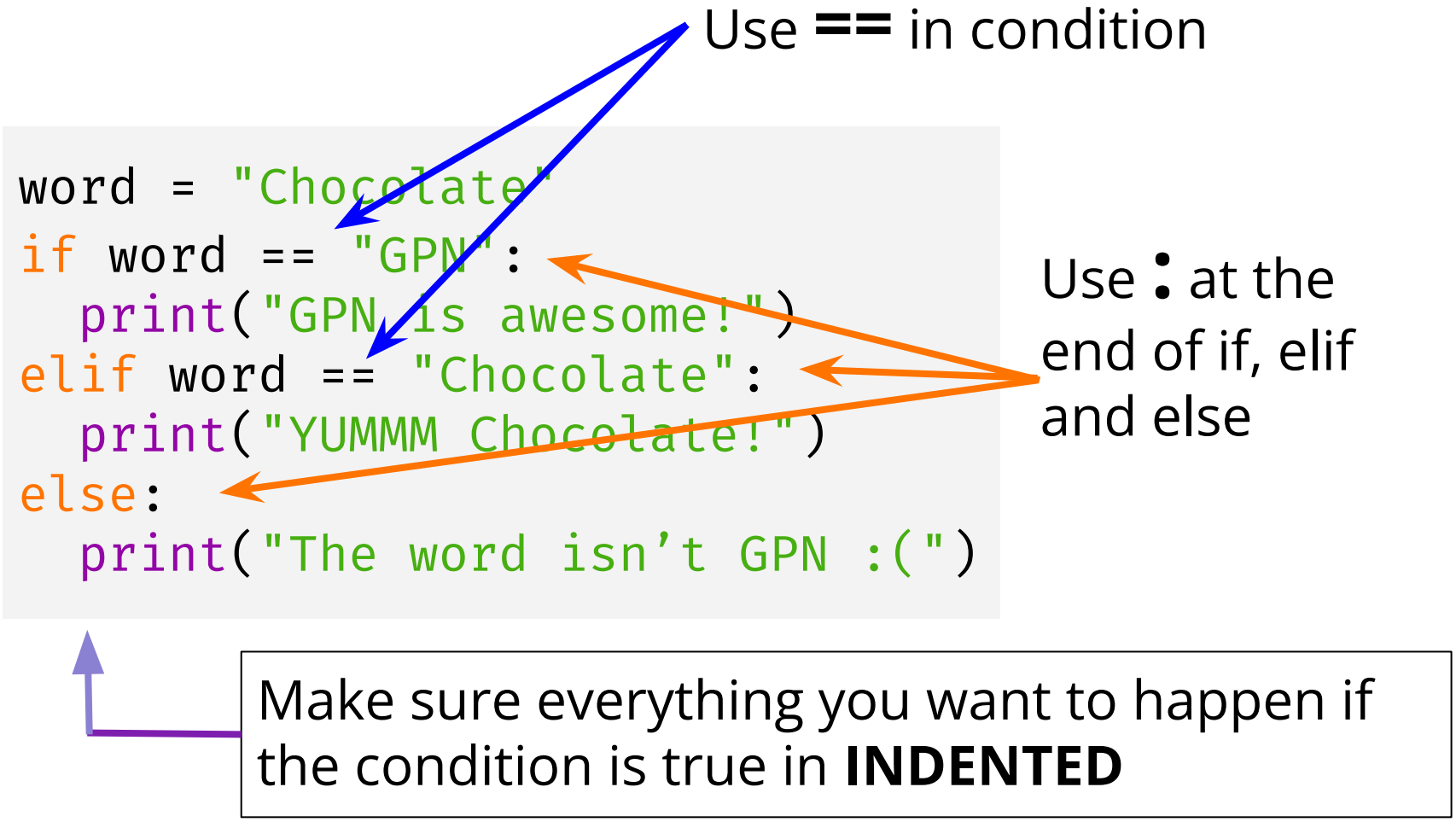
`else` statements means something still happens if the **`if`** statement was **`False`**

```
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!
```

Things to watch out for in if statements!



Project Time!

