

# Welcome to the Labs

Scissors Paper Rock!



# Thank you to our Sponsors!

Platinum Sponsor:



# Who are the tutors?



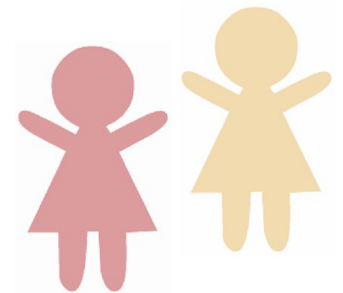
Who are you?



# Ultimate Scissors Paper Rock

1. Start with a partner
2. play scissors paper rock!

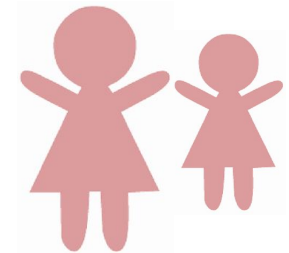
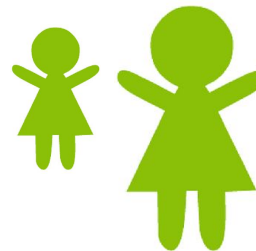
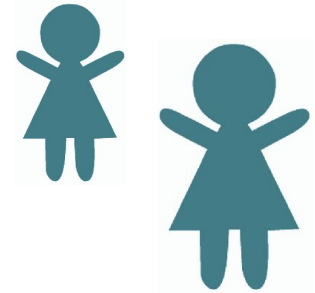
**Who will be the champion?**



# Ultimate Scissors Paper Rock

1. Start with a partner
2. play scissors paper rock!
3. If you win they become your cheer squad!  
And their squad becomes your squad!
4. Find a new partner!
5. Keep playing until there is only one person left!

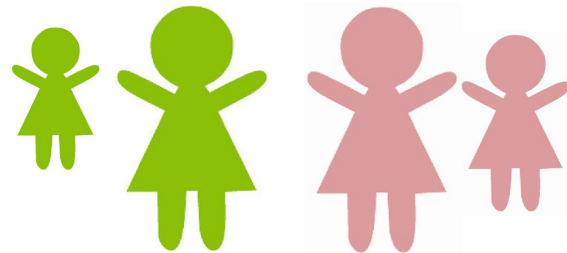
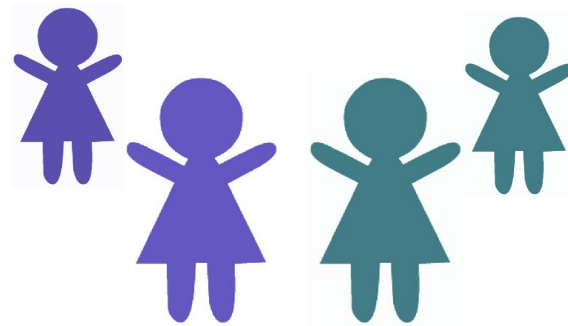
**Who will be the champion?**



# Ultimate Scissors Paper Rock

1. Start with a partner
2. play scissors paper rock!
3. If you win they become your cheer squad!  
And their squad becomes your squad!
4. Find a new partner!
5. Keep playing until there is only one person left!

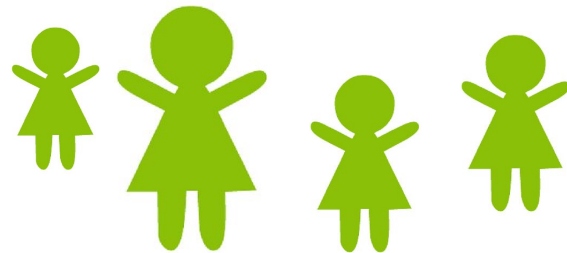
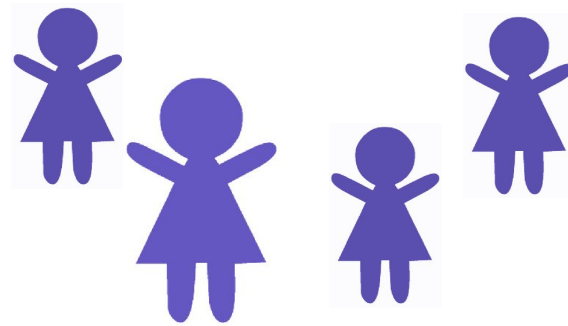
**Who will be the champion?**



# Ultimate Scissors Paper Rock

1. Start with a partner
2. play scissors paper rock!
3. If you win they become your cheer squad!  
And their squad becomes your squad!
4. Find a new partner!
5. Keep playing until there is only one person left!

