Guess Who!

Welcome to the Labs

Thank you to our Sponsors!

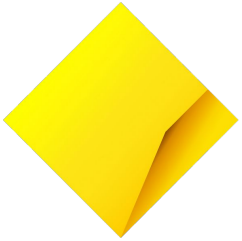
Platinum Sponsors:

ATLASSIAN

Australian Government

Australian Signals Directorate

Gold Sponsor:



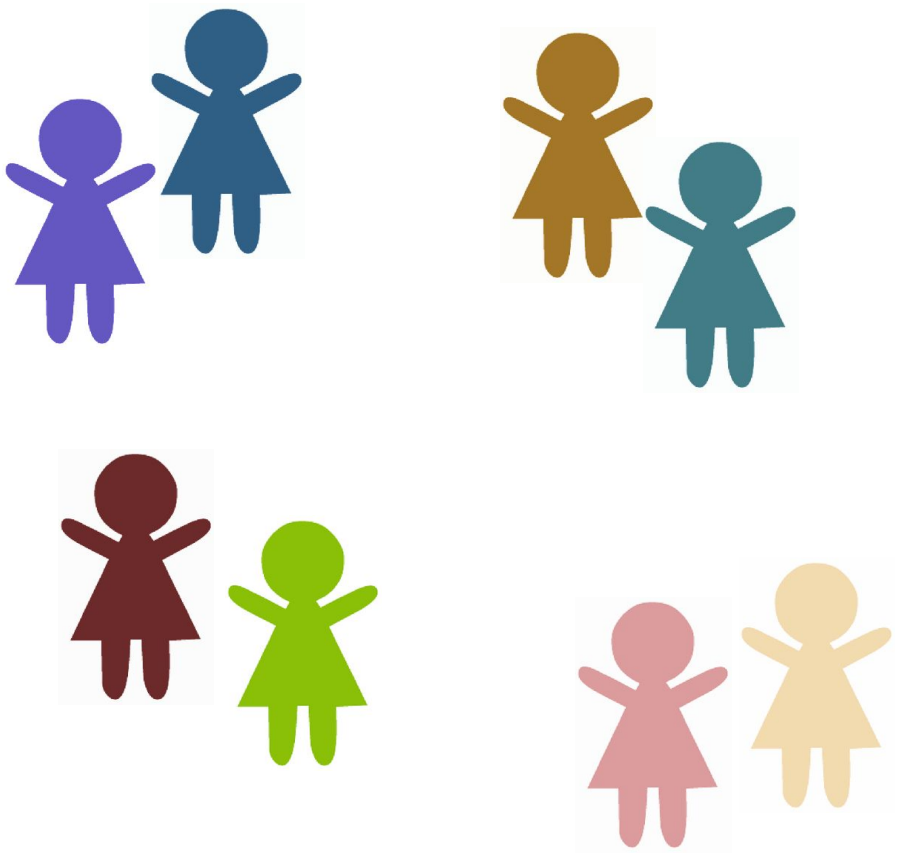
Commonwealth Bank

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

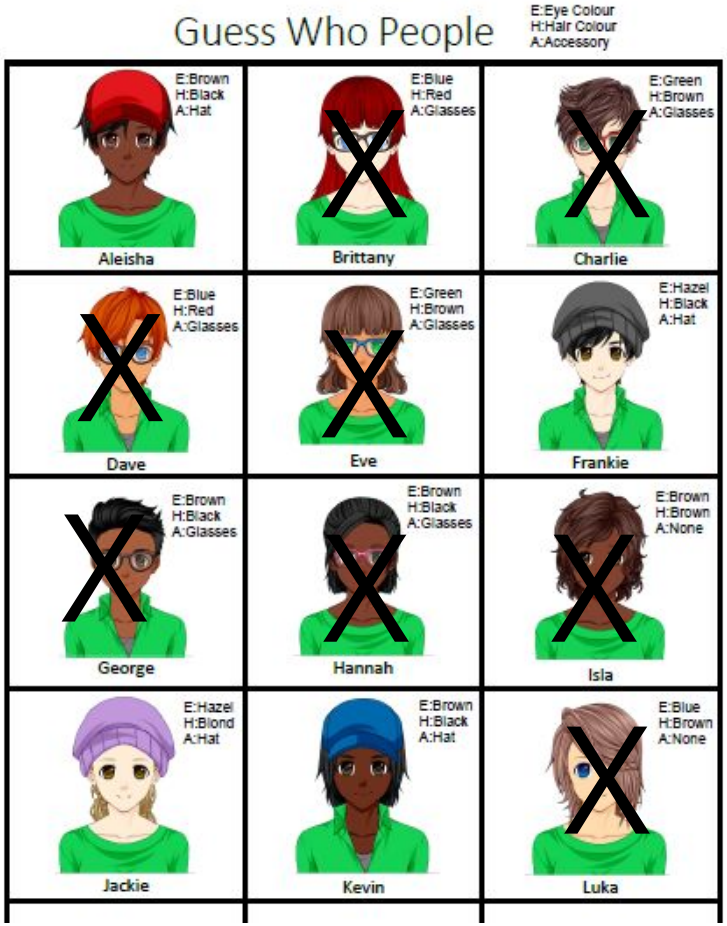
Today’s project!

Guess Who?

What is Guess Who?

- Fun Board Game
- We're going to code our own version
- You pick a character, keep is secret!
- The computer will ask you questions to figure out who it is

What is Guess Who?



Q: Do they have a hat?

A: YES

Eliminates everyone without a hat

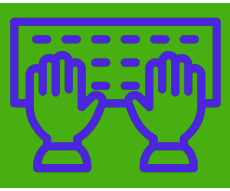
Keep asking questions until figure out who it is

Introduction to Edstem

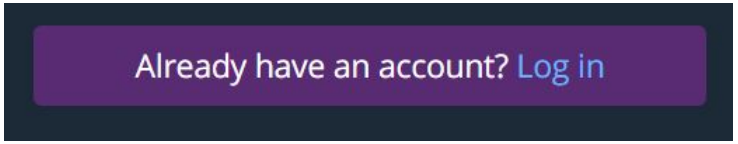
Log on

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

Signing up to Edstem

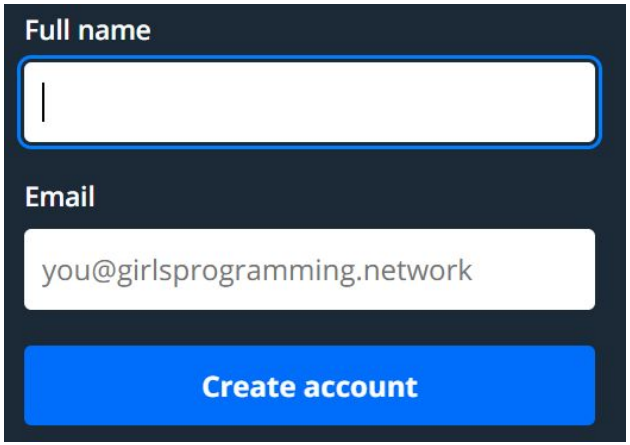


Log in if you already have a an “Edstem” account from a past GPN



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

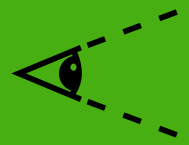
The main page:

