Guess Who!

Welcome to the labs!

Thank you to our Sponsors!

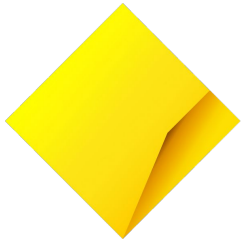
Platinum Sponsors:

ATLASSIAN

Australian Government

Australian Signals Directorate

Gold Sponsor:



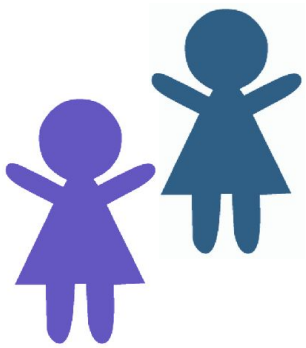
Commonwealth Bank

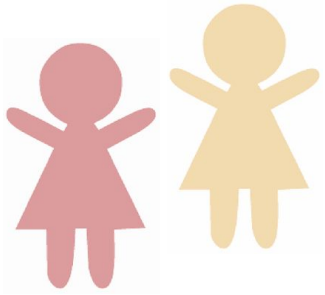
Who are the tutors?

Who are you?

Introduce your partner

1. Find a partner (someone you've never met before)
2. Find out:
   a. Their name
   b. What (school) year they are in
   c. A fun fact about them!
3. Introduce them to the rest of the group!





Log on

Log on and jump on the GPN website

girlsprogramming.network/workshop

Click on your node location

Click on your room.

From this page you can see:

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

Tell us you're here!

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

Start of Day Survey

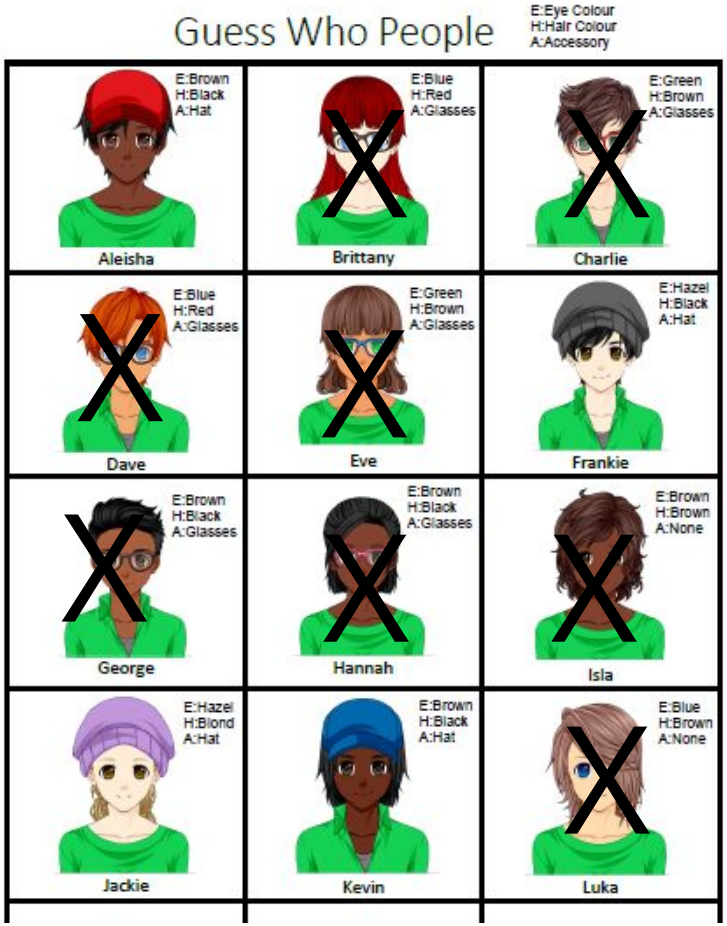
Today’s project!

Guess Who?

What is Guess Who?

- Fun Board Game
- We're going to code our own version
- Computer picks a character
- You ask questions to figure out who it is

What is Guess Who?



Q: Do they have a hat?

A: YES

Eliminates everyone without a hat

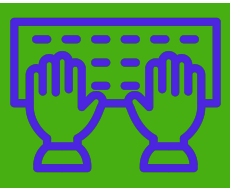
Keep asking questions until figure out who it is

Introduction to Edstem

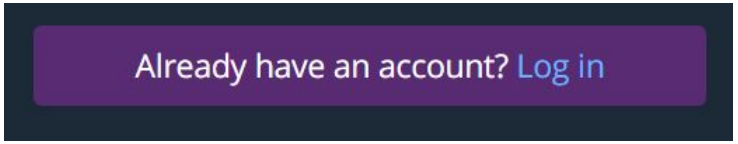
Log on

Click on your **Workbook** link to take you into EdStem

Signing up to Edstem

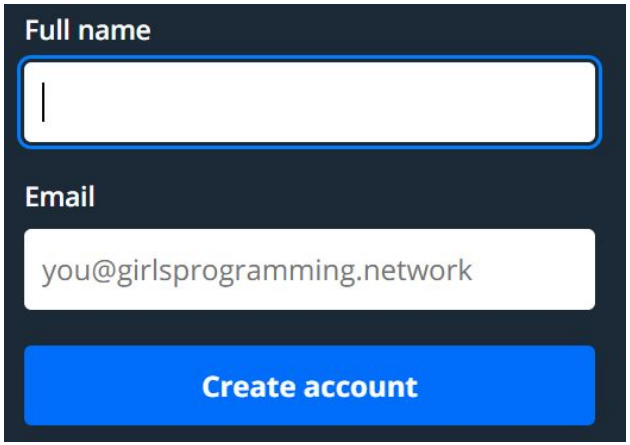


Log in if you already have a an “Edstem” account from a past GPN



If you haven’t got an account, let’s make one:

1. Type in your Full Name
2. Type in your personal email
3. Click Create Account
4. Go to your email and verify your new account
5. Create a password



Click Join Course

The name of your course will be at the top : →

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

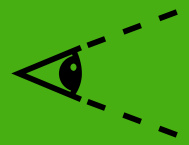
The main page:

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

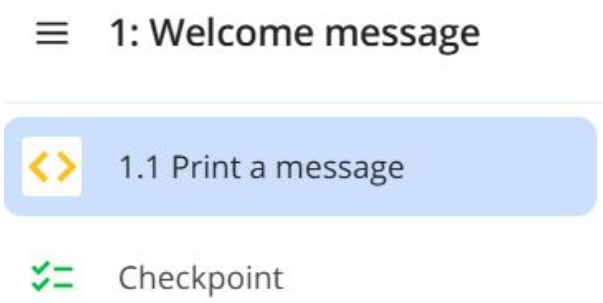
2. List of “Chapters” called something like **1:Welcome Message**
   They have an icon that looks like this:

3. 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:

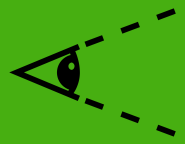


- **Lessons** where you will do your coding.
  - They have this icon: 
- **Checkpoints** 

  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.