**Who will be the champion?**

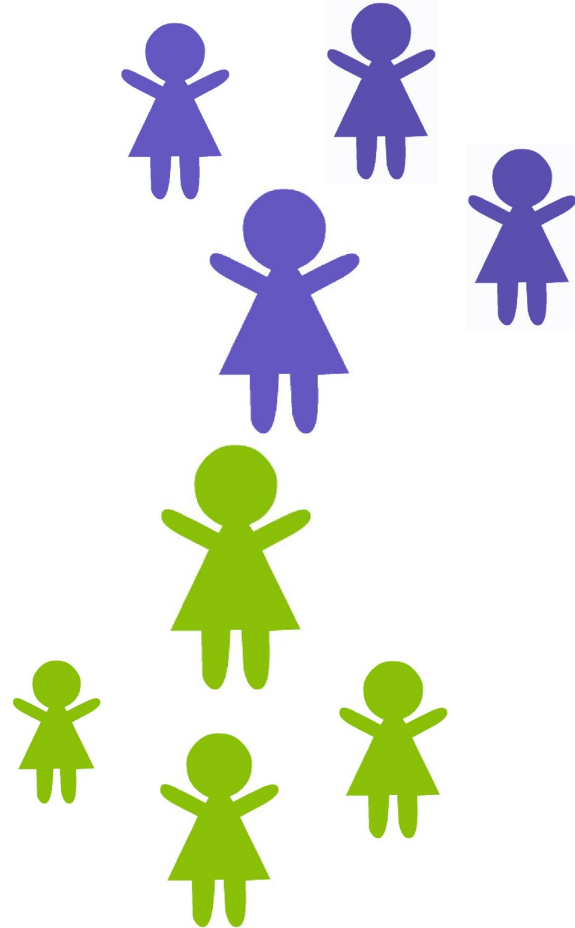




# Ultimate Scissors Paper Rock

1. Start with a partner
2. play scissors paper rock!
3. If you win they become your cheer squad!  
And their squad becomes your squad!
4. Find a new partner!
5. Keep playing until there is only one person left!

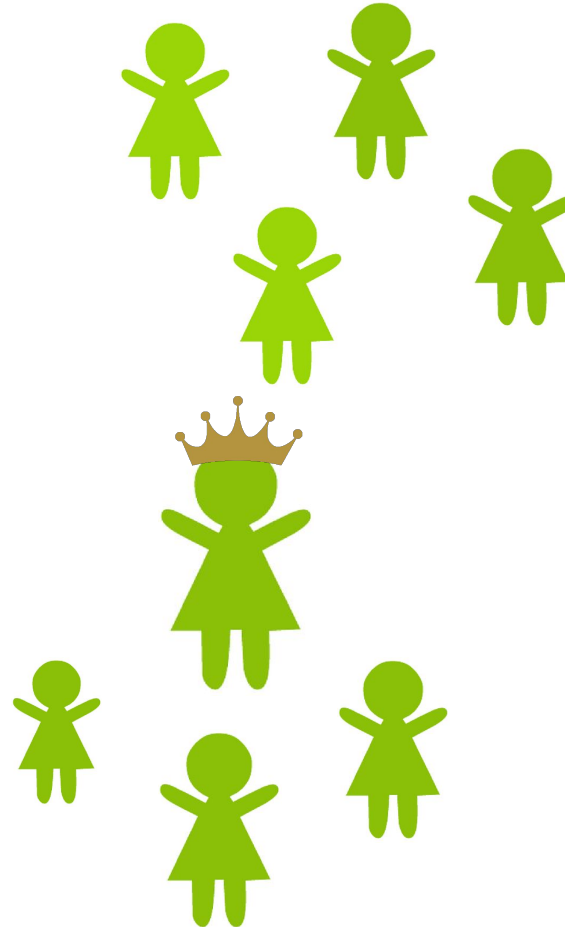
**Who will be the champion?**



# Ultimate Scissors Paper Rock

1. Start with a partner
2. play scissors paper rock!
3. If you win they become your cheer squad!  
And their squad becomes your squad!
4. Find a new partner!
5. Keep playing until there is only one person left!

**Who will be the champion?**



# Log on

## Log on and jump on the GPN website

[girlsprogramming.network/workshop](https://girlsprogramming.network/workshop)

You can see:

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

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



Tell us you're here!

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



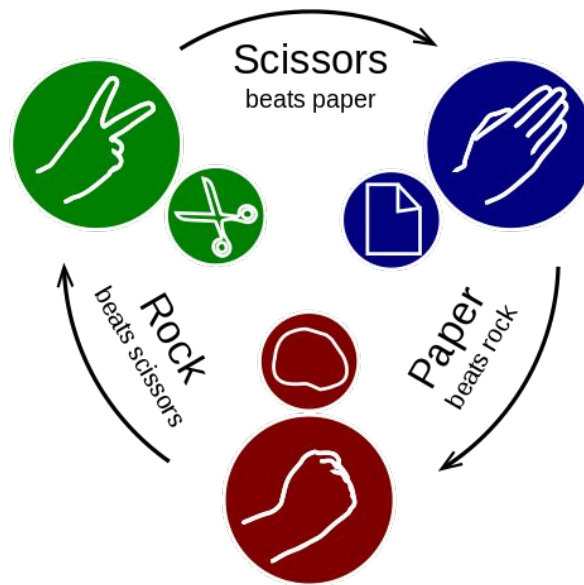
# Today's project!

Scissors Paper Rock!



# Best of Three?

Let's go round the room, and play some Scissors Paper Rock!



It's what we'll be programming today, so have a think about some of the actions required to play!

# Scissors Paper Rock

How did you go? Did you win?

Some of the things that we need to do to play scissors paper rock include:

- We have to select a move (out of scissors, paper and rock)
- Our opponent has to select a move
- We need to know what combinations of move result in win, lose or tie.
- We need to compare our moves to see who won!
- We have to congratulate the winner!

We'll be programming these actions today! Our opponent is going to be the computer.



# Using the workbook!

The workbooks will help you put your project together!

Each **Part** of the workbook is made of tasks!

## Tasks - The parts of your project

Follow the tasks **in order** to make the project!