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

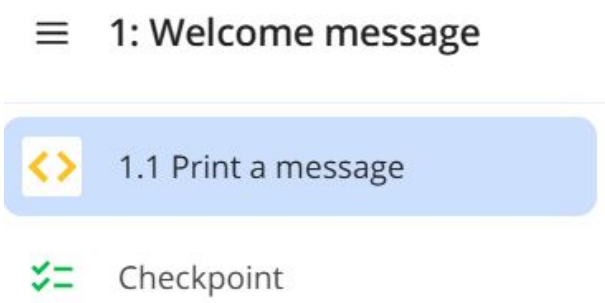
2. List of “Chapters” called something like **1:Welcome Message**
   They have an icon that looks like this:

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

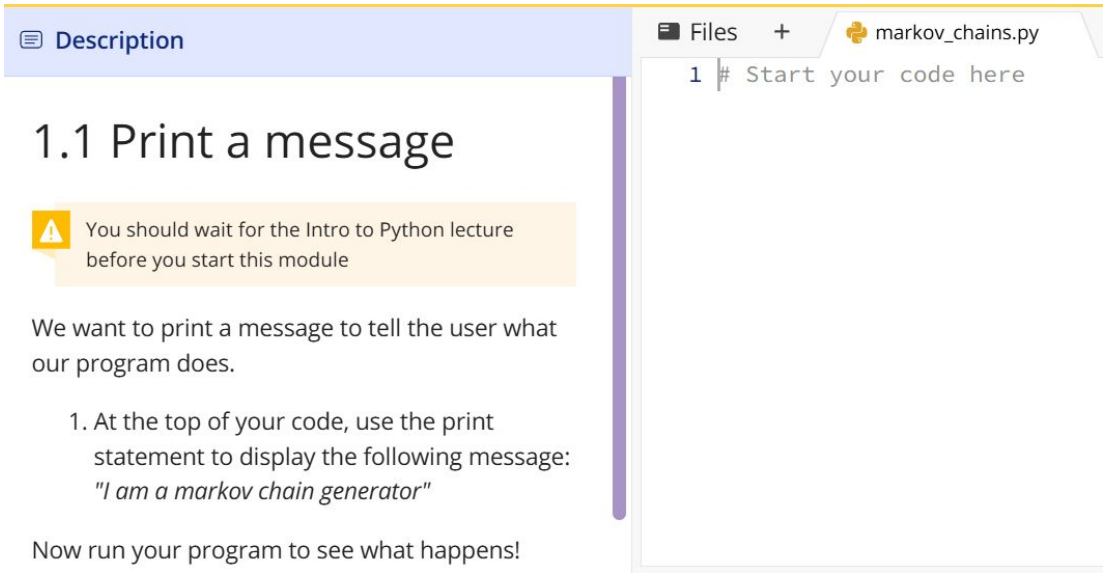
How to do the work

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:

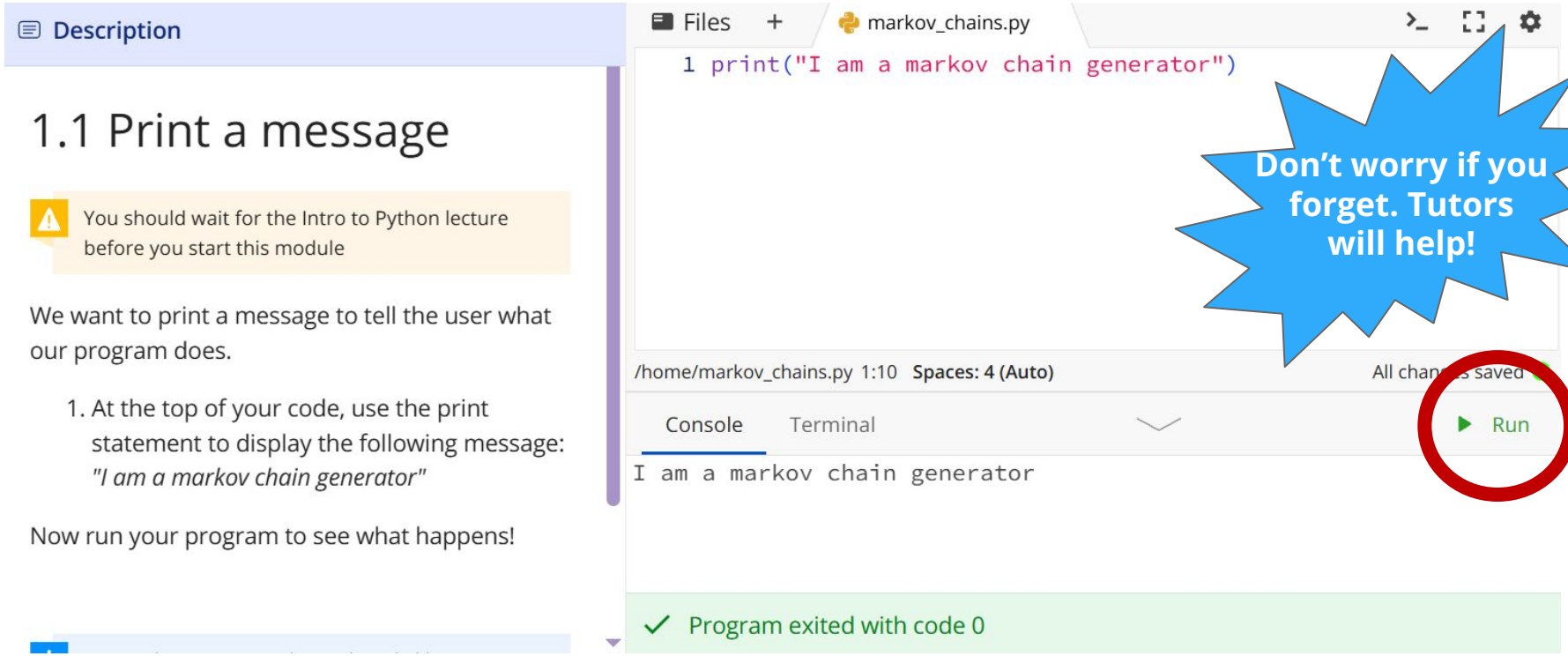
If you press the blue run button it will show you what that code does, you can even change the code to see if/how it changes.

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 ▶ Run 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!