There may also be **Bonus Lessons** to try if you want to or if you are waiting for the next lecture

How to do the work

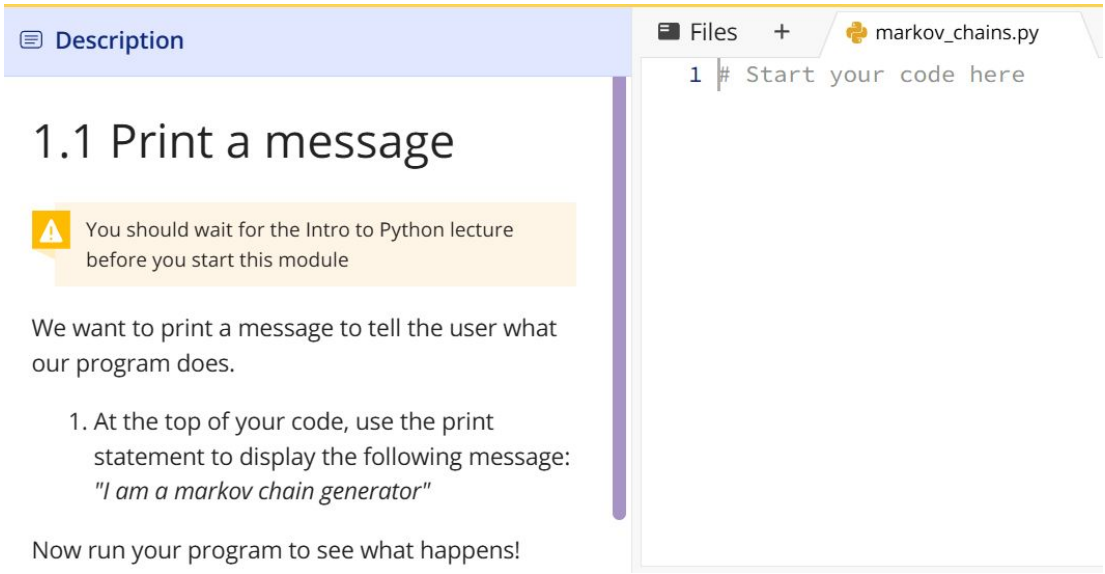


In each Lesson there is:

1. A section on the left with instructions
2. A section on the 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



Hints

Sometimes in a lesson, there's some code we want you to do that might be a bit tricky, to help you out we've added some hints. They look like this:

If you press the blue run button it will show you what that code does, you can even change the code to see if/how it changes.

These are **just hints** make sure you're not copying the hint into your code as it will likely end up breaking. They are just to show you the kinds of things you can do.

Running your code...

Click  in the bottom right hand corner

Your code will run and any output will display in the Console

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

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 if you haven't done this already.

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!

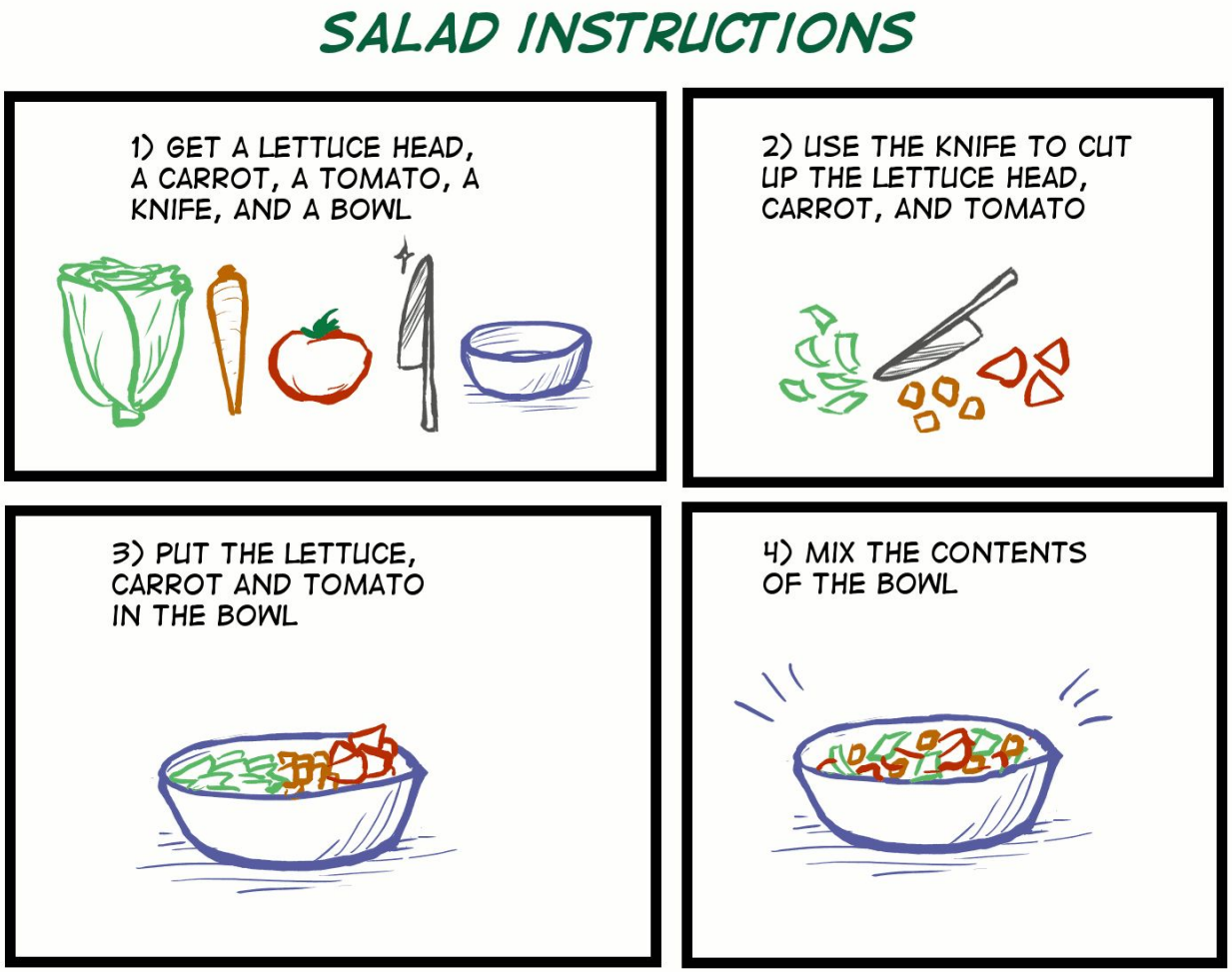
Programming is a language to talk to computers

People are smart! Computers are dumb!

