

# Welcome to the Labs

Scissors Paper Rock!

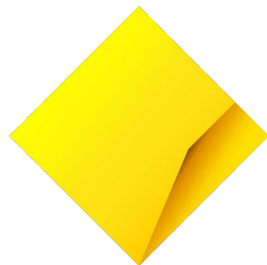


# Thank you to our Sponsors!

Platinum Sponsor:



Gold Sponsor:



**Commonwealth  
Bank**



# Who are the tutors?



Who are you?



# Log on

## Log on and jump on the GPN website

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

Click Content for your room. You can see:

- These **slides** (to take a look back or go on ahead).
- A link to our **workbook** in EdStem.
- A **cheatsheet** of python shortcuts!

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!

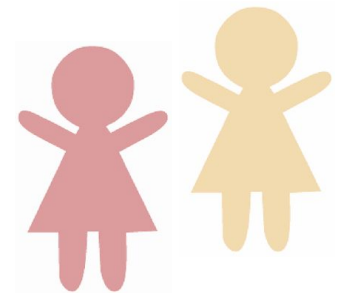
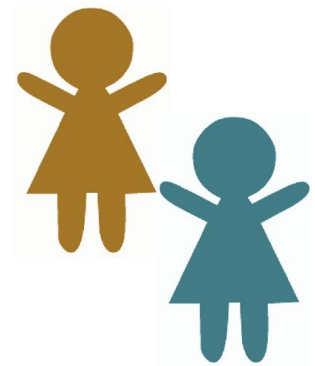
## Scissors Paper Rock



# Ultimate Scissors Paper Rock

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

**Who will be the champion?**

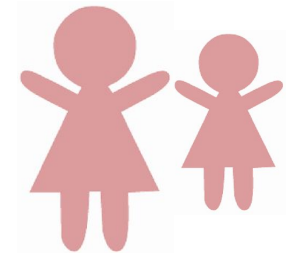
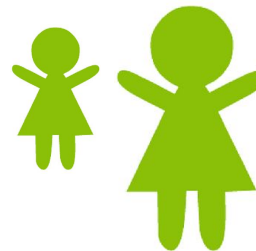
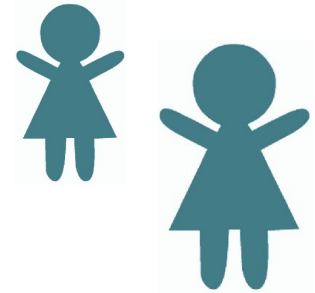




# Ultimate Scissors Paper Rock

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!

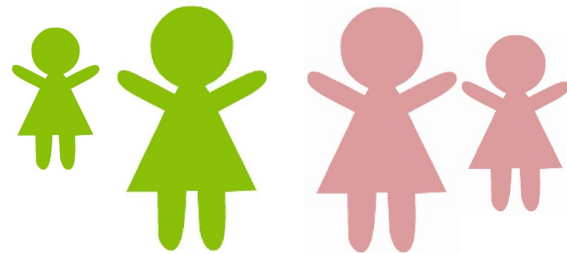
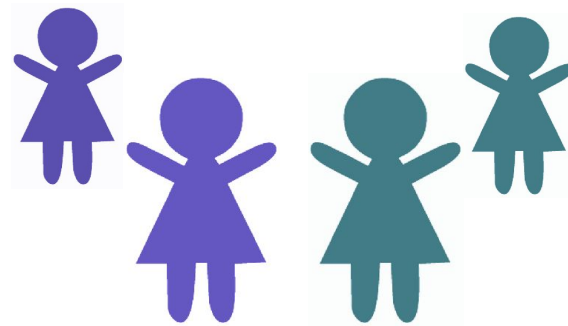
**Who will be the champion?**



# 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!

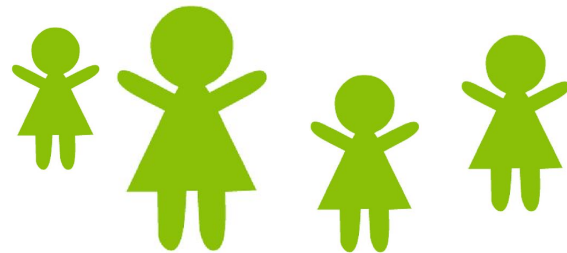
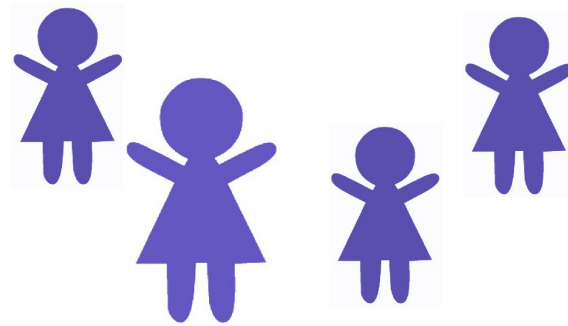
**Who will be the champion?**