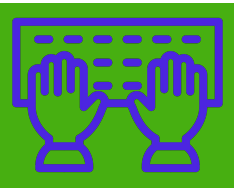
You should now sign up and join our EdStem class if you haven't done this already.

Remember the tutors will be around to help!

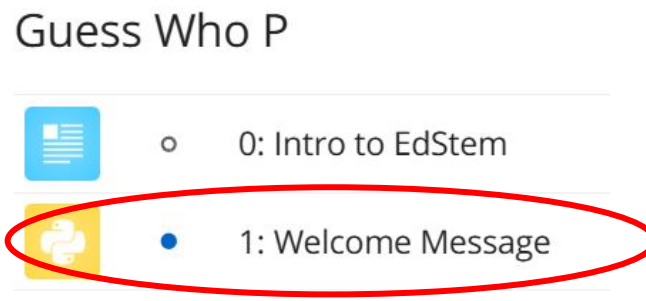
Intro to Python

Let’s get coding!

Let's make a mistake!

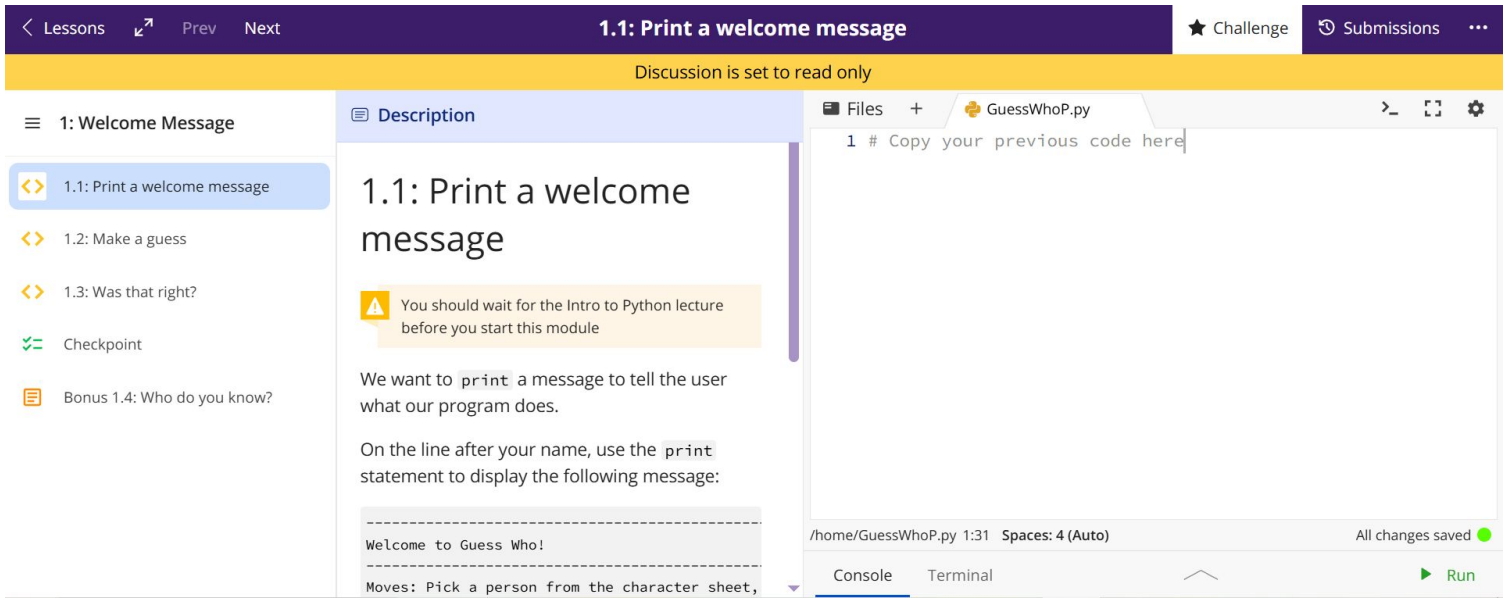


Click on Chapter 1 **"Welcome Message"**

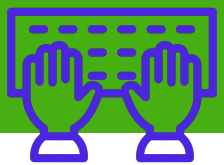


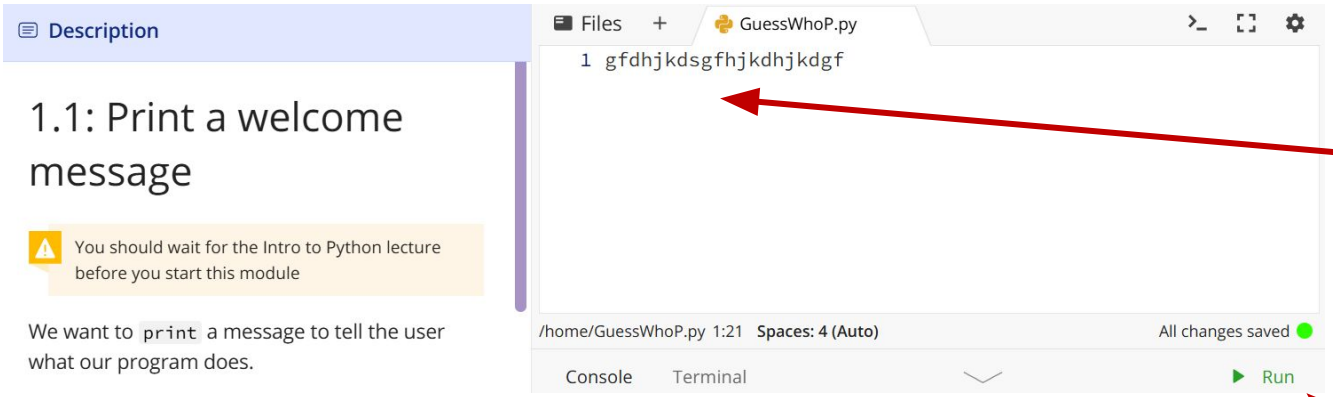
The **Print a Welcome Message** Lesson opens.

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!")
```

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!

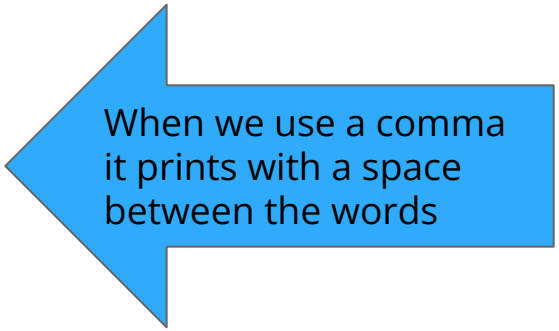
We can `print` things in lots of different ways in python!

```
>>> 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!