Programming is like a recipe!

Computers do EXACTLY what you say, every time.

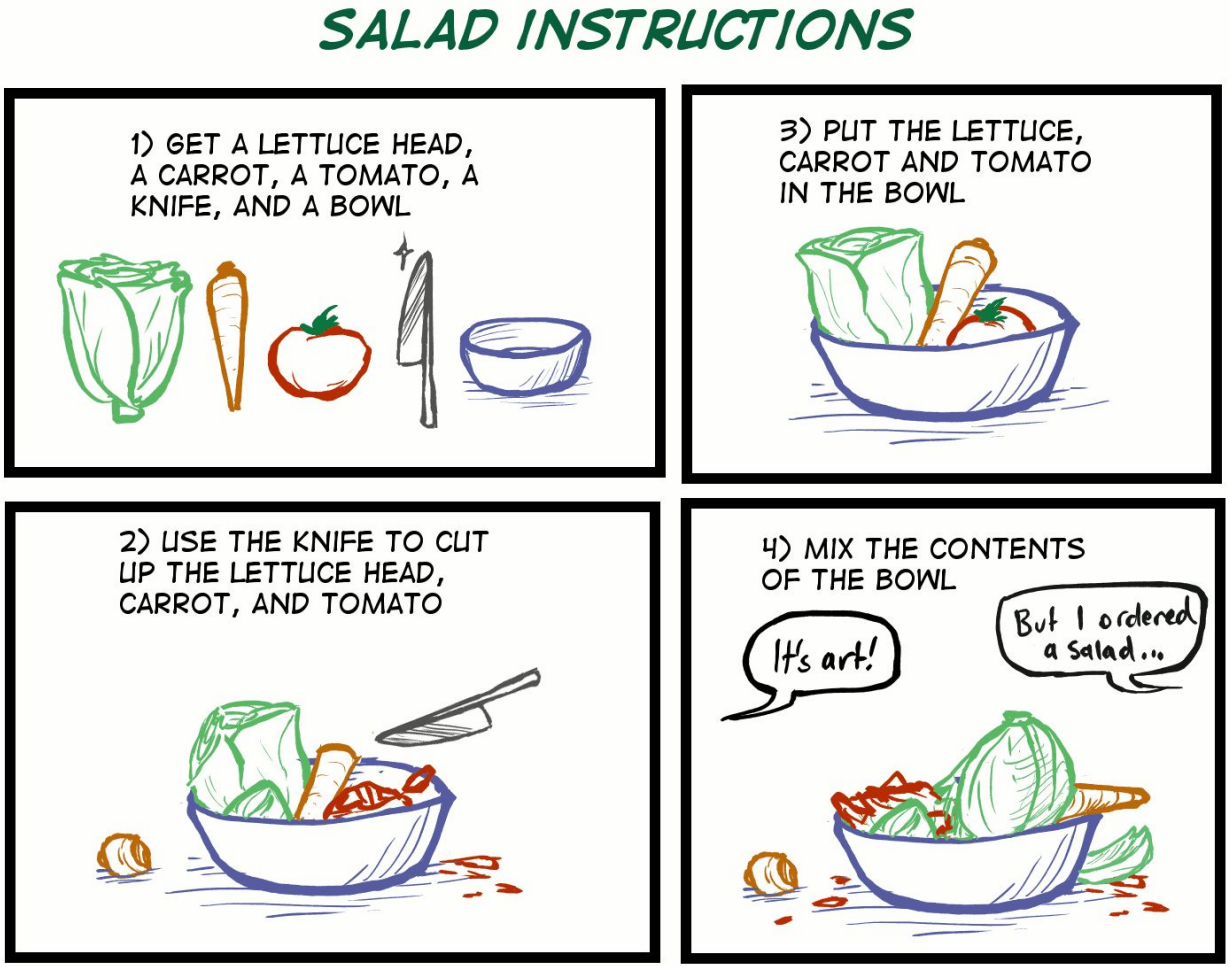
Which is great if you give them a good recipe!



People are smart! Computers are dumb!

But if you get it out of order….

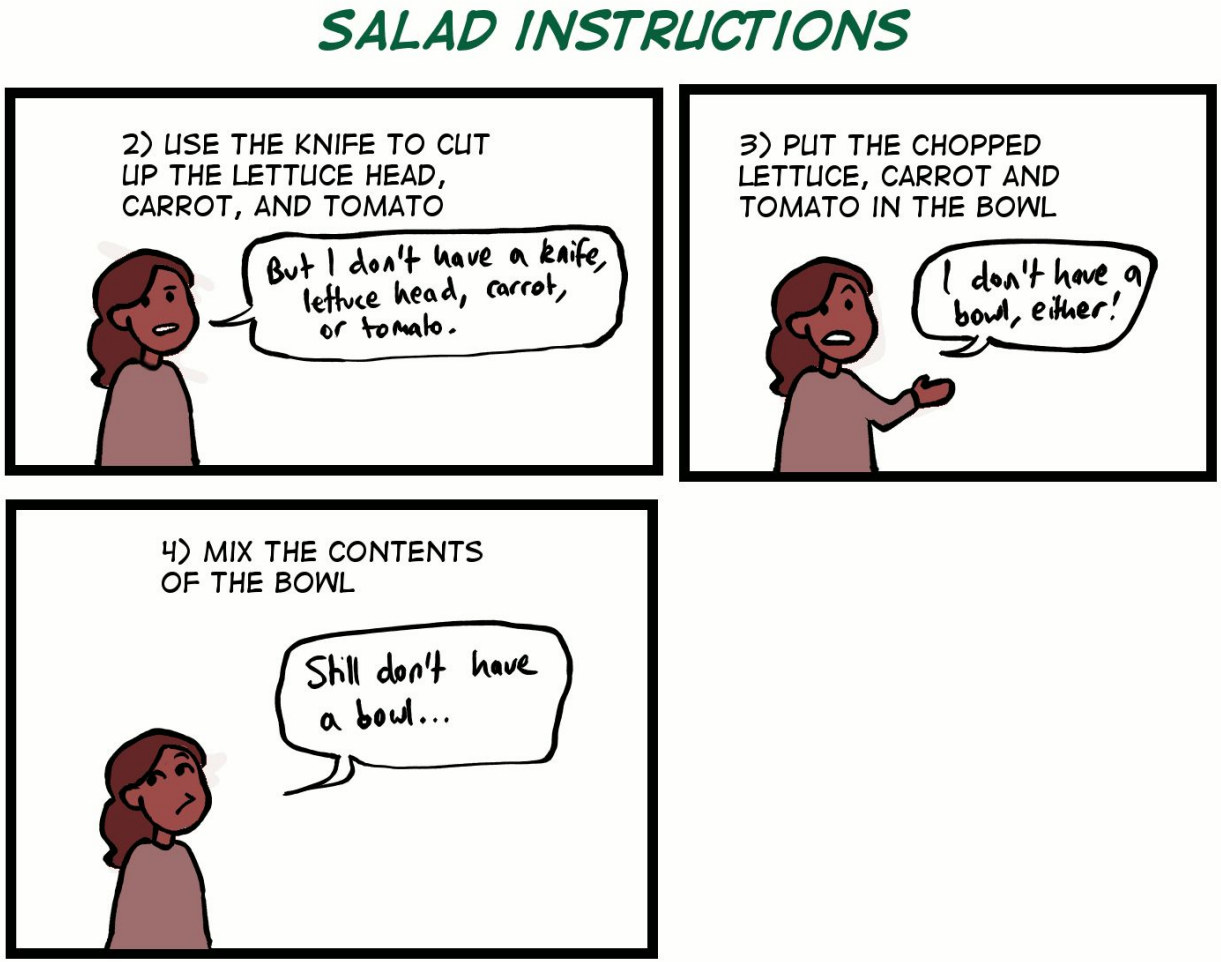
A computer wouldn't know this recipe was wrong!