# 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!

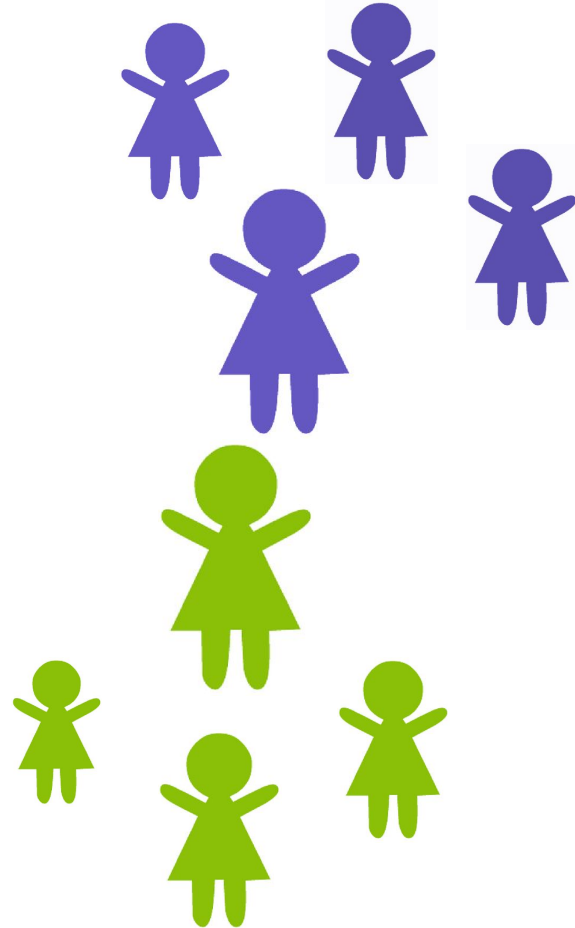
**Who will be the champion?**



# 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!

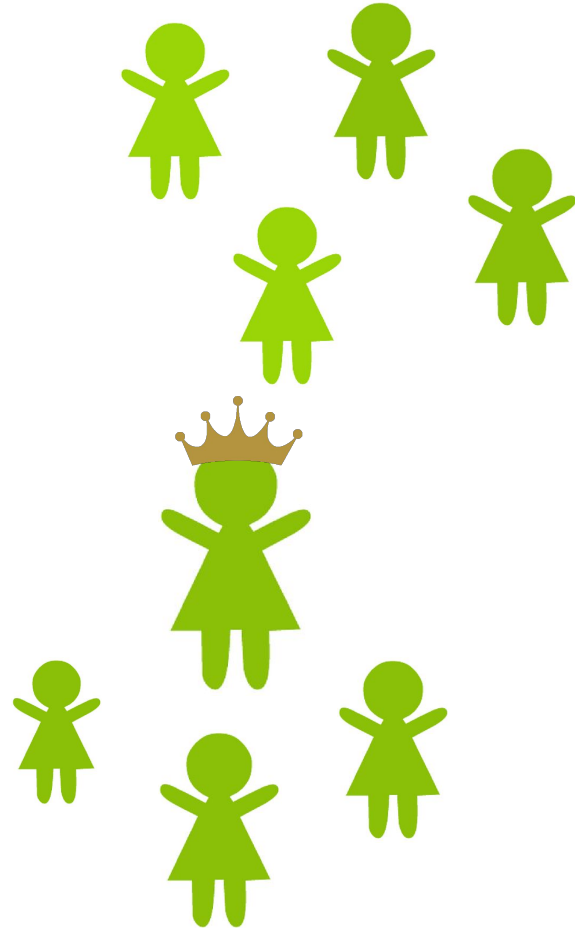
**Who will be the champion?**



# 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?**



# 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 moves 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.



# Intro to programming



# What is programming?



**Programming is not a bunch of crazy numbers!**

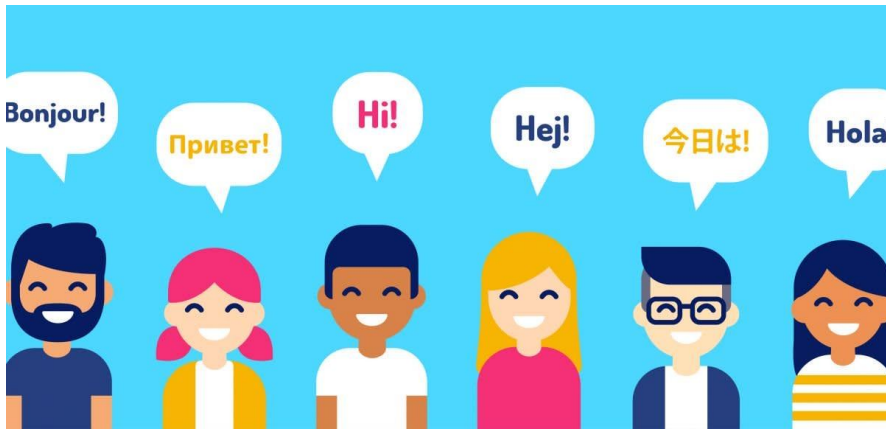
**It's giving computers a set of instructions!**





# A special language

Humans have languages like English, French, Spanish, Mandarin



[https://images.saymedia-content.com/.image/t\\_share/MTc0MTAyNzI3ODUxMjU1MjQx/how-to-easily-learn-a-language.jpg](https://images.saymedia-content.com/.image/t_share/MTc0MTAyNzI3ODUxMjU1MjQx/how-to-easily-learn-a-language.jpg)

And computers have languages like Python, Java, C and PHP



# Problem solving

Programming is how we get computers to solve complicated problems for us, saving us both time and effort!

This might be solving maths problems or counting words in a paragraph!



# People are smart, computers are dumb!

Computers do exactly what they're told. They follow instructions given to them in order, just like a cook following a recipe.



If the instructions are not in the correct order, we will end up with a mess!