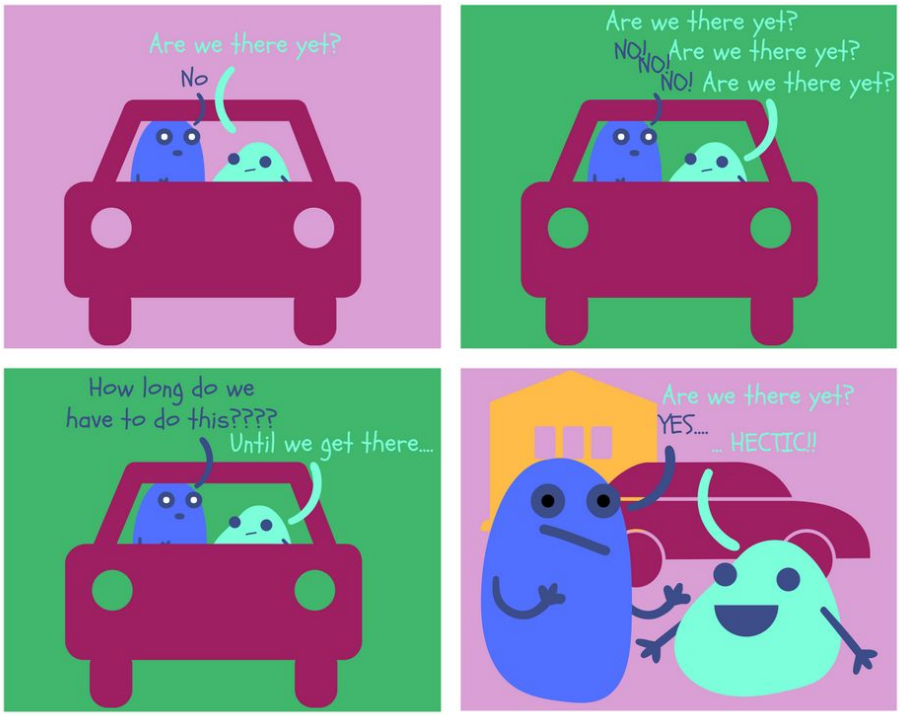
You now know all about `if, elif and else`!

See `if` you can do the Part 3

`else` 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
>>>
```

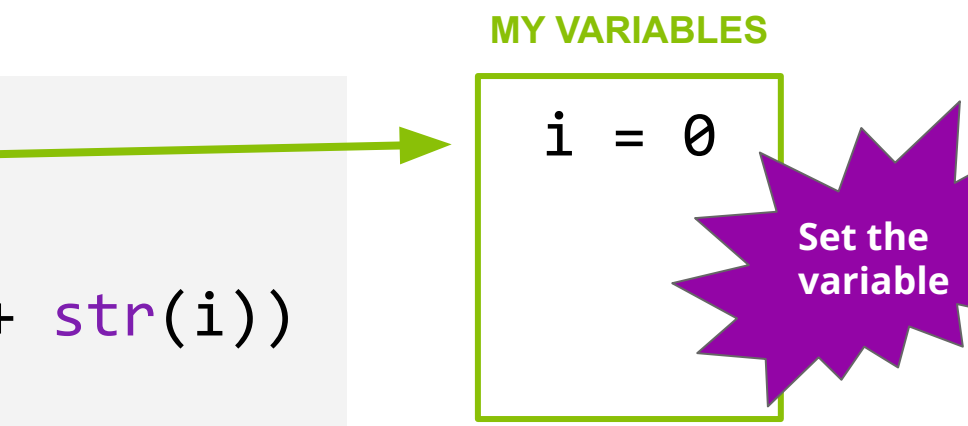
Introducing … `while` loops!

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!

0 is less than 3 !

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

MY VARIABLES

```
i = 0
```

Introducing ... `while` loops!

One step at a time!


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

MY VARIABLES

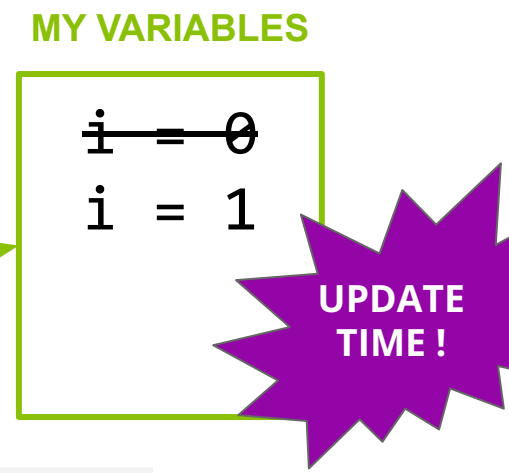
```
i = 0
```

```
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

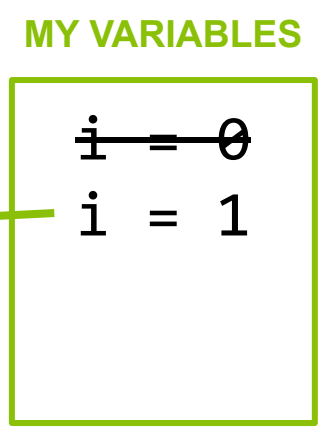
```
i is 0
```

Introducing … `while` loops!