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/thing has strengths!

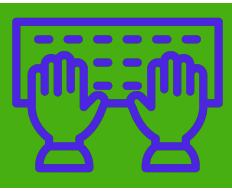
- Understand instructions despite:
  - Spelling mistakes
  - Typos
  - Confusing parts
- Solve problems
- Tell computers what to do
- Get smarter every day

- Does exactly what you tell it
- Does it the same every time
- Doesn't need to sleep!
- Will work for hours on end!
- Get smarter when you tell them how

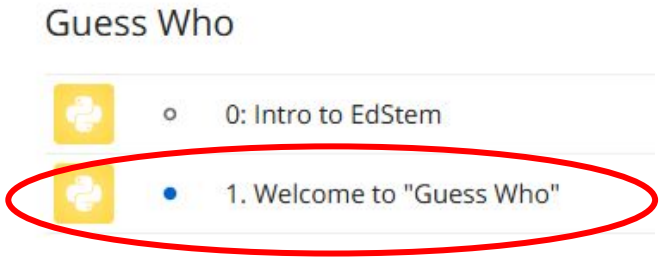
Intro to Python

Let’s get coding!

Let's make a mistake!

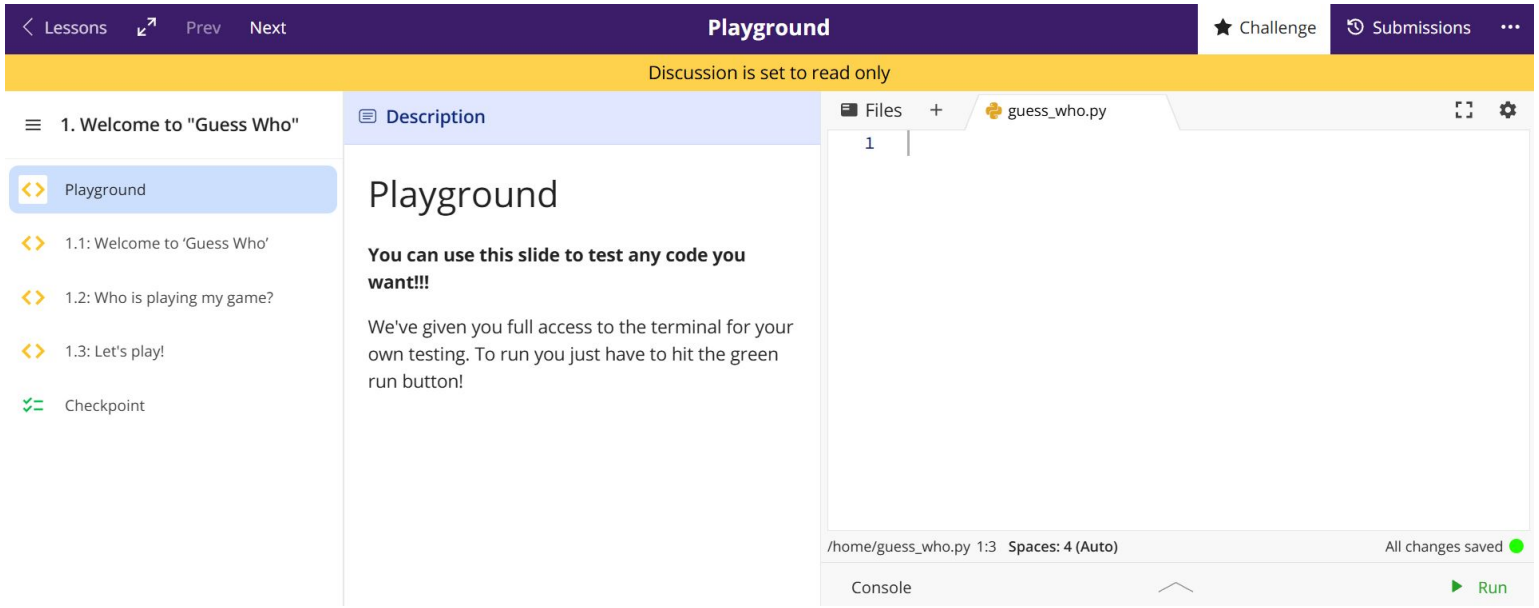


Click on Chapter 1 **Welcome to "Guess Who"**

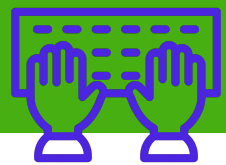


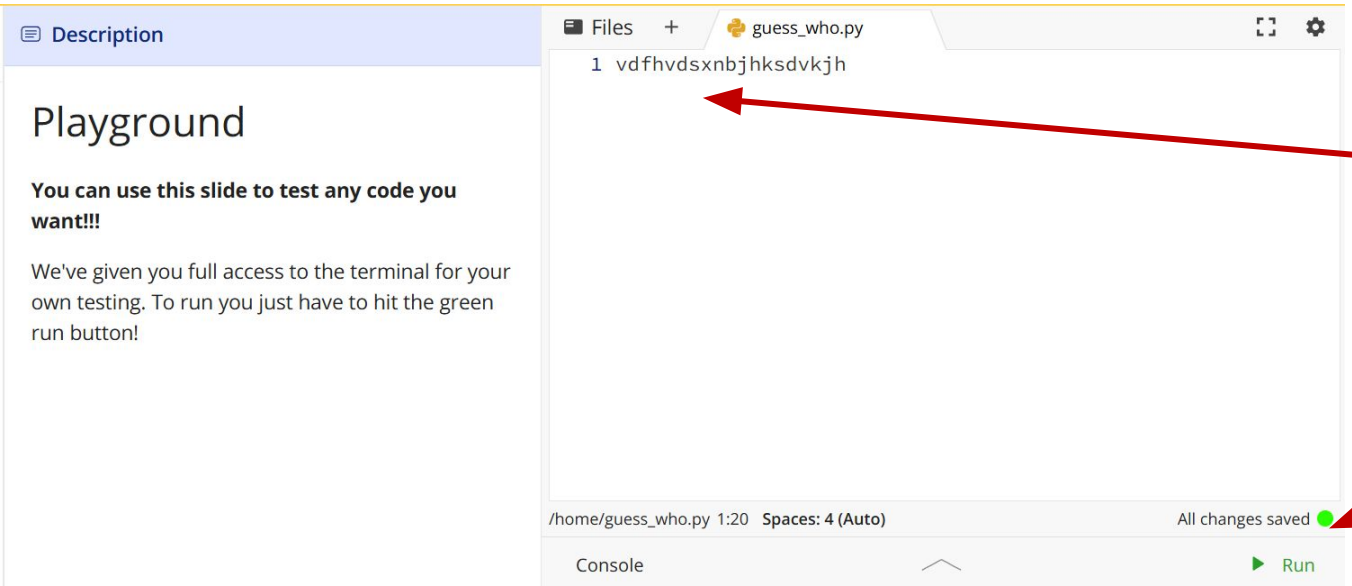
The **Playground** Lesson will open.

It looks like this



Let's make a mistake!

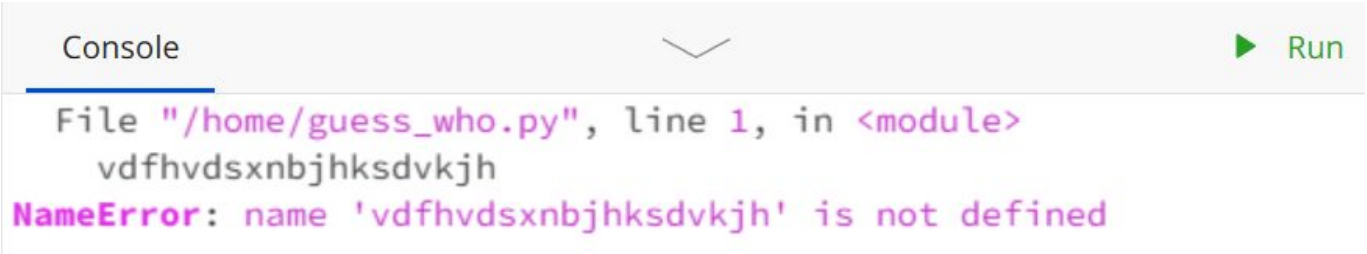




Type by **button mashing** the keyboard here - type anything you want

Click Run 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!

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

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!")
```

Write some code!!

This is the first bit of code we will do. What do you think it does?

```
print(“hello world”)
```

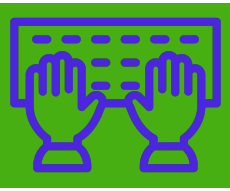
Write some code!!

This is the first bit of code we will do. What do you think it does?