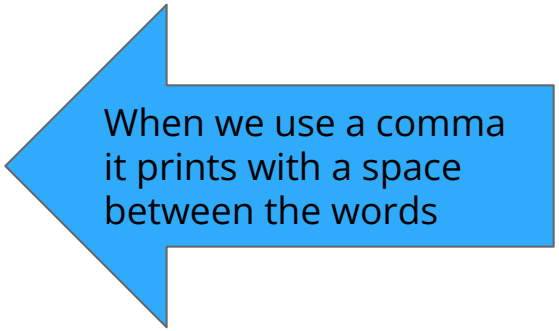
```
>>> print("Hello world!")
Hello world!
>>> print("Hello", "world!")

>>> 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

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!

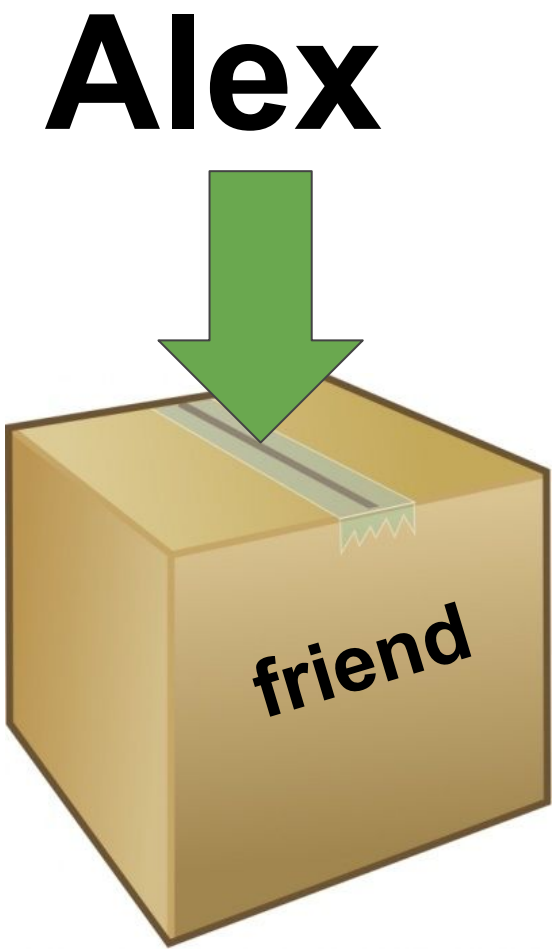
We can `print` on many lines at once!

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

Variables

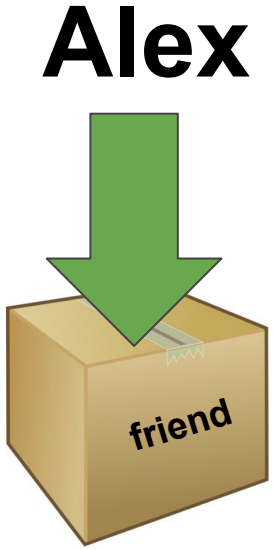
When coding, we can make a variable called **friend** and set it to a value like this

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)
```

```
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?

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

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?

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!

Project time!

Guess Who!

Now that we've had a Python refresher, try and do Part0 and 1 !

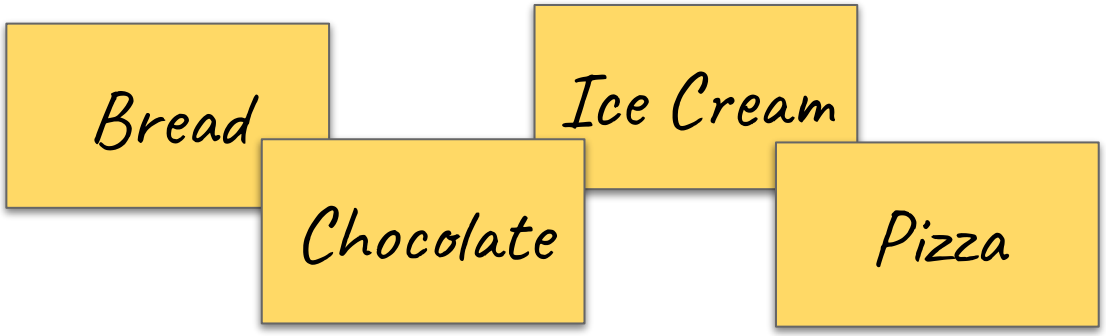
The tutors will be around to help!

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

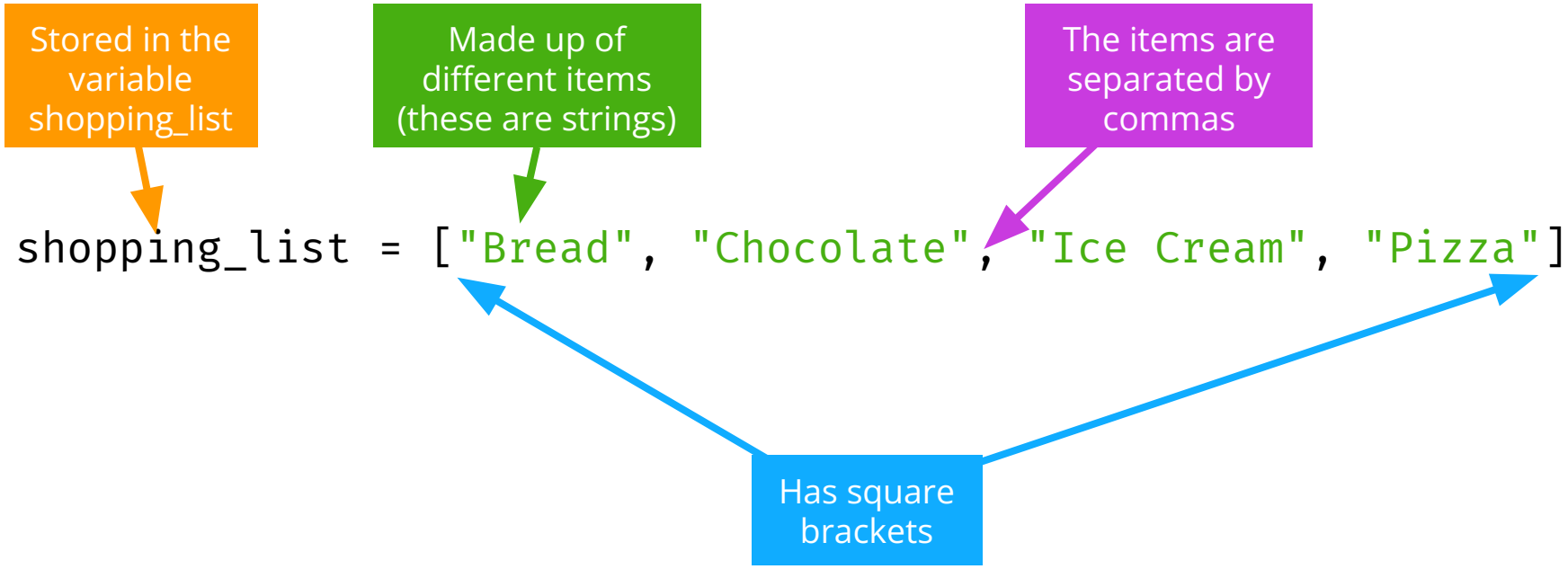
```
>>> shopping_item1 = "Bread"
>>> shopping_item2 = "Chocolate"
>>> shopping_item3 = "Ice Cream"
>>> shopping_item4 = "Pizza"
```

So much repetition!

Instead we use a python list!

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

List anatomy



Accessing Lists!

The favourites `list` holds four strings in order.

We can count out the items using index numbers!

0

1

2

3

Remember: Indices start from zero!

Accessing Lists

We access the items in a `list` with an index such as `[0]`:

```
>>> faves[0]
'books'
```

What code do you need to access the second item in the list?

Accessing Lists

We access the items in a `list` with an index such as `[0]`:

```
>>> faves[0]
'books'
```

What code do you need to access the second item in the list?

```
>>> faves[1]
'butterfly'
```

0	**[1]**	2	3
			

Falling off the edge

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

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


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

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']
```

List of lists!

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

We could use a list of lists to store different sports teams!

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

Get the first pair in the list

```
>>> first_pair = tennis_pairs[0]
>>> ["Alex", "Emily"]
```

Now we have the first pair handy, we can get the first the first player of the first pair

```
>>> fist_player = first_pair[0]
>>> "Alex"
```

Project time!

You now 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 decision.

First we test if the condition is met!

Then maybe we'll do the thing

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

Yep it's raining

..... take an umbrella

Booleans (True and False)

Computers store whether a condition is met in the form of

`True` and `False`

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

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

Booleans (True and False)

Computers store whether a condition is met in the form of

`True` and `False`

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

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

Booleans (True and False)

Computers store whether a condition is met in the form of

`True` and `False`

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

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

Booleans (True and False)

Computers store whether a condition is met in the form of

`True` and `False`

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

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

Booleans (True and False)

Computers store whether a condition is met in the form of

`True` and `False`

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

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

Booleans (True and False)

Computers store whether a condition is met in the form of

`True` and `False`

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

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

Booleans (True and False)

Computers store whether a condition is met in the form of

`True` and `False`

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

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

Conditions

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

```
fave_num = 5
if fave_num < 10:
    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???

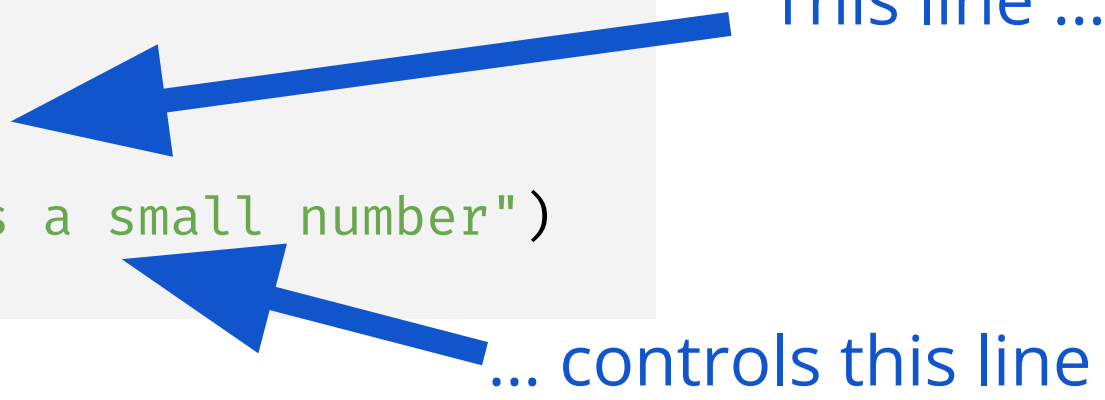
```
fave_num = 9000
if fave_num < 10:
    print("that's a small number")
```

What do you think happens?

```
>>>
```

If statements

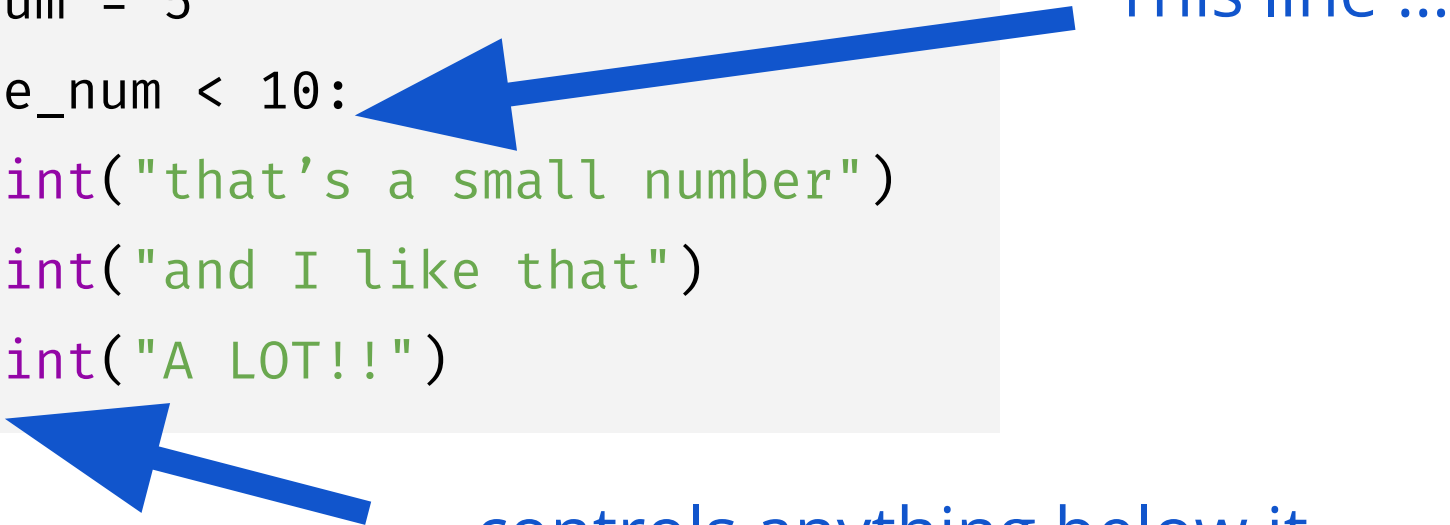
```
fave_num = 5
if fave_num < 10:
    print("that's a small number")
```



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

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