# Everyone/thing has strengths!



- Incomplete instructions are okay - we can fill in the blanks!
- Improves everyday



- Incomplete instructions are not okay
- Improves when you tell it how to

# Intro to Python

Let's get coding!



# Signing up to Edstem

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

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

1. Follow the link on the website [www.girlsprogramming.network/workshop](http://www.girlsprogramming.network/workshop)
2. Type in your name and your personal email address
3. Click Create Account
4. Go to your email to verify your account
5. Create a password
6. It should then take you to the courses home page.

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


# Getting to the lessons

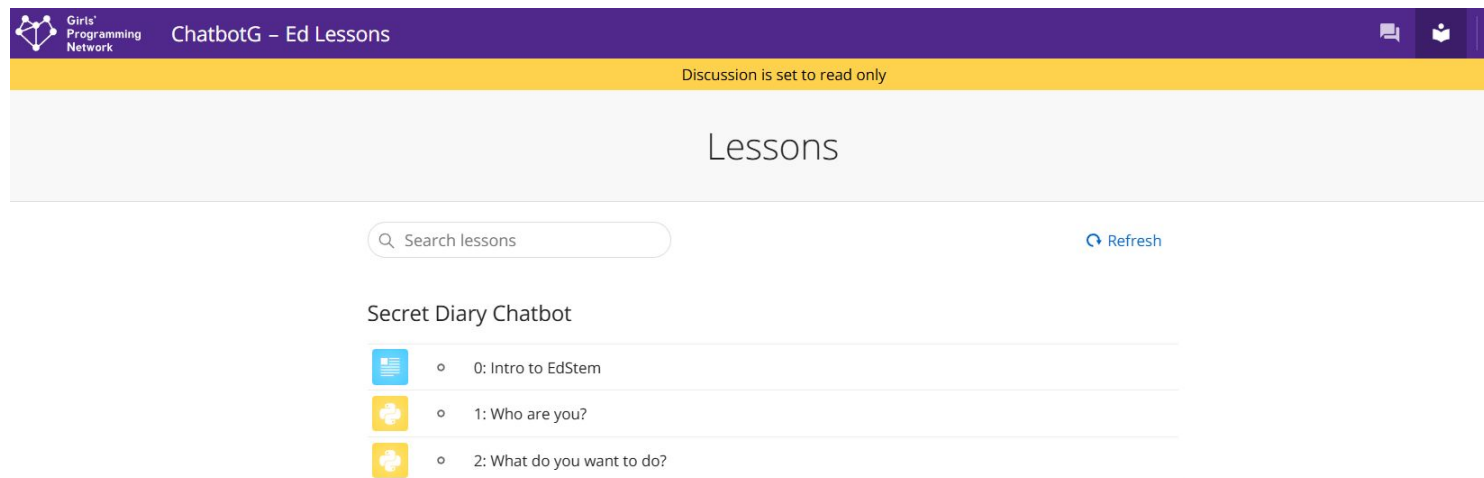
1. Once you are in the course, you'll be taken to a discussion page.
2. Click the button for the lessons page (top right - looks like a book)



# The Anatomy of the workbook

## The main page:

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





# Inside a Chapter

Inside a chapter there are two main types of pages:

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

They have this icon:



2. **Checkpoints**



Checkpoint

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

≡ 2: What do you want to do?

<> 2.1 Welcome to the Secret Diary

<> 2.2 What do you want to write?

<> 2.3 Do you want to read?

<> 2.4 I don't understand!

 Checkpoint

# How to do the work

In each lesson there is:

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

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

There are also  
Hints and  
Code Blocks to  
help you



**Description**

### Example Lesson Page

This is an example lesson page. This is where you're instructions and hints will go in each of your lessons.

To test out how to write code in this new online workbook...

1. First, print `"Hello EdStem!"`
2. Then run your code in the terminal. Remember you will need to enter the code `python chatbot.py`

**Hint:** You can print using the code;

**Run** PYTHON

```
1 print("Hello World")
```

**Files** chatbot.py

```
1 # Put your testing code here!!!
2 print("Hello EdStem!")
```

/home/chatbot.py Spaces: 4 (Auto) All changes saved

**Terminal**



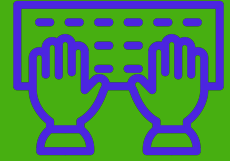
# Some shortcuts...