```
print(“hello world”)
```

It prints the words “hello world” onto the screen!

Your turn!



1. Type the following into the code window (make sure you include the quotes (“) at the start and end)
2. Run the code by clicking Run

```
print(“hello world”)
```

Did it print the text:

```
hello world
```

Try printing something different - don’t forget the quotes

Tell me more!

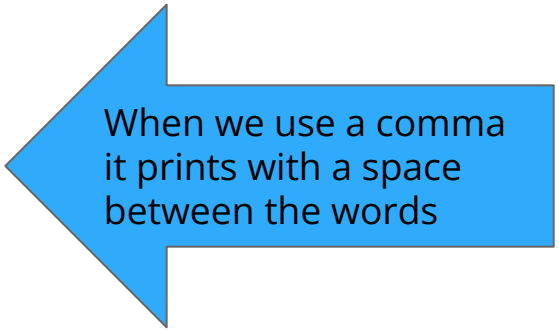
We can `print` things in lots of different ways in python!

```
>>> print("Hello world!")

>>> print("Hello", "world!")

>>> print("Hello" + "world")
```

Tell me more!

We can `print` things in lots of different ways in python!

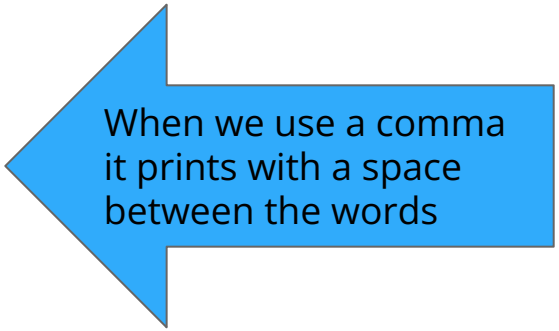
```
>>> print("Hello world!")
Hello world!
>>> print("Hello", "world!")

>>> print("Hello" + "world")
```



Tell me more!

We can `print` things in lots of different ways in python!

```
>>> print("Hello world!")
Hello world!
>>> print("Hello", "world!")
Hello world!
>>> print("Hello" + "world")
```



Tell me more!

We can `print` things in lots of different ways in python!

```
>>> print("Hello world!")
Hello world!
>>> print("Hello", "world!")
Hello world!
>>> print("Hello" + "world")
```

When we use a + it just adds the words together - no spaces

Tell me more!

We can `print` things in lots of different ways in python!