## Hints - Helpers for your tasks!

Stuck on a task, we might have given you a hint to help you **figure it out**!

The hints have **unrelated** examples, or tips. **Don't copy and paste** in the code, you'll end up with something **CRAZY**!

### Task 6.2: Add a blah to your code!

This has instructions on how to do a part of the project

1. **Start by doing this part**
2. **Then you can do this part**

### Task 6.1: Make the thing do blah!

Make your project do blah ....

#### Hint

A clue, an example or some extra information to help you **figure out** the answer.

```
print('This example is not part of the project' )
```





# Using the workbook!

The workbooks will help you put your project together!

Check off before you move on from a **Part!** Do some bonuses while you wait!

## Checklist - Am I done yet?

Make sure you can tick off every box in this section before you go to the next Part.

## Lecture Markers

This tells you you'll find out how to do things for this section during the names lecture.

## Bonus Activities

Stuck waiting at a lecture marker? Try a purple bonus. They add extra functionality to your project along the way.



## CHECKPOINT



If you can tick all of these off you're ready to move the next part!

- ☐ Your program does blah
- ☐ Your program does blob



## ★ BONUS 4.3: Do some extra!

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



# Classes



# What is an object?

## What do you think an object is?



# What is an object?

## What do you think an object is?



# What is an object?

What do you think an object is?



# What is an object?

What do you think an object is?



# What is an object?

What do you think an object is?



# What is an object?

What do you think an object is?





# What is an object in code?

An object is something that we know information about and that can sometimes do things



# What is an object in code?

An object is something that we know information about and that can sometimes do things

Like a cat!



# What is an object in code?

An object is something that we know information about and that can sometimes do things

Like a cat!



What information might we know about a cat?

# What is an object in code?

An object is something that we know information about and that can sometimes do things

Like a cat!



What information might we know about a cat?

**Name**

# What is an object in code?

An object is something that we know information about and that can sometimes do things

Like a cat!



What information might we know about a cat?

**Name**

**Age**

# What is an object in code?

An object is something that we know information about and that can sometimes do things

Like a cat!



What information might we know about a cat?

**Name**

**Age**

**Colour**

# What is an object in code?

An object is something that we know information about and that can sometimes do things

Like a cat!



What information might we know about a cat?

**Name**

**Owner**

**Age**

**Colour**



# What is an object in code?

An object is something that we know information about and that can sometimes do things

Like a cat!



What information might we know about a cat?

**Name**

**Owner**

**Age**

**Weight**

**Colour**





# What is an object in code?

An object is something that we know information about and that can sometimes do things

Like a cat!



What information might we know about a cat?

**Name**

**Owner**

**Age**

**Weight**

**Colour**

**Microchip #**



# What is an object in code?

An object is something that we know information about and that can sometimes do things

Like a cat!

What things might a cat do?



# What is an object in code?

An object is something that we know information about and that can sometimes do things

Like a cat!



What things might a cat do?

**Meow**

# What is an object in code?

An object is something that we know information about and that can sometimes do things

Like a cat!



What things might a cat do?

**Meow**

**Eat**

# What is an object in code?

An object is something that we know information about and that can sometimes do things

Like a cat!



What things might a cat do?

**Meow**

**Eat**

**Scratch**



# What is an object in code?

An object is something that we know information about and that can sometimes do things

Like a cat!



What things might a cat do?

**Meow**

**Sleep**

**Eat**

**Scratch**



# What is an object in code?

An object is something that we know information about and that can sometimes do things

Like a cat!



What things might a cat do?

**Meow**

**Sleep**

**Eat**

**Purr**

**Scratch**



# What is an object in code?

An object is something that we know information about and that can sometimes do things

Like a cat!



What things might a cat do?

**Meow**

**Sleep**

**Eat**

**Purr**

**Scratch**

**Jump**





# What does that look like in Python?

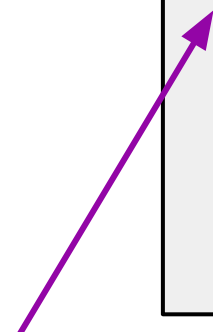
Let's have a look at how we might make a Cat object in Python code!



# What does that look like in Python?

Let's have a look at how we might make a Cat object in Python code!

```
class Cat():  
    def __init__(self, name, age, colour):  
        self.name = name  
        self.age = age  
        self.colour = colour
```



Here we tell python that we are making a new type (or class) of object called Cat

# What does that look like in Python?

Let's have a look at how we might make a Cat object in Python code!

`__init__` is how we tell Python how to make a new Cat

```
class Cat():  
    def __init__(self, name, age, colour):  
        self.name = name  
        self.age = age  
        self.colour = colour
```

# What does that look like in Python?

Let's have a look at how we might make a Cat object in Python code!

```
class Cat():  
    def __init__(self, name, age, colour):  
        self.name = name  
        self.age = age  
        self.colour = colour
```

Here we tell Python what information we need to know about the Cat


Note: self is special and we always need it



# What does that look like in Python?

Let's have a look at how we might make a Cat object in Python code!

```
class Cat():  
    def __init__(self, name, age, colour):  
        self.name = name  
        self.age = age  
        self.colour = colour
```



Here we save the  
information we got so  
we can use it again

# What does that look like in Python?

## How do we make a new Cat?

```
class Cat():  
    def __init__(self, name, age, colour):  
        self.name = name  
        self.age = age  
        self.colour = colour  
  
emmy = Cat("Emmy", 3, "Dark brown")
```