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

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



# Need help with EdStem?



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

**It's called 0: Intro to EdStem**

## Secret Diary Chatbot



◦ 0: Intro to EdStem

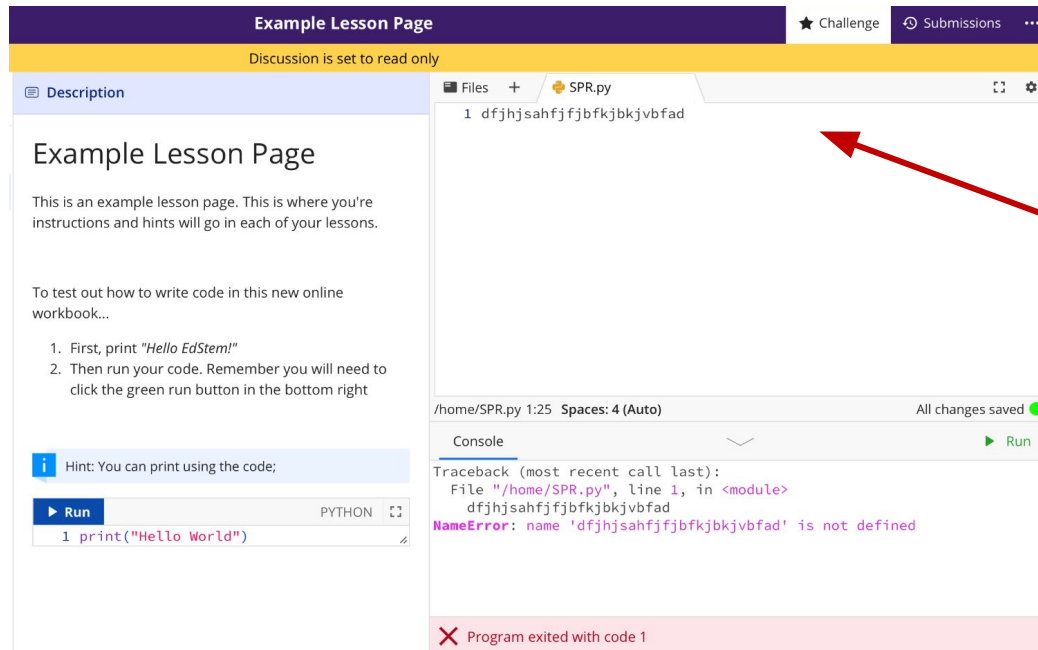


◦ 1: Who are you?

**Go to Part 0 and have a look!**



# Let's make a mistake!



1. Type by **button mashing** the keyboard here e.g.  
**ks@674dbkjSDfk1**
2. **Click** in the Run panel here to run your code!

**Did you get a big ugly error message?**



# Mistakes are great!

*SyntaxError:  
Invalid Syntax*

**Good work you made an error!**

*ImportError:  
No module  
named humour*

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



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

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

*KeyError:  
'Hairy Potter'*



# We can learn from our mistakes!

Traceback (most recent call last):

File "C:/Users/Madeleine/Desktop/tmp.py", line 9, in<module>

print("I have " + 5 + " apples")

TypeError: can only concatenate str (not "int") to str

1. What went wrong

2. Which bit of code didn't work

3. Where that code is

**We read error messages from bottom to top**



# Write some code!

Type this into the code window  
Then press Run!

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

Did it print:

hello world

???





# Python the calculator!

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

1. `print(1 + 5)`

2. `print(2 - 7)`

3. `print(2 * 8)`

4. `print(12 / 3)`



# Python the calculator!

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

1. `print(1 + 5)`

6

2. `print(2 - 7)`

3. `print(2 * 8)`

4. `print(12 / 3)`



# Python the calculator!

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

1. `print(1 + 5)`

6

2. `print(2 - 7)`

-5

3. `print(2 * 8)`

4. `print(12 / 3)`



# Python the calculator!

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

1. `print(1 + 5)`

6

2. `print(2 - 7)`

-5

3. `print(2 * 8)`

16

4. `print(12 / 3)`



# Python the calculator!

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

1. `print(1 + 5)`

6

2. `print(2 - 7)`

-5

3. `print(2 * 8)`

16

4. `print(12 / 3)`

4



# A calculator for words!?

What do you think these bits of code do?

**Try them and see!**

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

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



# Calculator for... words!?

What do you think these bits of code do?

**Try them and see!**

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

catdog

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



# Calculator for... words!?

What do you think these bits of code do?

**Try them and see!**

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

catdog

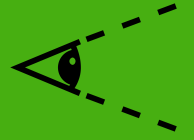
```
print("tortoise" * 3)
```

tortoisetortoisetortoise



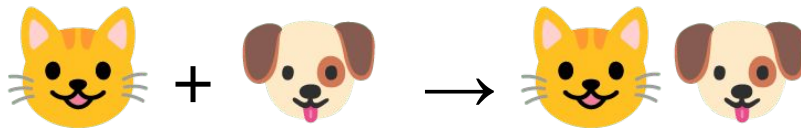


# Strings!



**Strings** are things with "quotes"

1. Strings can be added!



2. Strings can be multiplied!



# Strings and Ints!

**I**ntegers are numbers in python.

We can do maths with **i**ntegers but not **s**trings.

**Predict** what will happen if we write and run the following code.

```
1. print(5 + "5")
```



# Strings and Ints!

**I**ntegers are numbers in python.

We can do maths with **i**ntegers but not **s**trings.

**Predict** what will happen if we write and run the following code.

1. `print(5 + "5")`

`TypeError: can only concatenate str ("not int") to str`



# Variables and Input



# No Storing is Boring!

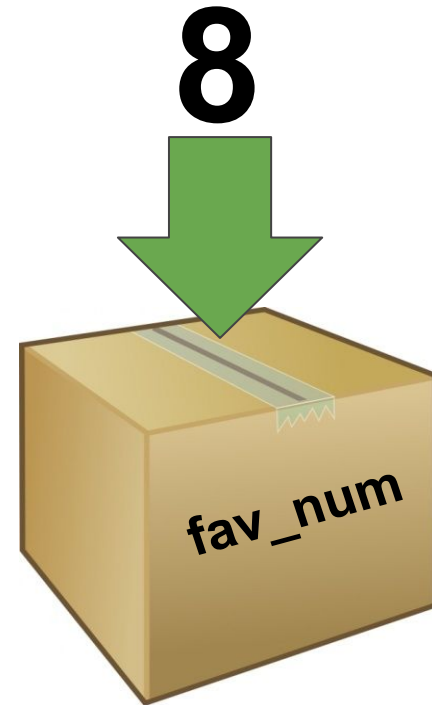
**It's useful to be able to remember things for later!**

Computers remember things in "**variables**"

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

**Let's make our favourite number 8 today!**

In our code we would write  
**`fav_num = 8`**



# Variables

Instead of writing the number 8, we can now use **fav\_num** in our code.