```
>>> print("Hello world!")
Hello world!
>>> print("Hello", "world!")
Hello world!
>>> print("Hello" + "world")
Helloworld!
```

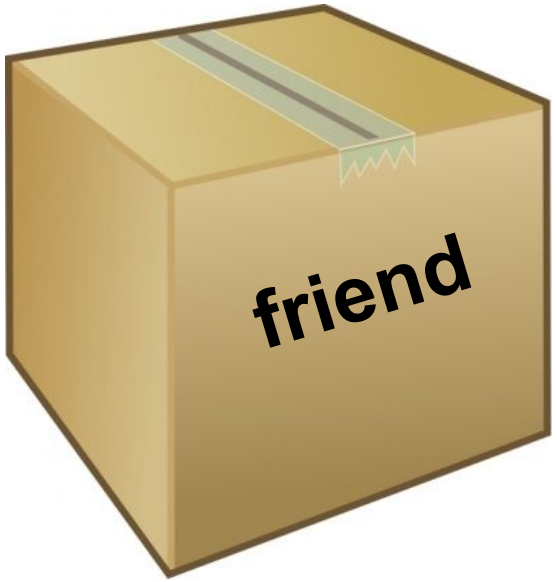
When we use a + it just adds the words together - no spaces

Variables

Variables are useful for storing things that change

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

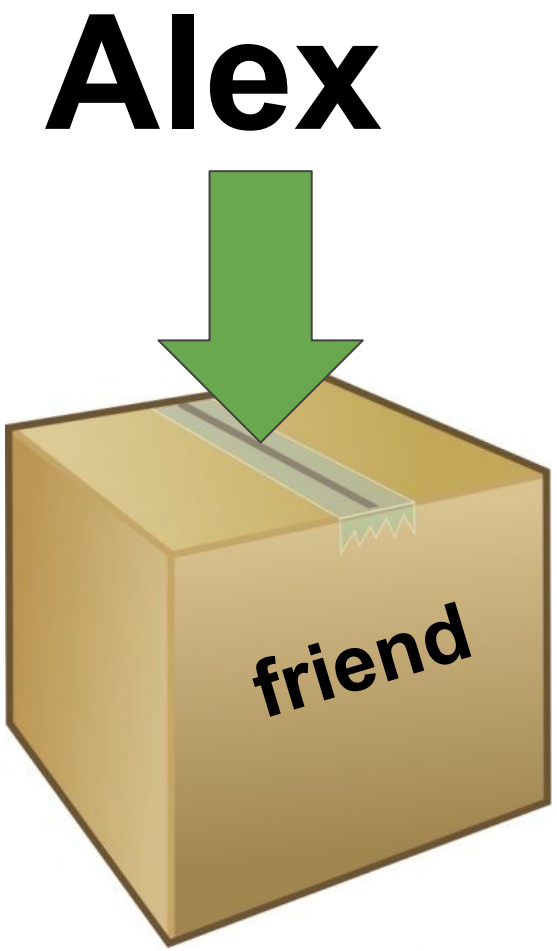
You can think of it like putting information in a box and giving it a label



Variables

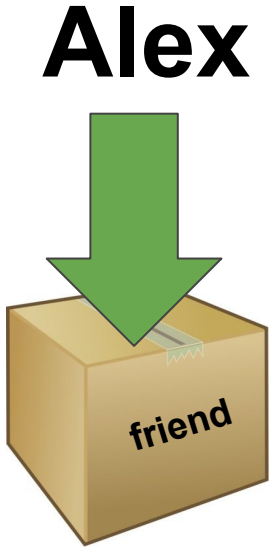
When coding, we can make a variable called **friend** and set it to a value like this

friend = “Alex”



Variables

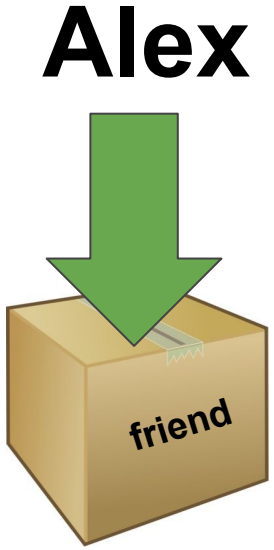
Instead of writing the word “Alex”, we can write **friend** (the variable’s name).
The computer will substitute the current value of the variable.
It’s like we’re getting the value out of the box!




```
print(friend)
```

Variables

Instead of writing the word “Alex”, we can write **friend** (the variable’s name).
The computer will substitute the current value of the variable.
It’s like we’re getting the value out of the box!




```
print(friend)
```

```
Alex
```

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)
```

What will this output? `My favourite animal is a dog`

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?

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

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?

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

Asking a question!

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

We can write code to get the computer to ask us a question!

We use **`input`** and it looks like this:

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

Asking a question!

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

This is what happens ...

1. Computer prints what's in quotes after `input`
2. Computer waits for someone to type in a response
3. Computer stores response in variable called my_name
4. Computer prints 'Hello' and whatever is in the variable my_name

```
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!

Big Tip : Put a space at the end of the question so it won't be squished together with your answer - it looks nicer!

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

SPACE 🙂

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

NO SPACE 😕

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

Project time!

You now know all about printing, variables and input!

Let’s put what we learnt into our project

Try to do Part 1

Don't forget to copy your code when you move to a new Lesson!

The tutors will be around to help!

Lists

Storing groups of things in variables

- We know how to store individual things, but what if we have a group of things?
- We can try to do this with variables

```
>>> day1 = 'Monday'
>>> day2 = 'Tuesday'
>>> day3 = 'Wednesday'
>>> day4 = 'Thursday'
>>> day5 = 'Friday'
>>> day6 = 'Saturday'
>>> day7 = 'Sunday'
```

- But this can get long and hard to deal with *really* quickly...

Lists can store multiple things

- It's better to create a `list`
- A `list` is an ordered group of related items, all in the same variable
- So instead of using 7 variables to store the days, we can use one:

```
>>> days = ['Monday', 'Tuesday',
'Wednesday', 'Thursday', 'Friday',
'Saturday', 'Sunday']
```

Creating lists

- A `list` is created using square brackets in Python
- An example

```
>>> words = ['This', 'is', 'a', 'sentence']
```

- Think of your four favourite things.....what are they?
- How could we store them in a list?




Your Favourite Things!

Stored in the variable favourites

Made up of different items (these are strings)

The items are separated by commas

```
favourites=[‘books','chocolate','butterfly','skateboard']
```

Has square brackets

Accessing Lists!