# What does that look like in Python?

What does this print out?

```
class Cat():  
    def __init__(self, name, age, colour):  
        self.name = name  
        self.age = age  
        self.colour = colour  
  
emmy = Cat("Emmy", 3, "Dark brown")  
print(emmy.name)  
print(emmy.age)  
print(emmy.colour)
```



# What does that look like in Python?

What does this print out?

```
class Cat():  
    def __init__(self, name, age, colour):  
        self.name = name  
        self.age = age  
        self.colour = colour  
  
emmy = Cat("Emmy", 3, "Dark brown")  
print(emmy.name)  
print(emmy.age)  
print(emmy.colour)
```

Emmy

3

Dark Brown





# What about doing things?

We said an object was something with information that could sometimes do things. Our Cat object doesn't do anything right now - let's add a way for it to meow!



# What about doing things?

We said an object was something with information that could sometimes do things. Our Cat object doesn't do anything right now - let's add a way for it to meow!

```
class Cat():  
    def __init__(self, name, age, colour):  
        self.name = name  
        self.age = age  
        self.colour = colour  
  
    def meow(self):  
        print("Meow")
```

# What about doing things?

What does this code do?

```
class Cat():  
    def __init__(self, name, age, colour):  
        self.name = name  
        self.age = age  
        self.colour = colour  
  
    def meow(self):  
        print("Meow")  
  
emmy = Cat("Emmy", 3, "Dark brown")  
emmy.meow()
```



# What about doing things?

What does this code do?

```
class Cat():  
    def __init__(self, name, age, colour):  
        self.name = name  
        self.age = age  
        self.colour = colour  
  
    def meow(self):  
        print("Meow")  
  
emmy = Cat("Emmy", 3, "Dark brown")  
emmy.meow()
```

Meow



# What else can it do?

Let's have our cat have a Birthday that makes it get older by 1 year!



# What else can it do?

Let's have our cat have a Birthday that makes it get older by 1 year!

```
class Cat():  
    def __init__(self, name, age, colour):  
        self.name = name  
        self.age = age  
        self.colour = colour  
  
    def meow(self):  
        print("Meow")  
  
    def birthday(self):  
        self.age = self.age + 1
```



# What else can it do?

## What does this code do?

```
class Cat():  
    def __init__(self, name, age, colour):  
        self.name = name  
        self.age = age  
        self.colour = colour  
  
    def meow(self):  
        print("Meow")  
  
    def birthday(self):  
        self.age = self.age + 1  
  
emmy = Cat("Emmy", 3, "Dark brown")  
emmy.birthday()  
print(emmy.age)
```



# What else can it do?

## What does this code do?

```
class Cat():
    def __init__(self, name, age, colour):
        self.name = name
        self.age = age
        self.colour = colour

    def meow(self):
        print("Meow")

    def birthday(self):
        self.age = self.age + 1

emmy = Cat("Emmy", 3, "Dark brown")
emmy.birthday()
print(emmy.age)
```

4





# I have more than 1 cat!

Emmy has a little sister, Saphira! Let's add her to our code too!

```
cat1 = Cat("Emmy", 3, "Dark brown")  
cat2 = Cat("Saphira", 1, "Grey")
```



# Cat Crime!

There has been a cat crime!

One of the cats has gotten on the kitchen counter and eaten some of my lunch!

They both look innocent but they left a hair behind at the scene of the crime! Let's write some code to work out who did it



# Cat Crime

Who did it??

```
cat1 = Cat("Emmy", 3, "Dark brown")
cat2 = Cat("Saphira", 1, "Grey")

hair_colour = "Grey"

if hair_colour == cat1.colour:
    print("That hair belongs to", cat1.name)
elif hair_colour == cat2.colour:
    print("That hair belongs to", cat2.name)
```



# Cat Crime

Who did it??

```
cat1 = Cat("Emmy", 3, "Dark brown")
cat2 = Cat("Saphira", 1, "Grey")

hair_colour = "Grey"

if hair_colour == cat1.colour:
    print("That hair belongs to", cat1.name)
elif hair_colour == cat2.colour:
    print("That hair belongs to", cat2.name)
```

That hair belongs to Saphira



# Project time!

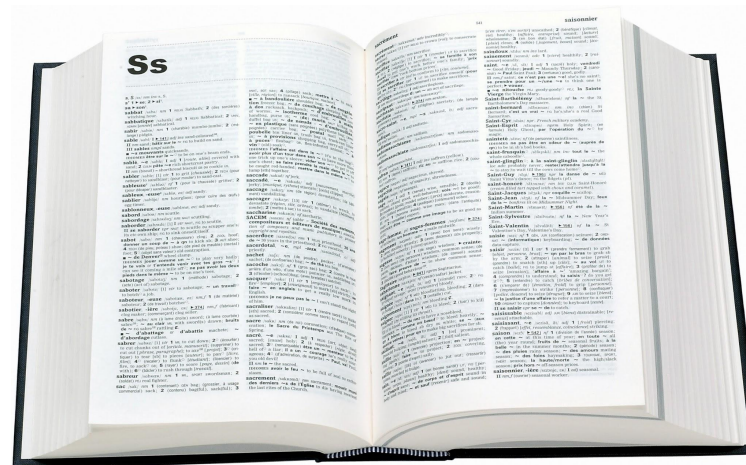
You now know all about **classes**!