One step at a time!

1 is less than 3 !

```
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

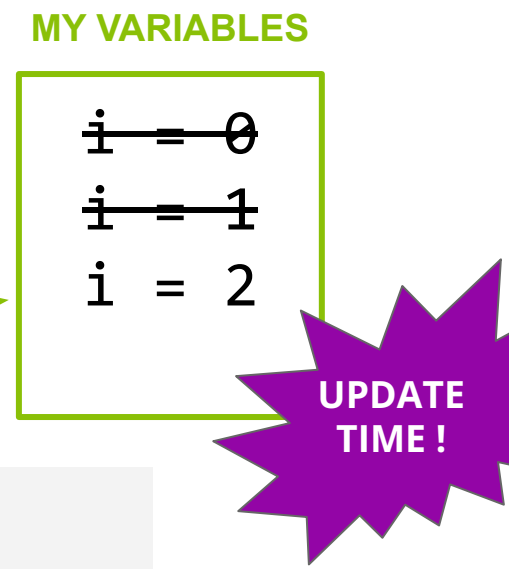
~~i = 0~~
i = 1

```
i is 0
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!

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!


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

MY VARIABLES

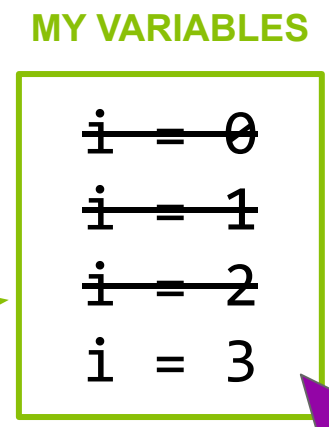
~~i = 0~~
~~i = 1~~
i = 2

```
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
```




```
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
```

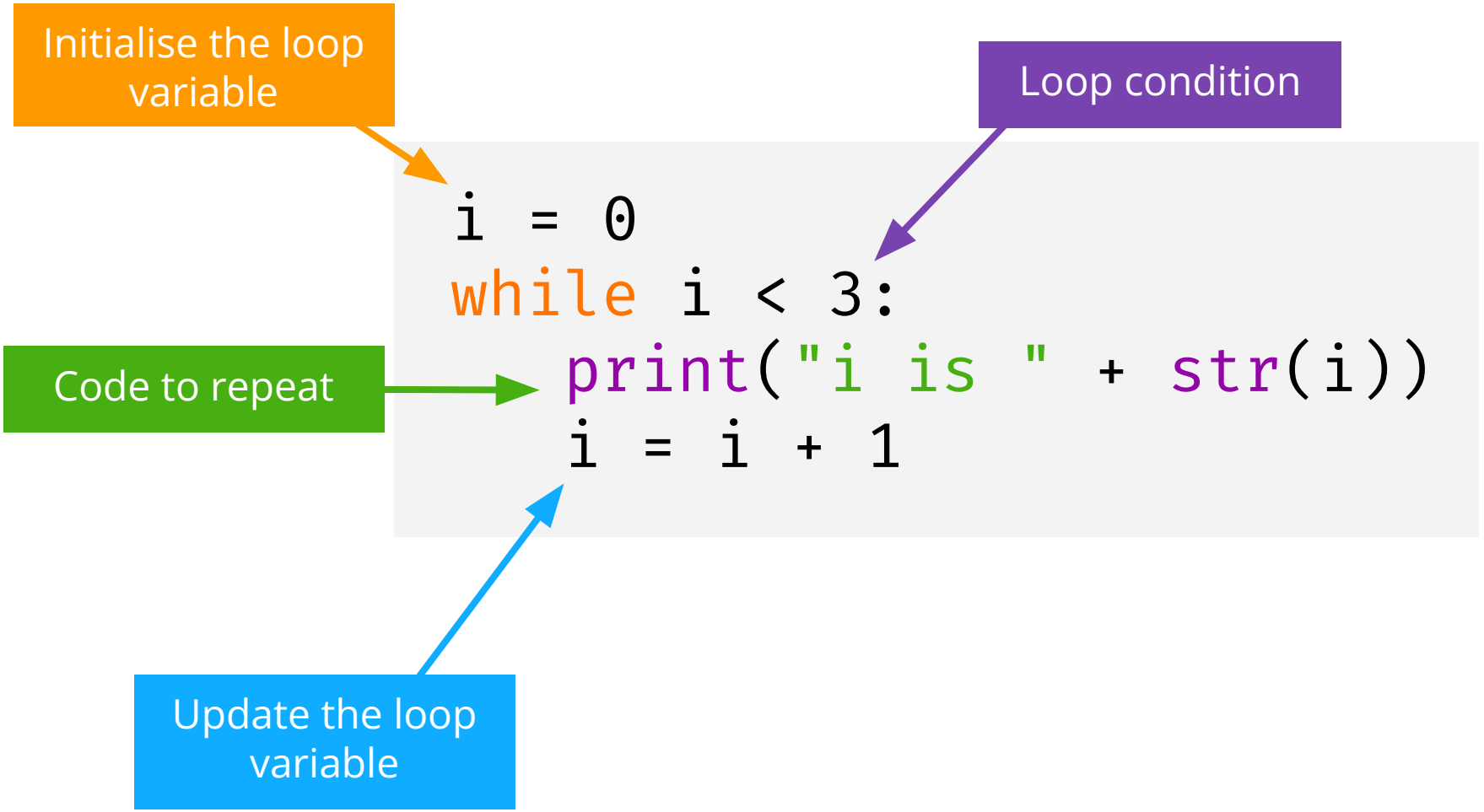
We are are done with this loop!