```
What happens??
```

If statements

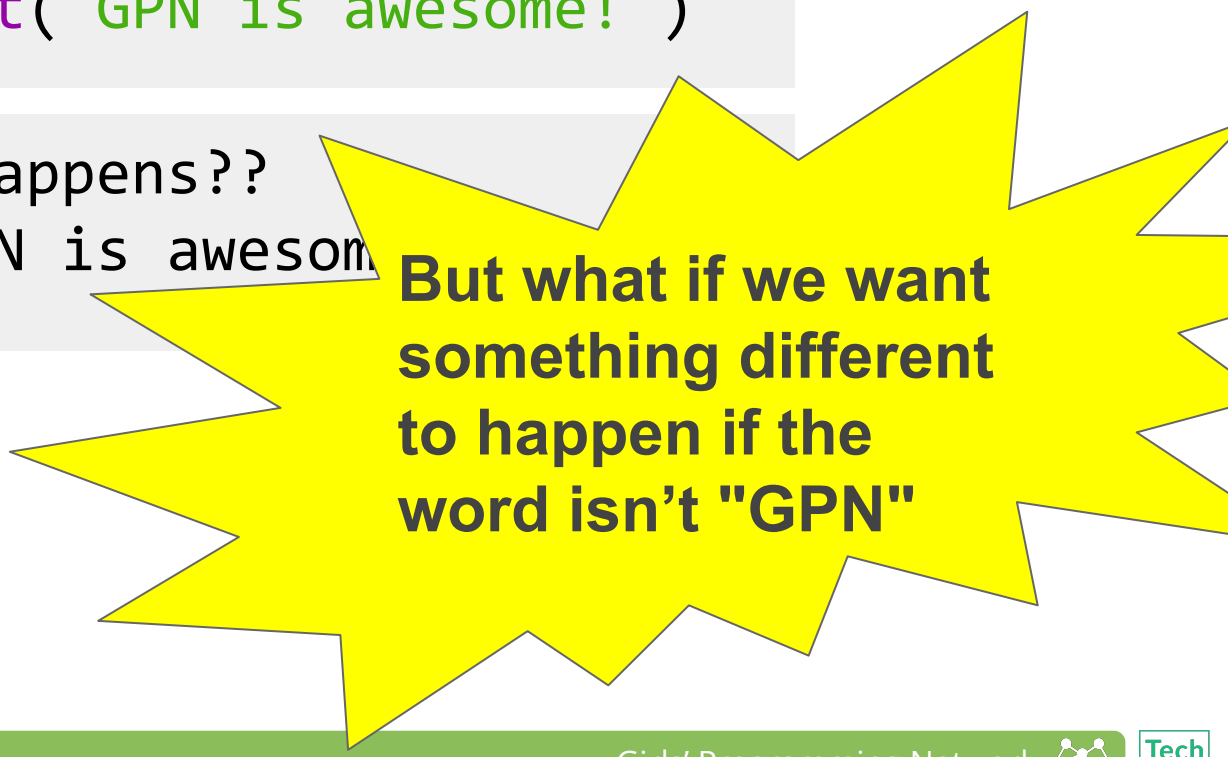
```
word = "GPN"
if word == "GPN":
  print("GPN is awesome!")
```

```
What happens??
>>> GPN is awesome!
```

Else statements

```
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

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!
```

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 go 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.

Project Time!

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

See `if` you can do the next part

The tutors will be around to help!

For Loops

Looping through lists!

What would we do if we wanted to print out this list, one word at a time?

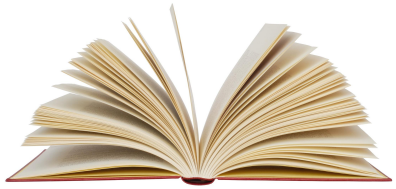
```
words = ['This', 'is', 'a', 'sentence']


print(words[0])
print(words[1])
print(words[2])
print(words[3])
```

What if it had a 100 items??? That would be **BORING!**

For Loops

For loops allow you to do something for **each** item in a **group** of things

There are many real world examples, like:



For each page in this book:
    Read page

For each chip in this bag of chips:
    Eat chip

Looping over a list of ints

We can loop through a list:

```
numbers = [1, 2, 3, 4]
for i in numbers:
    print(i)
```

What's going to happen?

Looping over a list of ints

We can loop through a list:

```
numbers = [1, 2, 3, 4]
for i in numbers:
    print(i)
```

What's going to happen?

```
>>> 1
>>> 2
>>> 3
>>> 4
```

- Each item of the list takes a turn at being the variable i
- Do the body once for each item
- We're done when we run out of items!

Looping over a list of ints

Strings are lists of letters!

```
word = "cat"
for i in word:
    print(i)
```

What's going to happen?

Looping over a list of ints

Strings are lists of letters!

```
word = "cat"
for i in word:
    print(i)
```

What's going to happen?

```
>>> c
>>> a
>>> t
```

How does it work??

Somehow it knows how to get one fruit out at a time!!

It's like it knows english!

```
fruits = ['apple', 'banana', 'mango']
for fruit in fruits:
    print('yummy ' + fruit)
```

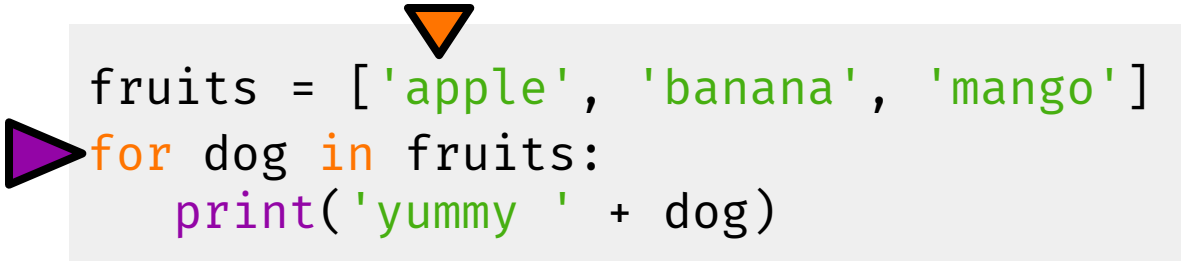
But fruit is just a variable! We could call it anything! Like dog!