**Let's put what we learnt into our project**  
**Try to do Parts 0-2**

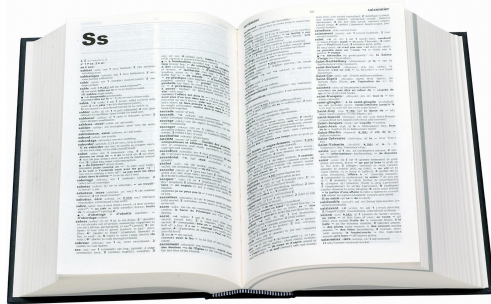
The tutors will be around to help!



# Dictionaries



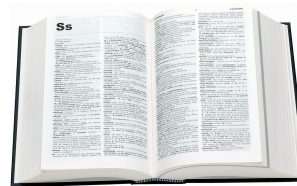
# Dictionaries!



***You know dictionaries!***

**They're great at looking up thing  
by a word, not a position in a list!**

Look up  
***Hello***



**Get back**

***A greeting (salutation) said  
when meeting someone or  
acknowledging someone's  
arrival or presence.***



# Looking it up!

**There are lots of times we want to look something up!**



**Competition registration**

Team Name → List of team members



**Phone Book**

Name → Phone number



**Vending Machine**

Treat Name → Price





Looking it up!



## Phone Book

Name → Phone number

↑  
Key

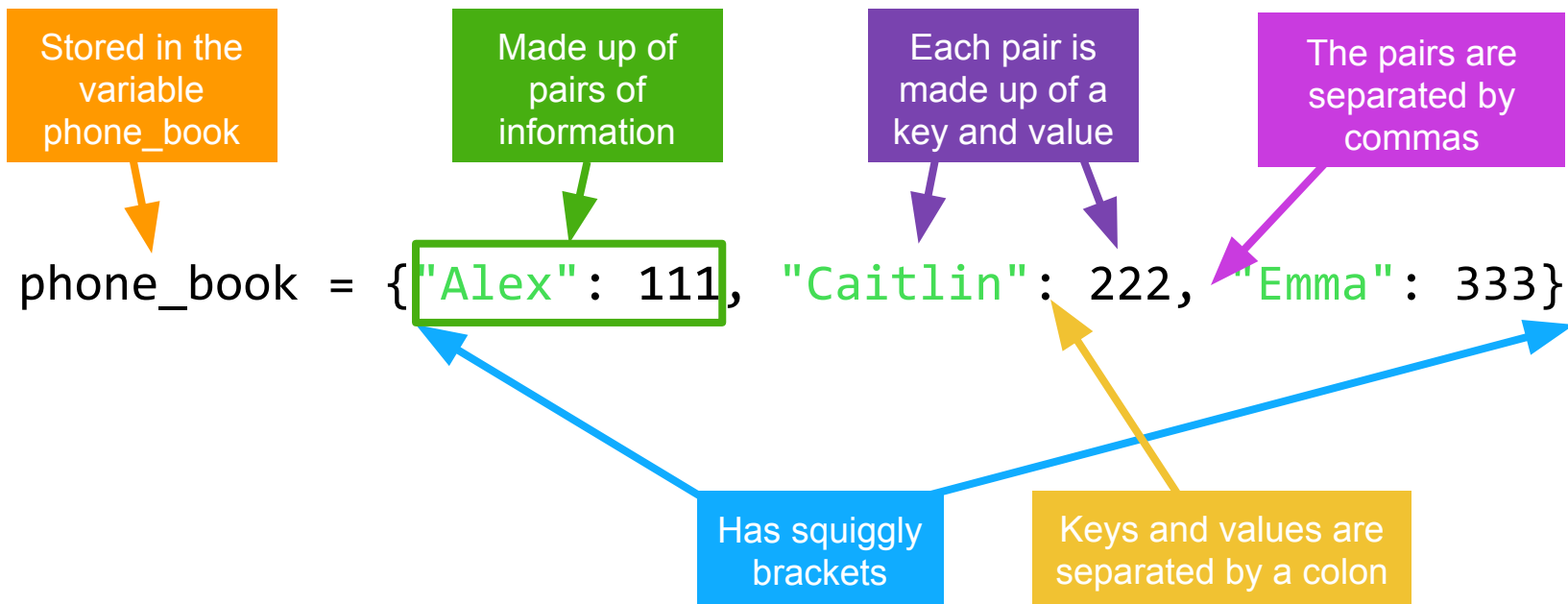
↑  
Value

**We can use a dictionary for anything with a  
key → value pattern!**



# Dictionaries anatomy!

**This is a python dictionary!**



**This dictionary has Alex, Caitlin and Emma's phone numbers**



# Playing with dictionaries!

Let's look at an example!

```
phone_book = {"Alex": 111, "Caitlin": 222, "Emma": 333}
```

1. What happens?

```
phone_book["Alex"]
```

2. How would you look up Emma's phone number?

3. Look up the name of someone who is not in the phone book? What happens?



# Playing with dictionaries!

Let's look at an example!

```
phone_book = {"Alex": 111, "Caitlin": 222, "Emma": 333}
```

1. What happens?

```
phone_book["Alex"]
```

111

2. How would you look up Emma's phone number?

3. Look up the name of someone who is not in the phone book? What happens?



# Playing with dictionaries!

Let's look at an example!

```
phone_book = {"Alex": 111, "Caitlin": 222, "Emma": 333}
```

1. What happens?

```
phone_book["Alex"]
```

111

2. How would you look up Emma's phone number?

```
phone_book["Emma"]
```

3. Look up the name of someone who is not in the phone book? What happens?



# Playing with dictionaries!

Let's look at an example!

```
phone_book = {"Alex": 111, "Caitlin": 222, "Emma": 333}
```

1. What happens?

```
phone_book["Alex"]
```

111

2. How would you look up Emma's phone number?

```
phone_book["Emma"]
```

3. Look up the name of someone who is not in the phone book? What happens?

**KeyError**



# Tuples!

## Some data sticks together!

Tuples are like lists that you can't edit or add too!