Wherever the computer sees **fav\_num**, it will use the **number 8**

fav\_num - 6

=> **2**

fav\_num \* 2

=> **16**

fav\_num + 21

=> **29**

fav\_num / 2

=> **4**



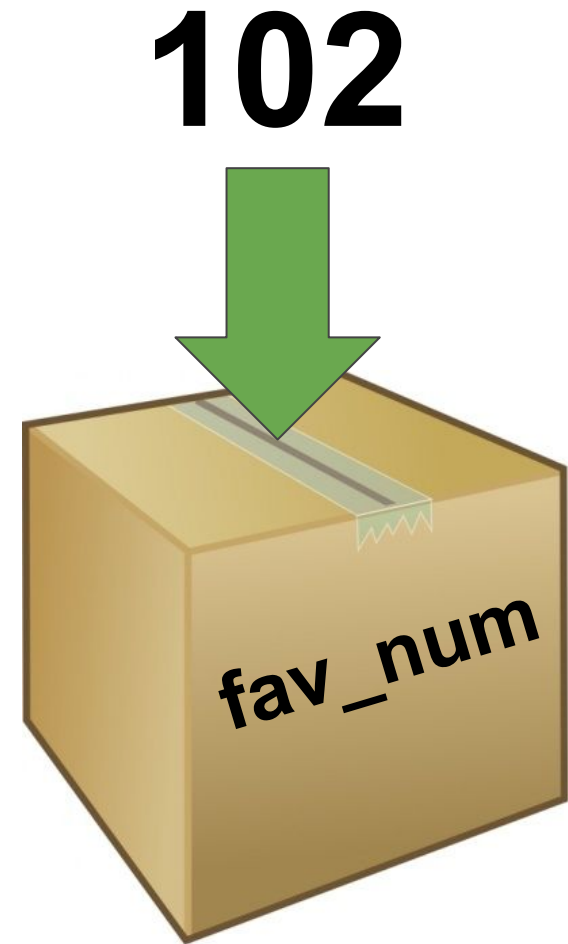
# Variables

**Variables are useful  
for storing things  
that change**

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

What if we changed  
fav\_num to **102**.

**fav\_num = 102**



# Variables

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



`fav_num - 6`  
**=> 96**

`fav_num + 21`  
**=> 123**

`fav_num * 2?`  
**=> 204**

`fav_num / 2?`  
**=> 51**





# Variables

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



`fav_num - 6`  
**=> 96**

`fav_num + 21`  
**=> 123**

`fav_num * 2?`  
**=> 204**

But writing 8 is  
much shorter than  
writing fav\_num???



# No variables VS using variables



4  
Changes

8 - 6

8 \* 2

8 + 21

8 / 2



102 - 6

102 \* 2

102 + 21

102 / 2



1  
Change

fav\_num = 8

fav\_num - 6

fav\_num \* 2

fav\_num + 21

fav\_num / 2



fav\_num = 102

fav\_num - 6

fav\_num \* 2

fav\_num + 21

fav\_num / 2



# Reusing variables

We can replace values in variables:

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

What will this output?



# Reusing variables

We can replace values in variables:

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

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



# Variables

Your turn!

Can you guess what each `print` will do?

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

>>> print(x + x)

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

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



# Variables

Your turn!

Can you guess what each `print` will do?

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

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

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



# Variables

Your turn!

Can you guess what each `print` will do?

```
>>> x = 3
>>> print(x)
3
>>> print(x + x)
6
>>> y = x
>>> print(y)

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



# Variables

Your turn!

Can you guess what each `print` will do?

```
>>> x = 3
>>> print(x)
3
>>> print(x + x)
6
>>> y = x
>>> print(y)
3
>>> y = y + 1
>>> print(y)
```





# Variables

Your turn!

Can you guess what each `print` will do?

```
>>> x = 3
>>> print(x)
3
>>> print(x + x)
6
>>> y = x
>>> print(y)
3
>>> y = y + 1
>>> print(y)
4
```



# Different data!

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

## Strings

Things in quotes used for storing text

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

## Ints

Whole numbers we can do maths with

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

## Floats

Decimal numbers for maths

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

## Booleans

For **True** and **False**

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



# 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!")
```

## Try it!

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



# Asking a question!

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

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

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

What do you think happens?



# Asking a question!

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

What do you think happens?

What is your name? Maddie

Hello Maddie



# Asking a question!

Store the answer  
in the variable  
my\_name

Writing input tells  
the computer to  
wait for a response

This is the question  
you want printed to  
the screen

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

What do you think happens?

What is your name? Maddie

Hello Maddie

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



# Asking a question!

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

How would we print their answer?

**Give it a try on your own computer first!**



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



# Asking a question!

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

How would we print their answer?

**Give it a try on your own computer first!**

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

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





# Asking a question!

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

How would we print their answer?

**Give it a try on your own computer first!**

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

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



# Asking a question with a number answer!

What if our question needed a number as an answer?

**Input** always gives us a string of text

We can turn a **string** into a number by using **int()**

We have to do this if we want to use it as a number

```
age = int(input("How old are you? "))  
very_old = age + 100
```



# Project time!

You now know all about printing and variables!