```
fruits = ['apple', 'banana', 'mango']
for dog in fruits:
    print('yummy ' + dog)
```

```
>>> Yummy apple
>>> Yummy banana
>>> Yummy mango
```

How does it work??

Everything in the list gets to have a turn at being the dog variable



```
fruits = ['apple', 'banana', 'mango']
for dog in fruits:
    print('yummy ' + dog)
```

Let's set dog to to the first thing in the list!
dog is now 'apple'!

How does it work??

Everything in the list gets to have a turn at being the dog variable

```
fruits = ['apple', 'banana', 'mango']
for dog in fruits:
    print('yummy ' + dog)
```

```
>>> Yummy apple
```

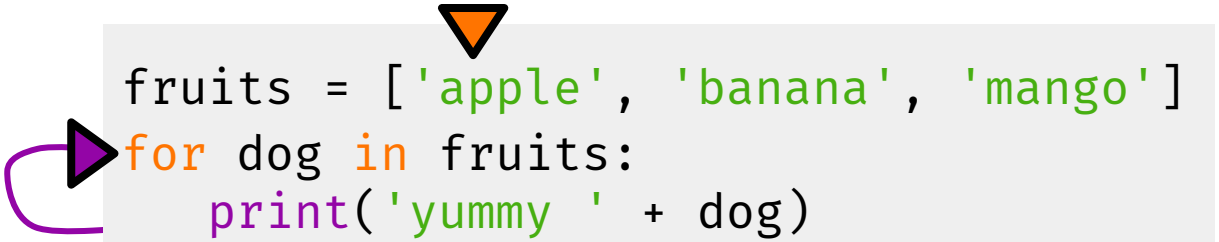
Let's set dog to to the first thing in the list!

dog is now 'apple'!

```
print('yummy ' + dog)
```

How does it work??

Everything in the list gets to have a turn at being the dog variable



```
fruits = ['apple', 'banana', 'mango']
for dog in fruits:
    print('yummy ' + dog)
```

```
>>> Yummy apple
```

Let's set dog to to the first thing in the list!

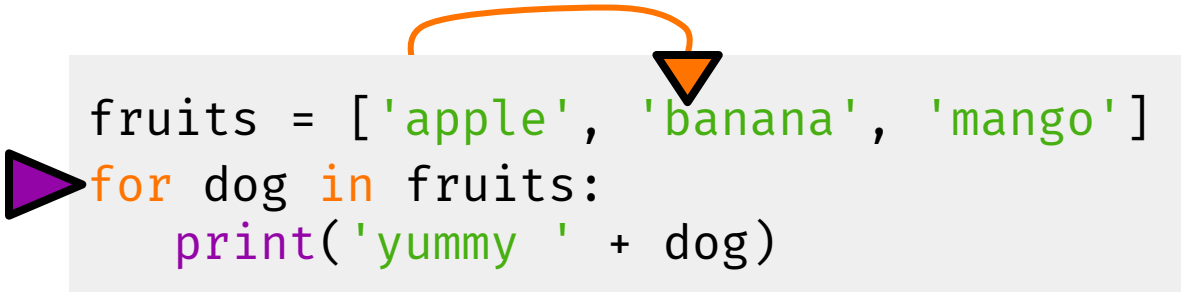
dog is now 'apple'!

```
print('yummy ' + dog)
```

We're at the end of the loop body, back to the top!

How does it work??

Everything in the list gets to have a turn at being the dog variable



```
fruits = ['apple', 'banana', 'mango']
for dog in fruits:
    print('yummy ' + dog)
```

```
>>> Yummy apple
```

Let's set dog to to the first thing in the list!
dog is now 'apple'!
`print('yummy ' + dog)`
We're at the end of the loop body, back to the top!

Let's set dog to to the next thing in the list!
dog is now 'banana'!

How does it work??

Everything in the list gets to have a turn at being the dog variable

```
fruits = ['apple', 'banana', 'mango']
for dog in fruits:
    print('yummy ' + dog)
```

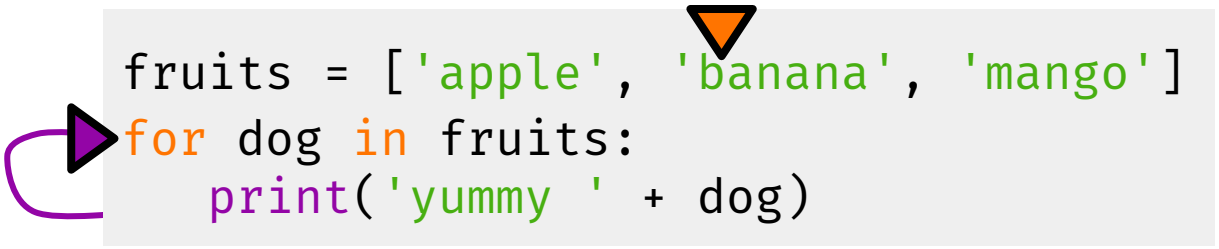
```
>>> Yummy apple
>>> Yummy banana
```

Let's set dog to to the first thing in the list!
dog is now 'apple'!
`print('yummy ' + dog)`
We're at the end of the loop body, back to the top!

Let's set dog to to the next thing in the list!
dog is now 'banana'!
`print('yummy ' + dog)`

How does it work??

Everything in the list gets to have a turn at being the dog variable



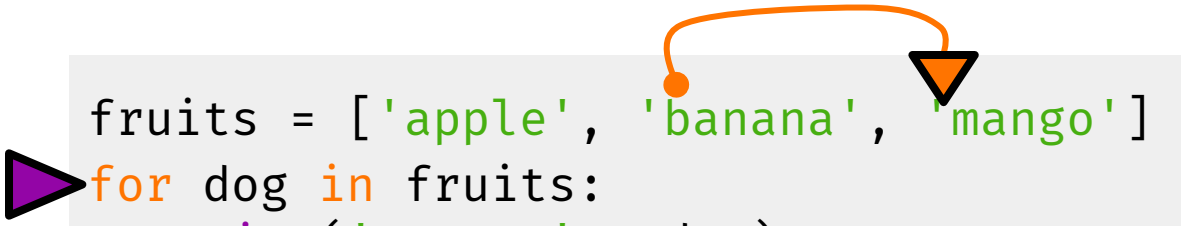

```
>>> Yummy apple
>>> Yummy banana
```

Let's set dog to to the first thing in the list!
dog is now 'apple'!
`print('yummy ' + dog)`
We're at the end of the loop body, back to the top!

Let's set dog to to the next thing in the list!
dog is now 'banana'!
`print('yummy ' + dog)`
Out of body, back to the top!

How does it work??

Everything in the list gets to have a turn at being the dog variable



```
fruits = ['apple', 'banana', 'mango']
for dog in fruits:
    print('yummy ' + dog)
```

```
>>> Yummy apple
>>> Yummy banana
```

Let's set dog to to the first thing in the list!
dog is now 'apple'!
print('yummy ' + dog)
We're at the end of the loop body, back to the top!

Let's set dog to to the next thing in the list!
dog is now 'banana'!
print('yummy ' + dog)
Out of body, back to the top!

Let's set dog to to the next thing in the list!
dog is now 'mango'!

How does it work??

Everything in the list gets to have a turn at being the dog variable

```
fruits = ['apple', 'banana', 'mango']
for dog in fruits:
    print('yummy ' + dog)
```