**It's a:**

- **list of items**
- **in round brackets**
- **separated by commas**

**Tuples are a way of grouping data!**

("January", "1st")

("December", "25th")

("April", "25th")



# Tuples in dictionaries!

## We can use tuples as the key to a dictionary

```
holidays = {("January", "1st"): "New Years",  
             ("December", "25th"): "Christmas Day",  
             ("April", "25th"): "ANZAC Day"}
```

1. What about this: `holidays[("January", "1st")]`
2. How would you look up what happens on the 25th of April
3. What happens if you we do: `phone_book[("25th", "December")]`



# Tuples in dictionaries!

## We can use tuples as the key to a dictionary

```
holidays = {("January", "1st"): "New Years",  
             ("December", "25th"): "Christmas Day",  
             ("April", "25th"): "ANZAC Day"}
```

1. What about this: `holidays[("January", "1st")]`  
**New Years**
2. How would you look up what happens on the 25th of April
3. What happens if you we do: `phone_book[("25th", "December")]`

# Tuples in dictionaries!

## We can use tuples as the key to a dictionary

```
holidays = {("January", "1st"): "New Years",  
             ("December", "25th"): "Christmas Day",  
             ("April", "25th"): "ANZAC Day"}
```

1. What about this: `holidays[("January", "1st")]`  
New Years

2. How would you look up what happens on the 25th of April

```
holidays[("April", "25th")]
```

3. What happens if you we do: `phone_book[("25th", "December")]`



# Tuples in dictionaries!

## We can use tuples as the key to a dictionary

```
holidays = {("January", "1st"): "New Years",  
             ("December", "25th"): "Christmas Day",  
             ("April", "25th"): "ANZAC Day"}
```

1. What about this: `holidays[("January", "1st")]`  
**New Years**

2. How would you look up what happens on the 25th of April

```
holidays[("April", "25th")]
```

3. What happens if you we do: `phone_book[("25th", "December")]`  
**KeyError**



# Project time!

You now know all about dictionaries!

**Let's put what we learnt into our project**  
**Try to do Part 3**

The tutors will be around to help!

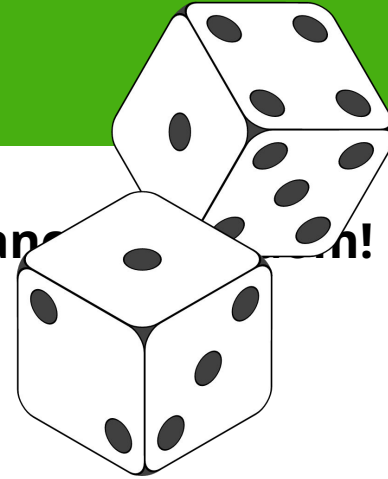


Random!



# That's so random!

There's lots of things in life that are up to chance!



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!

**Here's an example!**

**1. Import the random module!**

```
>>> import random
```

**2. Copy the shopping list into IDLE**

```
>>> shopping_list = ["eggs", "bread", "apples", "milk"]
```

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





# Project Time!

**Raaaaaaaaaandom! Can you handle that?**

**Let's put what we learnt into our project**

**Try to do Part 4**

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])
```

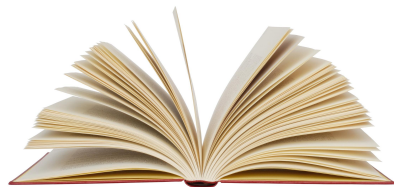
**What if it had a 100 items??? That would be BORING!**



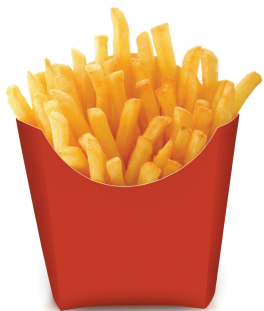
# For Loops

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**



**For each chip in this bag of chips:  
Eat**



# Looping over a list of ints

## We can loop through a list:

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

What's going to happen?



# Looping over a list of ints

## We can loop through a list:

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

What's going to happen?

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

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




# 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)
```

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

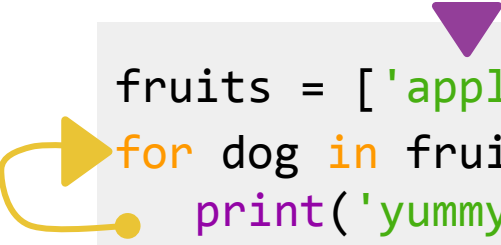
dog is now 'apple'!  
print('yummy ' + dog)