```
favourites=[‘books','chocolate','butterfly','skateboard']
```

- The favourites `list` holds four words in order.
- We can count out the items using index numbers!

0 **1** **2** **3**




- **Indices start from zero!**

Accessing Lists

- We access the items in a `list` using an index in square brackets

```
                  [0]          [1]              [2]                [3]
favourites=['books','chocolate','butterfly','skateboard']
```

```
>>> favourites[0]
'books'
>>> favourites[1]
'chocolate'
```

- What code do you need to access the **3rd** item in the list?

Accessing Lists

- We access the items in a `list` using an index in square brackets

```
                  [0]          [1]              [2]                [3]
favourites=['books','chocolate','butterfly','skateboard']
```

```
>>> favourites[0]
'books'
>>> favourites[1]
'chocolate'
```

- What code do you need to access the **3rd** item in the list?

```
>>> favourites[2]
'butterfly'
```

Falling off the edge

- Python complains if you try to go past the end of a `list`

```
>>> favourites = ['books', 'chocolate',
'butterfly', 'skateboard']
>>> favourites[4]


Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
IndexError: list index out of range
```

List of lists!

You can put anything in a list, even more lists!

We could use a list of lists to store tennis partners.!

```
tennis_pairs =[["Alex", "Emily"], ["Kass",
"Annie"], ["Amara", "Viv"]]
```

Project time!

Now you know all about lists!

Let's put what we learnt into our project

Try to do Part 2

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

```
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:
    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?

```
>>>
```

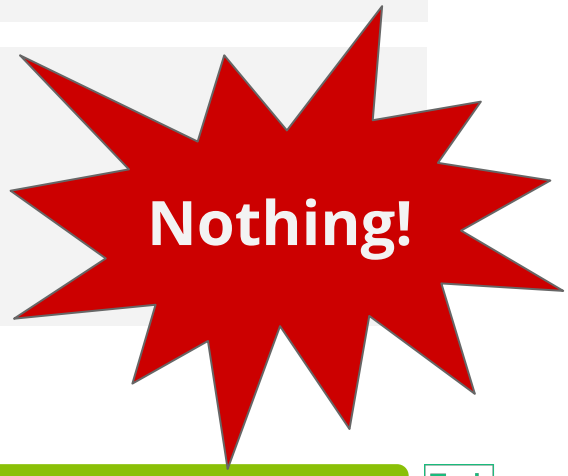
Conditions

How about a different number???

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

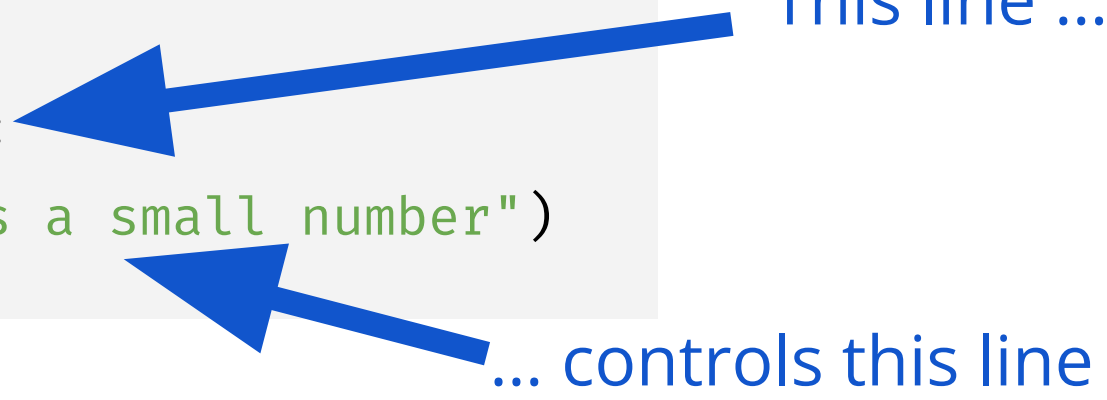
What do you think happens?

```
>>>
```



If statements

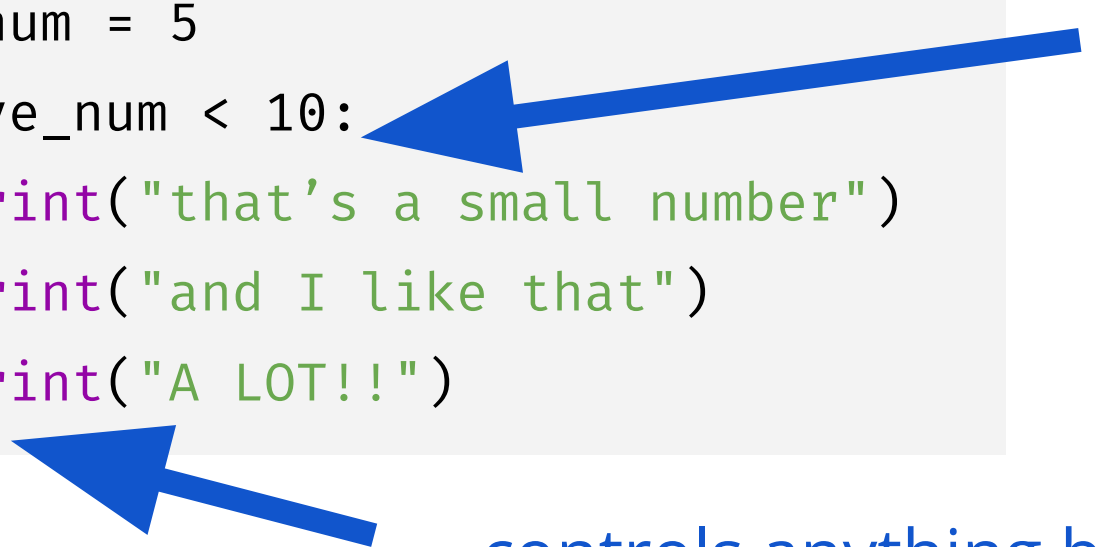
```
fave_num = 5
if fave_num < 10:
    print("that's a small number")
```



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!
```

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 :(
```

Else statements

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

Remember ...

==

When testing for equals in your condition

:

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

Project Time!

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

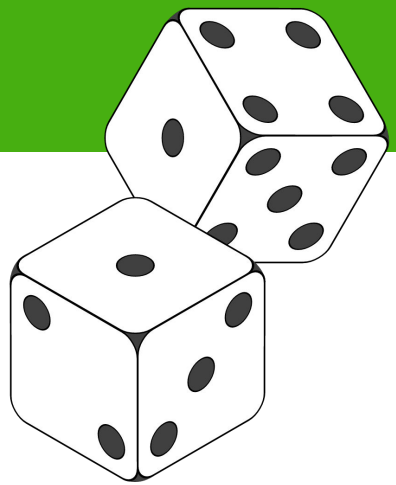
Let's put what we learnt into our project

Try to do Parts 3 and 4

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!