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

MY VARIABLES

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

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?")
```

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?")
```

```
Are we there yet?
Are we there yet?
Are we there yet?
Are we there yet?
Are we there yet?
Are we there yet?
```

Give me a break!

What if I want to stop looping?

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

What if I want to skip the rest of the code in the loop body and start the 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:

Try to do the Part 4!

The tutors will be around to help!

Complex Logic

Simple Conditions!

We've learned about simple conditions like this one before.

They're really useful when you only want something to happen sometimes.

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

Complex Conditions!

But what if you want to only take an umbrella if it's raining and you're going outside?

You might do it like this:

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

Complex Conditions!

But what if you want to only take an umbrella if it's raining and you're going outside?

You might do it like this:

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

But that starts to get messy quickly.

AND

Instead you can do it like this!

```
weather = "raining"
location = "outside"
if weather == "raining" and location == "outside":
  print("Take an umbrella!")
```

This is easier to read and stops things getting messy, especially if you have lots of conditions to check.

NOT EQUAL

What if we want to test if something is NOT EQUAL

For that we'll use the NOT EQUAL symbol **!=**

!= means NOT EQUAL in computer speak

NOT EQUAL

What if we want to test if something is NOT EQUAL

For that we'll use the NOT EQUAL symbol **!=**

!= means NOT EQUAL in computer speak

Let's try this!

```
secret_number = 10

my_guess =input("Enter your guess: ")
if my_guess != secret_number:
   print("You're wrong")
```

NOT EQUAL

What if we want to test if something is NOT EQUAL

For that we'll use the NOT EQUAL symbol **!=**

!= means NOT EQUAL in computer speak

Let's try this!

```
secret_number = 10

my_guess =input("Enter your guess: ")
if my_guess != secret_number:
   print("You're wrong")
```

```
Enter your guess: 2
You're wrong
```

Project time!

That’s all very logical

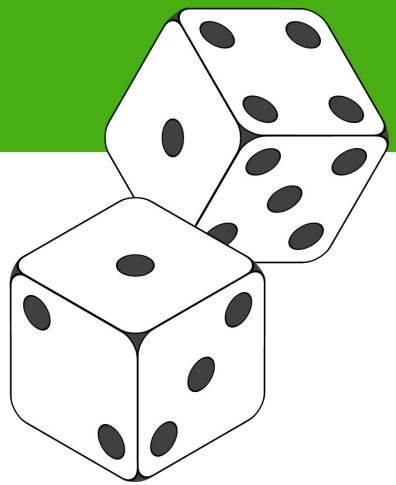
Let’s put what we learnt into our project

Try to do Part 5 and the 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 your Repl It ***Console*** (black box side)

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

3. Choose randomly! Try it a few times!

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

Using the random module

You can also assign your random choice to a variable

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

Project Time!

Raaaaaaaaaandom! Can you handle that?

Let's try use it in our project!

Try to do the next Part

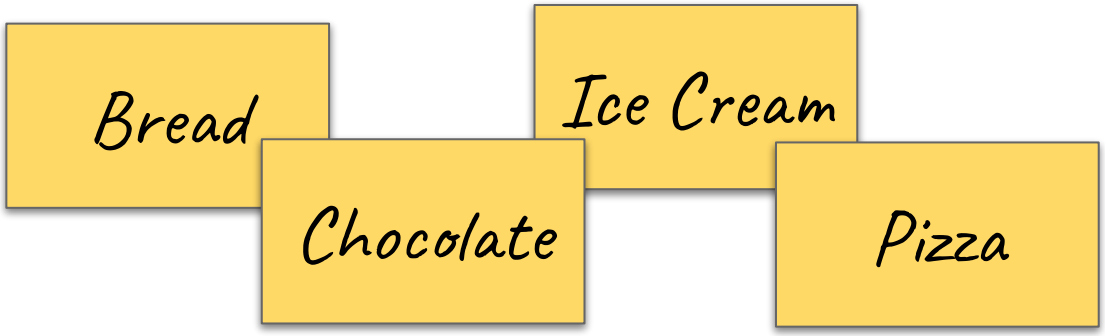
The tutors will be around to help!