>>>Yummy 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)
```

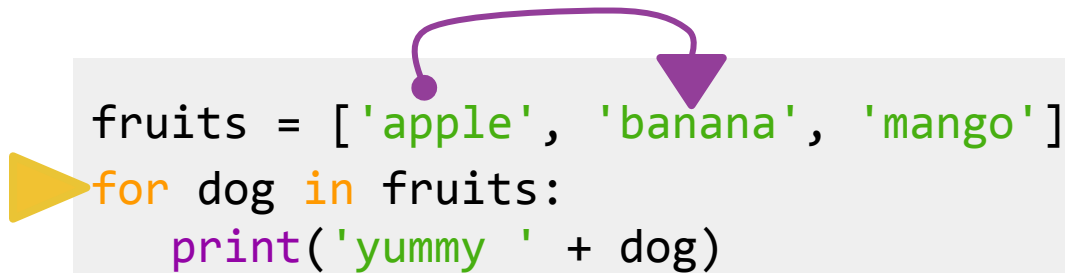
```
>>>Yummy apple
```

Let's set dog to to the **first** thing in the list!  
dog is now 'apple'!  
`print('yummy ' + dog)`  
***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
```

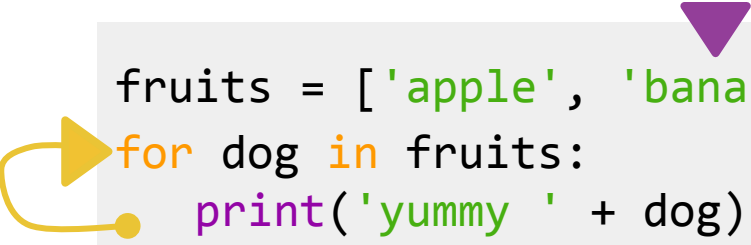
Let's set dog to to the **first** thing in the list!  
dog is now 'apple'!  
print('yummy ' + dog)  
*We're at the end of the loop body, back to the top!*

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



```
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'!

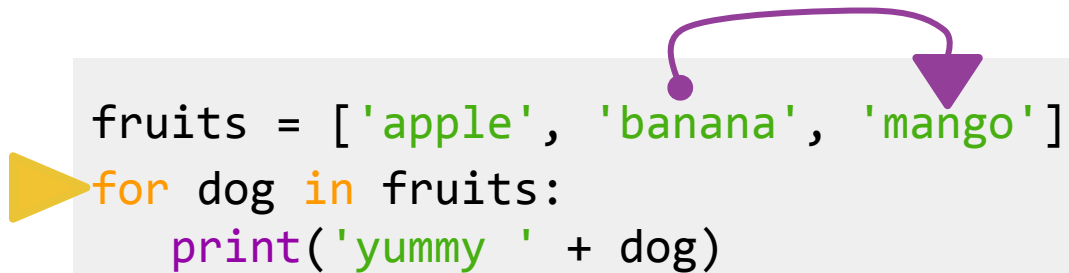
```
print('yummy ' + dog)
```

*Out of body, back to the top!*



# 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!*





# Generating a List!

**Sometimes you don't care about what is in the list!**

You just want to repeat 10 times or a 1000 times!

**Doing this is boring.....**

```
numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

**But python will make a list of things for you!**

**Try this!**

```
for num in range(50):  
    print(num)
```



# Project Time!

**Now you know how to use a for loop!**

**Try to do Part 5 & 6**

**...if you are up **for** it!**

And Extension parts 7-10

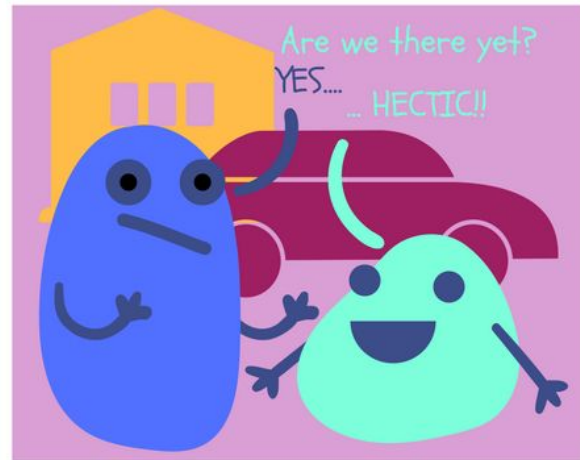
The tutors will be around to help!



# 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
```

MY VARIABLES

i = 0

Set the  
variable





# Introducing ... while loops!

## One step at a time!

0 is less  
than 3!

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

MY VARIABLES

i = 0



# Introducing ... while loops!

## One step at a time!

Print!

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

i is 0

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 = 1
```

UPDATE  
TIME!

```
i is 0
```



# Introducing ... while loops!

## One step at a time!

Take it  
from the  
top!

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

```
i is 0
```

### MY VARIABLES

```
i = 0
i = 1
```



# Introducing ... while loops!

## One step at a time!

i is less  
than 3!

```
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!

Print!

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

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

### MY VARIABLES


```
i = 0
i = 1
```



# Introducing ... while loops!

## One step at a time!

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



### MY VARIABLES

```
i = 0
i = 1
i = 2
```

UPDATE  
TIME!

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



# Introducing ... while loops!

## One step at a time!

Take it  
from the  
top!

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

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

### MY VARIABLES

```
i = 0
i = 1
i = 2
```





# Introducing ... while loops!

## One step at a time!

2 is less  
than 3!