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



**If it's raining** 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`

`3 + 2 == 5`

`5 != 5`

`"Dog" == "dog"`

`"D" in "Dog"`

`"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

<code>5 &lt; 10</code>	<b>True</b>	<code>"Dog" == "dog"</code>
<code>3 + 2 == 5</code>		<code>"D" in "Dog"</code>
<code>5 != 5</code>		<code>"Q" not in "Cat"</code>



# 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**

`3 + 2 == 5`      **True**

`5 != 5`

`"Dog" == "dog"`

`"D" in "Dog"`

`"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

<code>5 &lt; 10</code>	<code>True</code>	<code>"Dog" == "dog"</code>
<code>3 + 2 == 5</code>	<code>True</code>	<code>"D" in "Dog"</code>
<code>5 != 5</code>	<code>False</code>	<code>"Q" not in "Cat"</code>



# 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

<code>5 &lt; 10</code>	<code>True</code>	<code>"Dog" == "dog"</code>	<code>False</code>
<code>3 + 2 == 5</code>	<code>True</code>	<code>"D" in "Dog"</code>	
<code>5 != 5</code>	<code>False</code>	<code>"Q" not in "Cat"</code>	



# 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

<code>5 &lt; 10</code>	<code>True</code>	<code>"Dog" == "dog"</code>	<code>False</code>
<code>3 + 2 == 5</code>	<code>True</code>	<code>"D" in "Dog"</code>	<code>True</code>
<code>5 != 5</code>	<code>False</code>	<code>"Q" not in "Cat"</code>	



# 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

<code>5 &lt; 10</code>	<code>True</code>	<code>"Dog" == "dog"</code>	<code>False</code>
<code>3 + 2 == 5</code>	<code>True</code>	<code>"D" in "Dog"</code>	<code>True</code>
<code>5 != 5</code>	<code>False</code>	<code>"Q" not in "Cat"</code>	<code>True</code>



# Booleans (True and False)

Python has some special comparisons for checking if something is **in** something else.

```
>>> "A" in "AEIOU"  
>>> "Z" in "AEIOU"  
>>> "a" in "AEIOU"
```

```
>>> animals = ["cat", "dog", "goat"]  
>>> "banana" in animals  
>>> "cat" in animals
```



# Conditions

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

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

# Conditions

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

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

That's the  
condition!

# Conditions

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

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

That's the  
condition!

Is it **True** that fave\_num is less than 10?

- Well, fave\_num is 5
- And it's **True** that 5 is less than 10
- So it is **True**!





# Conditions

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

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

Put in the  
answer to  
the question

Is it **True** that fave\_num is less than 10?

- Well, fave\_num is 5
- And it's **True** that 5 is less than 10
- So it is **True**!



# Conditions

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

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

What do you think happens?

```
>>>
```



# Conditions

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

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

What do you think happens?

```
>>> that's a small number
```



# Conditions

How about a different number???



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

# Conditions

Find out if it's **True**!

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

Put in the  
answer to  
the question

Is it **True** that fave\_num is less than 10?

- Well, fave\_num is 9000
- And it's not **True** that 9000 is less than 10
- So it is **False**!



# Conditions

How about a different number???

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



What do you think happens?

```
>>>
```

# Conditions

How about a different number???

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



What do you think happens?

```
>>>
```



**Nothing!**



# If statements

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

This line ...

... controls this line





# If statements

## Actually .....

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

This line ...

... controls anything below it  
that is indented like this!