Lists

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!

- Bread
- Chocolate
- Ice Cream
- Pizza

Lists

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

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

So much repetition!

Instead we use a python list!

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

You can put (almost) anything into a list

- You can have a list of integers

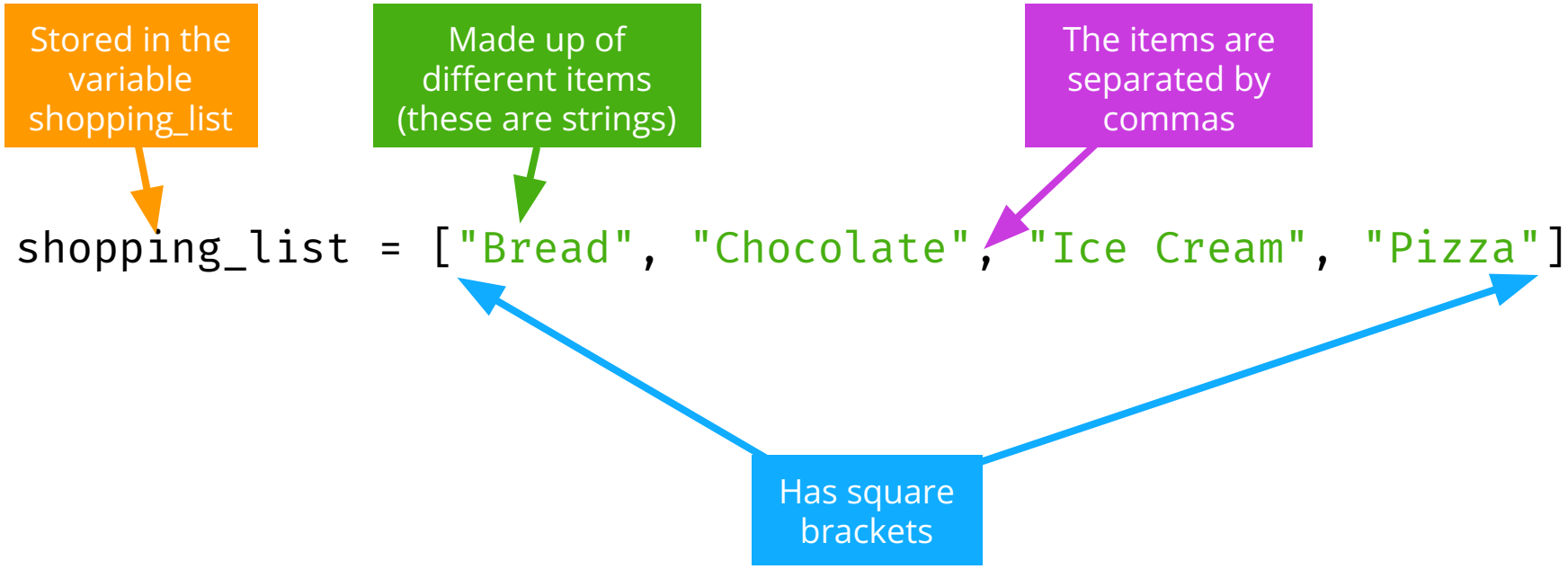
```
>>> primes = [1, 2, 3, 5, 11]
```

- You can have lists with mixed integers and strings

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

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

List anatomy



Removing items!

We can remove items from the list if they're no longer needed!

What if we decided that we didn't like butterflies anymore?

```
>>> faves
['books', 'butterfly', 'lollipops', 'skateboard']
>>> faves.remove('butterfly')
```

What does this list look like now?

Removing items!

We can remove items from the list if they're no longer needed!

What if we decided that we didn't like butterflies anymore?

```
>>> faves
['books', 'butterfly', 'lollipops', 'skateboard']
>>> faves.remove('butterfly')
```

What does this list look like now?

```
['books', 'lollipops', 'skateboard']
```

Project time!

You now know all about lists!

Let’s put what we learnt into our project

Try to do the next Part

The tutors will be around to help!