```
◆ i = 0
  while i < 3:
    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!

Print!

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

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

### MY VARIABLES


```
i = 0
i = 1
i = 2
```



# Introducing ... while loops!

## One step at a time!

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



### MY VARIABLES

```
i = 0
i = 1
i = 2
i = 3
```

UPDATE  
TIME!

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



# Introducing ... while loops!

## One step at a time!

Take it  
from the  
top!

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

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

### MY VARIABLES

```
i = 0
i = 1
i = 2
i = 3
```



# 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  
done  
with this  
loop!

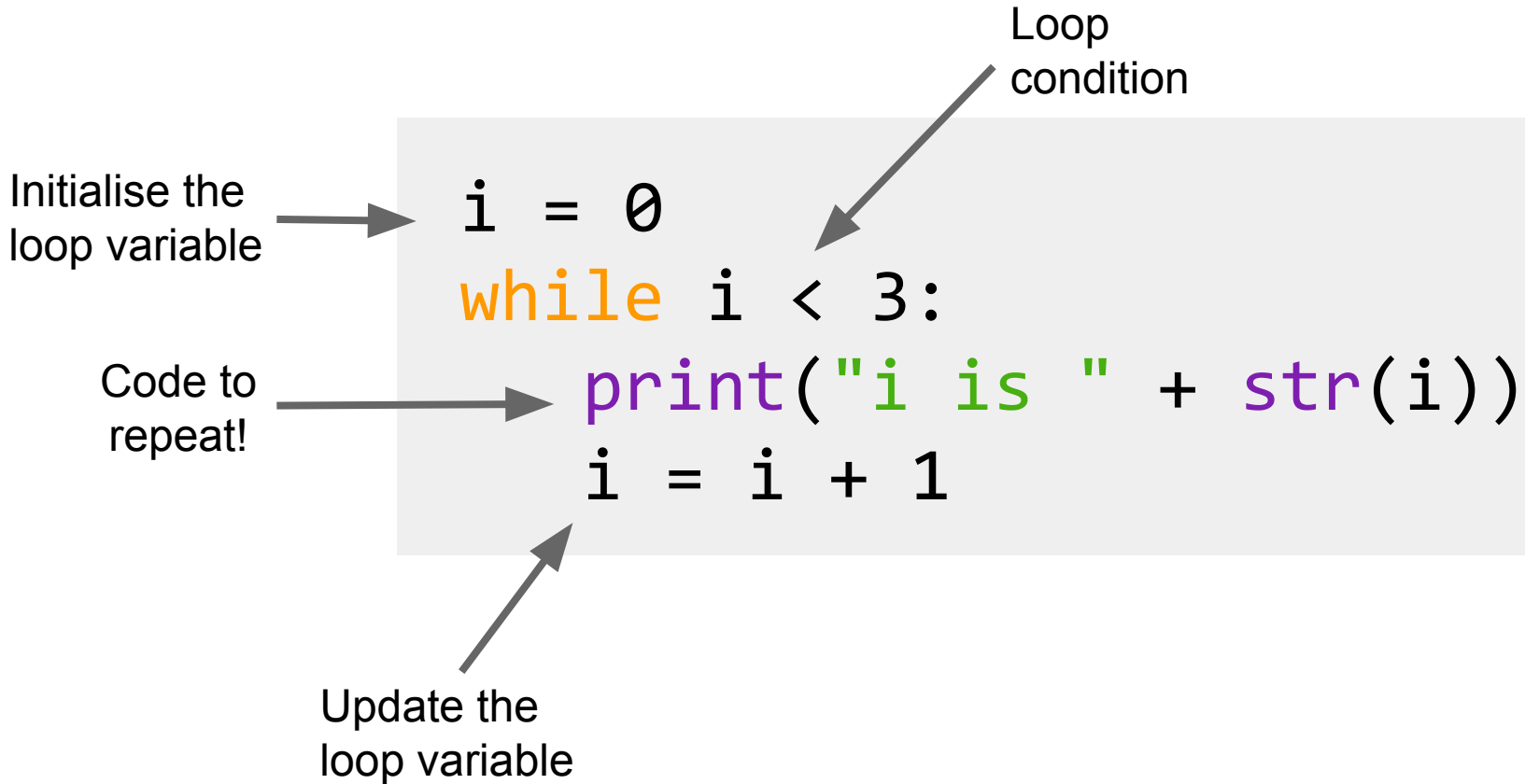
```
i is 0
i is 1
i is 2
```

### MY VARIABLES

```
i = 0
i = 1
i = 2
i = 3
```



# Introducing ... while loops!



The diagram illustrates the components of a while loop. It features a light gray rectangular box containing the following Python code:

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

Four annotations with arrows point to specific parts of the code:

- Initialise the loop variable**: Points to the line `i = 0`.
- Loop condition**: Points to the condition `i < 3:` in the `while` statement.
- Code to repeat!**: Points to the `print("i is " + str(i))` line inside the loop.
- Update the loop variable**: Points to the line `i = i + 1` inside the loop.

# 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

i is 0





# Infinite loop!

## Sometimes we want our loop to go forever!

So we set a condition that is always True!

We can even just write True!

```
while True:  
    print("Are we there yet?")
```



# Project Time!

**while** we're here:

**Try to do Part 7!**  
And the extensions

The tutors will be around to help!



Tell us what you think!

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