We use **random.choice** to randomly select something from a list

```
import random
shopping_list = ["eggs", "bread", "apples", "milk"]
random.choice(shopping_list)
```

Each time we run this we would probably get a different answer

`eggs OR bread OR apples OR milk`

Using the random module

You can also assign your random choice to a variable and then use that variable in your code

```
import random
shopping_list = ["eggs", "bread", "apples", "milk"]
random_food = random.choice(shopping_list)
print(random_food)
```

The variable **`random_food`** contains the random choice that was made from the list

Project Time!

Raaaaaaaaaandom! Can you handle that?

Let's put what we learnt into our project

Try to do Part 5

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!

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!

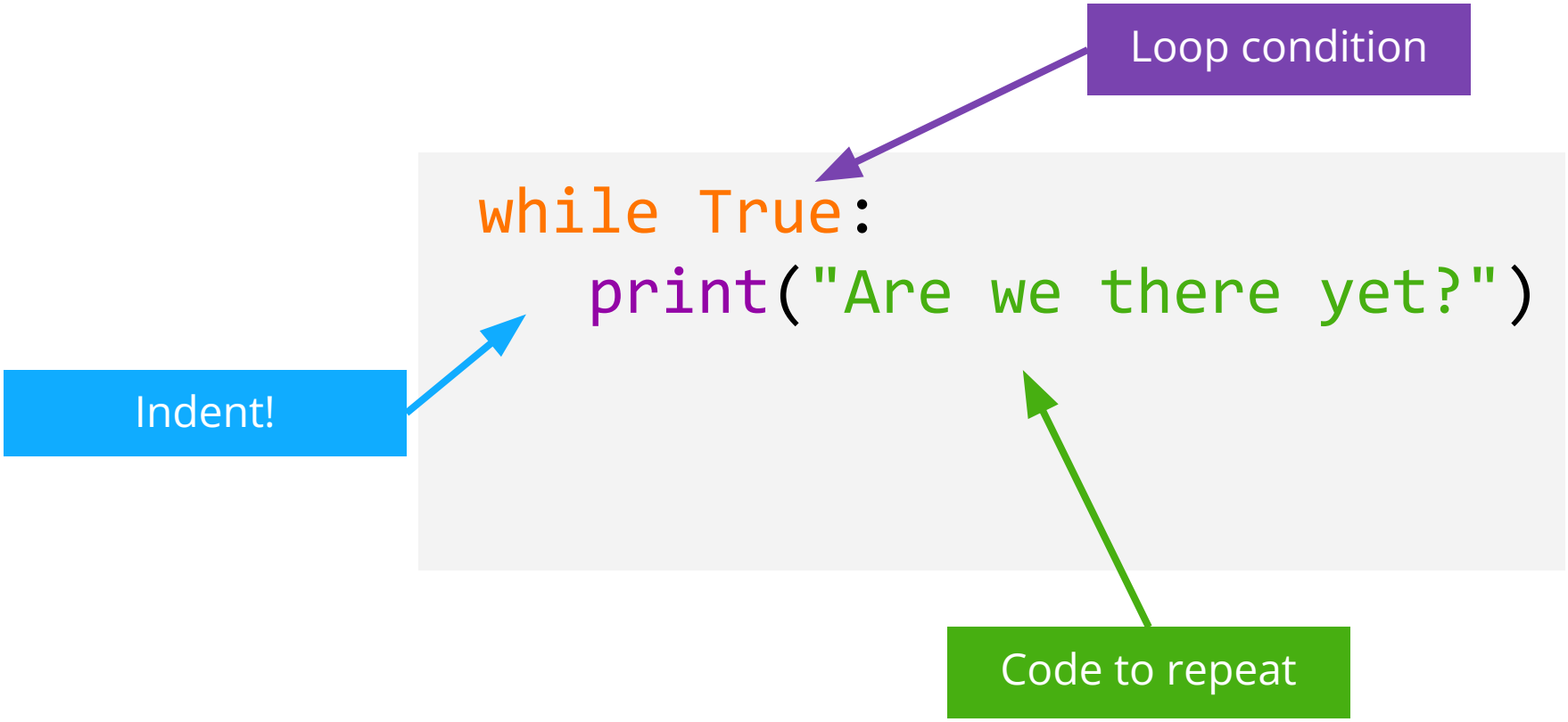
So we set a condition that is always True!

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?
```

Introducing … `while` loops!



Give me a break!

But what if I want to get out of a loop early?

That's when we use the break keyword!

```
while True :
    number = input("Guess a number: ")

    if number == "42":
        print("You got it right!")
        break
```

Give me a break!

But what if I want to get out of a loop early?

That's when we use the break keyword!

```
while True :
    number = input("Guess a number: ")

    if number == "42":
        print("You got it right!")
        break
```

```
Guess a number:
```

Give me a break!

But what if I want to get out of a loop early?

That's when we use the break keyword!

```
while True :
    number = input("Guess a number: ")

    if number == "42":
        print("You got it right!")
        break
```

```
Guess a number: 5
```

Give me a break!

But what if I want to get out of a loop early?

That's when we use the break keyword!

```
while True :
    number = input("Guess a number: ")

    if number == "42":
        print("You got it right!")
        break
```

```
Guess a number: 5
Guess a number:
```

Give me a break!

But what if I want to get out of a loop early?

That's when we use the break keyword!

```
while True :
    number = input("Guess a number: ")

    if number == "42":
        print("You got it right!")
        break
```

```
Guess a number: 5
Guess a number: 100
```

Give me a break!

But what if I want to get out of a loop early?

That's when we use the break keyword!

```
while True :
    number = input("Guess a number: ")

    if number == "42":
        print("You got it right!")
        break
```

```
Guess a number: 5
Guess a number: 100
Guess a number:
```

Give me a break!

But what if I want to get out of a loop early?

That's when we use the break keyword!

```
while True :
    number = input("Guess a number: ")

    if number == "42":
        print("You got it right!")
        break
```

```
Guess a number: 5
Guess a number: 100
Guess a number: 42
```

Give me a break!

But what if I want to get out of a loop early?

That's when we use the break keyword!

```
while True :
    number = input("Guess a number: ")

    if number == "42":
        print("You got it right!")
        break
```

```
Guess a number: 5
Guess a number: 100
Guess a number: 42
You got it right!
```

Project Time!

while we're here:

Let's put what we learnt into our project

Try to do Part 6!

Then try Extension Parts 7 - 10

The tutors will be around to help!

Tell us what you think!

Click on the

End of Day Form

and fill it in now!