# 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??

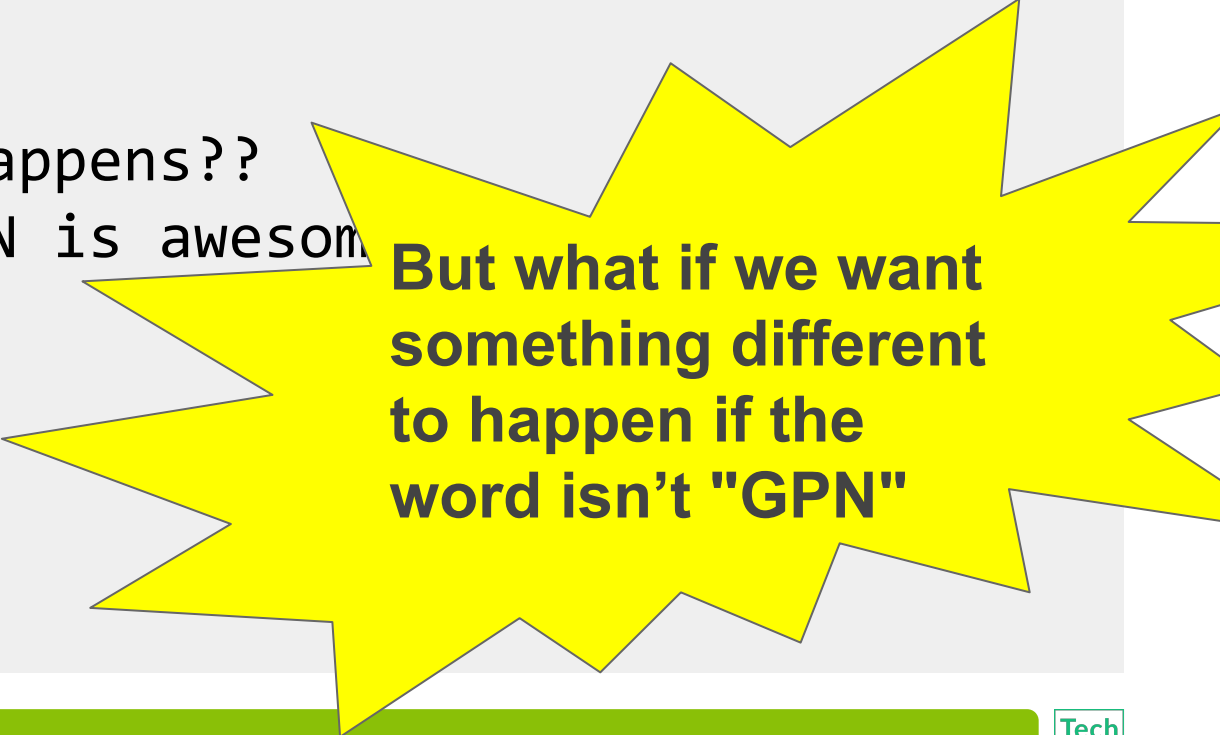
```
>>> GPN is awesome!
```

# Else statements

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

What happens??

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



**But what if we want  
something different  
to happen if the  
word isn't "GPN"**

# Else statements

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

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

What happens??



# Else statements

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

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

What happens??

```
>>> The word isn't GPN :(
```





# Elif statements

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

**elif**

Means we can  
give specific  
instructions for  
other words

What happens??

# Elif statements

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

**elif**

Means we can  
give specific  
instructions for  
other words

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 going outside?

You might do it like this:



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

But that starts to get messy quickly.

# AND

Instead you can do it like this!

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

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



# Project Time!



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

**See **if** you can do Part 3**

The tutors will be around to help!



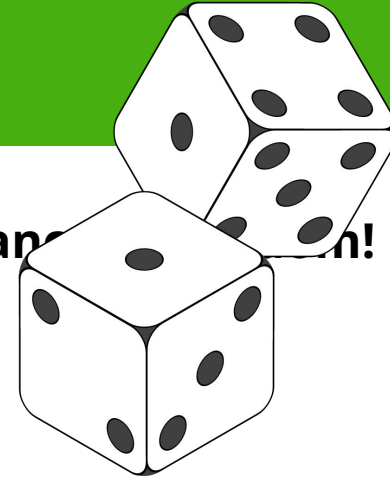
Random!





# 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!

## Try this!

### 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

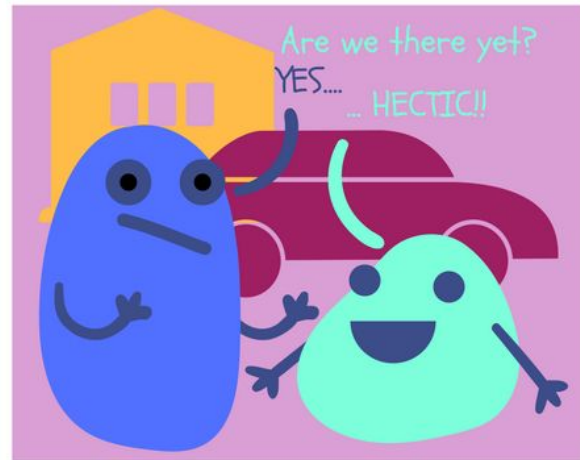
The tutors will be around to help!



# While Loops



# Loops



We know how to do things on repeat!

Sometimes we want to do some code on repeat!