```
>>> Yummy apple
>>> Yummy banana
>>> Yummy mango
```

Let's set dog to to the first thing in the list!
dog is now 'apple'!
print('yummy ' + dog)
We're at the end of the loop body, back to the top!

Let's set dog to to the next thing in the list!
dog is now 'banana'!
print('yummy ' + dog)
Out of body, back to the top!

Let's set dog to to the next thing in the list!
dog is now 'mango'!
print('yummy ' + dog)

How does it work??

Everything in the list gets to have a turn at being the dog variable

```
fruits = ['apple', 'banana', 'mango']
for dog in fruits:
    print('yummy ' + dog)
```

```
>>> Yummy apple
>>> Yummy banana
>>> Yummy mango
```

Let's set dog to to the first thing in the list!
dog is now 'apple'!
print('yummy ' + dog)
We're at the end of the loop body, back to the top!

Let's set dog to to the next thing in the list!
dog is now 'banana'!
print('yummy ' + dog)
Out of body, back to the top!

Let's set dog to to the next thing in the list!
dog is now 'mango'!
print('yummy ' + dog)
Out of body, and out of list!!
We're done here!

Project Time!

Now you know how to use a for loop!

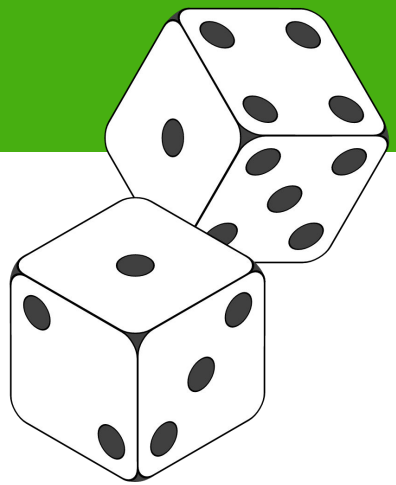
Try to do the next Parts

...if you are up for it!

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!

Raaaaaaaaaaandom! Can you handle that?

`Try to do the next Part!`

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!

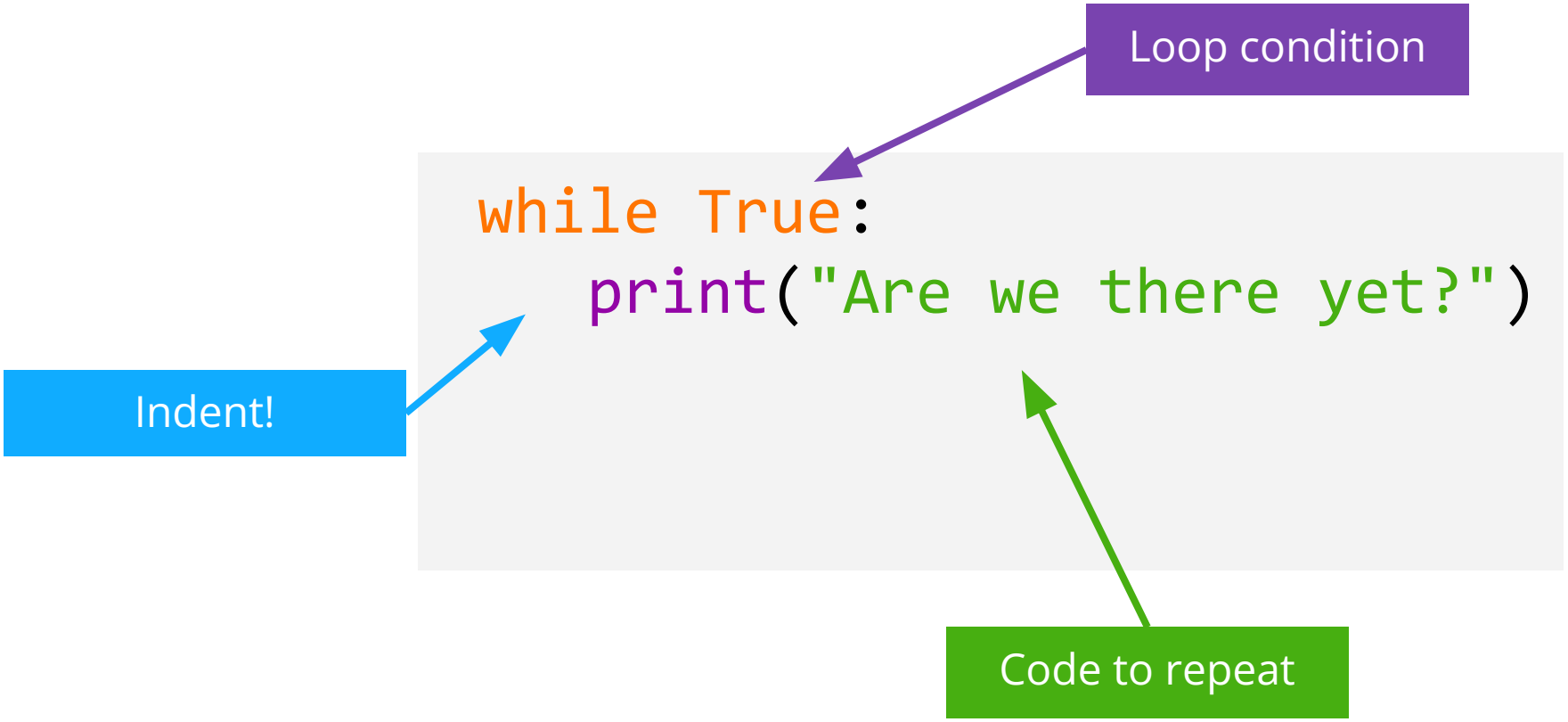
So we set a condition that is always True!

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:

Try to do the next Part

The tutors will be around to help!

Extension - Files

Filing it away!

What happens if we want to use different data in our program? What if that data is too big to write in with the keyboard?

We'd have to change our code!!

It would be better if we could keep all our data in a file and just be able to pick and choose what file we wanted to play today!

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!

```
with open("test.txt", "r") as f:
```

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:

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

You can read in one line at a time

You can use a for loop to read 1 line at a time!

```
with open("haiku.txt", "r") as f:
    for line in f:
        print(line)
```

```
Wanna go outside.

Oh NO! Help! I got outside!

Let me back inside!
```

Why is there an extra blank line each time?

Chomping off the newline

The newline character is represented by '\n':

```
print('Hello\nWorld')
Hello
World
```

We can remove it from the lines we read with .strip()

```
x = 'abc\n'
x.strip()
'abc'
```

x.strip() is safe as lines without newlines will be unaffected

Reading and stripping!

```
with open("haiku.txt", "r") as f:
    for line in f:
        line = line.strip()
        print(line)
```

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

No extra lines!

Project time!

I hope you **`file`**d that knowledge away

Use it in the next section of the project!

Try to do the next Part

The tutors will be around to help!

Tell us what you think!

Click on the
End of Day Form
and fill it in now!