# Introducing ... while loops!

**We can do something while a condition is met**

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



# Introducing ... while loops!

## What do you think this does?

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

0

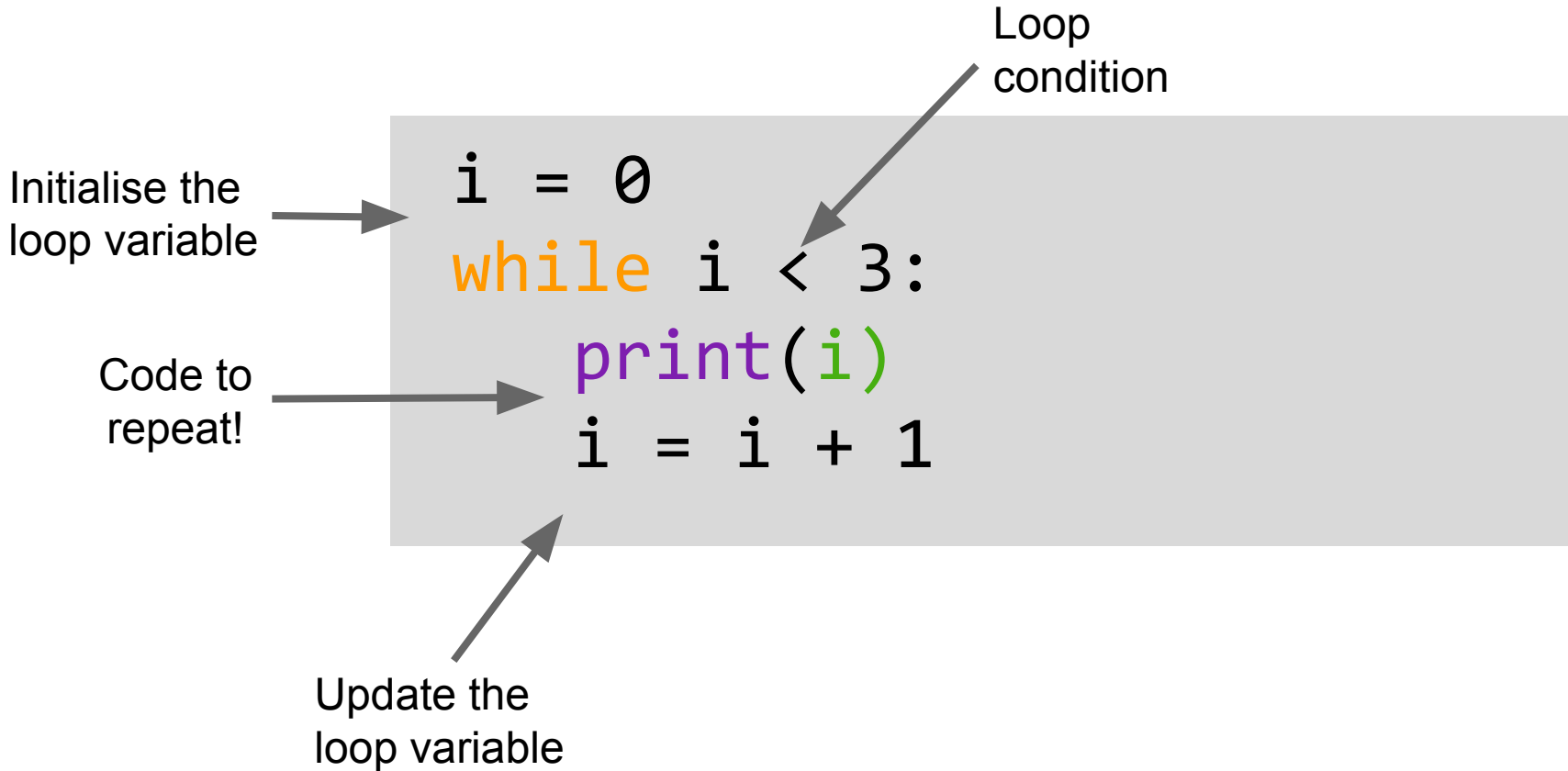
1

2





# Introducing ... while loops!



# Introducing ... while loops!

Stepping through a while loop...



# Introducing ... while loops!

## One step at a time!



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

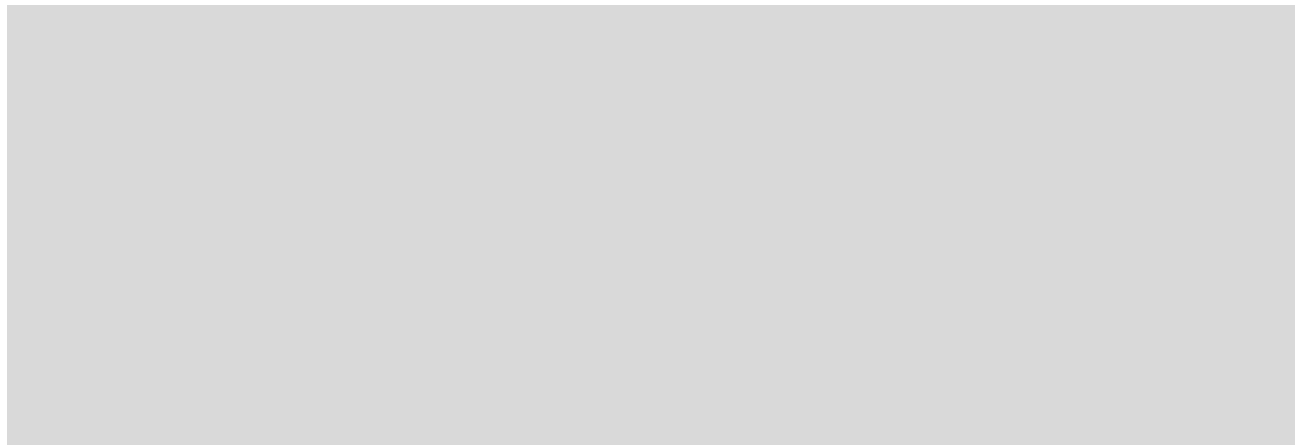


MY VARIABLES

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)
    i = i + 1
```

MY VARIABLES

i = 0



# Introducing ... while loops!

## One step at a time!

Print!

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

0


MY VARIABLES


i = 0



# Introducing ... while loops!

## One step at a time!

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



MY VARIABLES

~~i = 0~~  
i = 1

UPDATE  
TIME!

0



# Introducing ... while loops!

## One step at a time!

Take it  
from the  
top!

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

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)
    i = i + 1
```

MY VARIABLES

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

0





# Introducing ... while loops!

## One step at a time!

Print!

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

0

1


MY VARIABLES


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



# Introducing ... while loops!

## One step at a time!

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



### MY VARIABLES

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

UPDATE  
TIME!

0

1



# Introducing ... while loops!

## One step at a time!

Take it  
from the  
top!

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

0

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)
    i = i + 1
```

MY VARIABLES

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

0

1



# Introducing ... while loops!

## One step at a time!

Print!

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

0

1

2

### MY VARIABLES

~~i = 0~~


~~i = 1~~


i = 2



# Introducing ... while loops!

## One step at a time!

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



MY VARIABLES

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

UPDATE  
TIME!

0

1

2



# Introducing ... while loops!

## One step at a time!

Take it  
from the  
top!

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

0

1

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)
    i = i + 1
```

We are  
done  
with this  
loop!

```
0
1
2
```

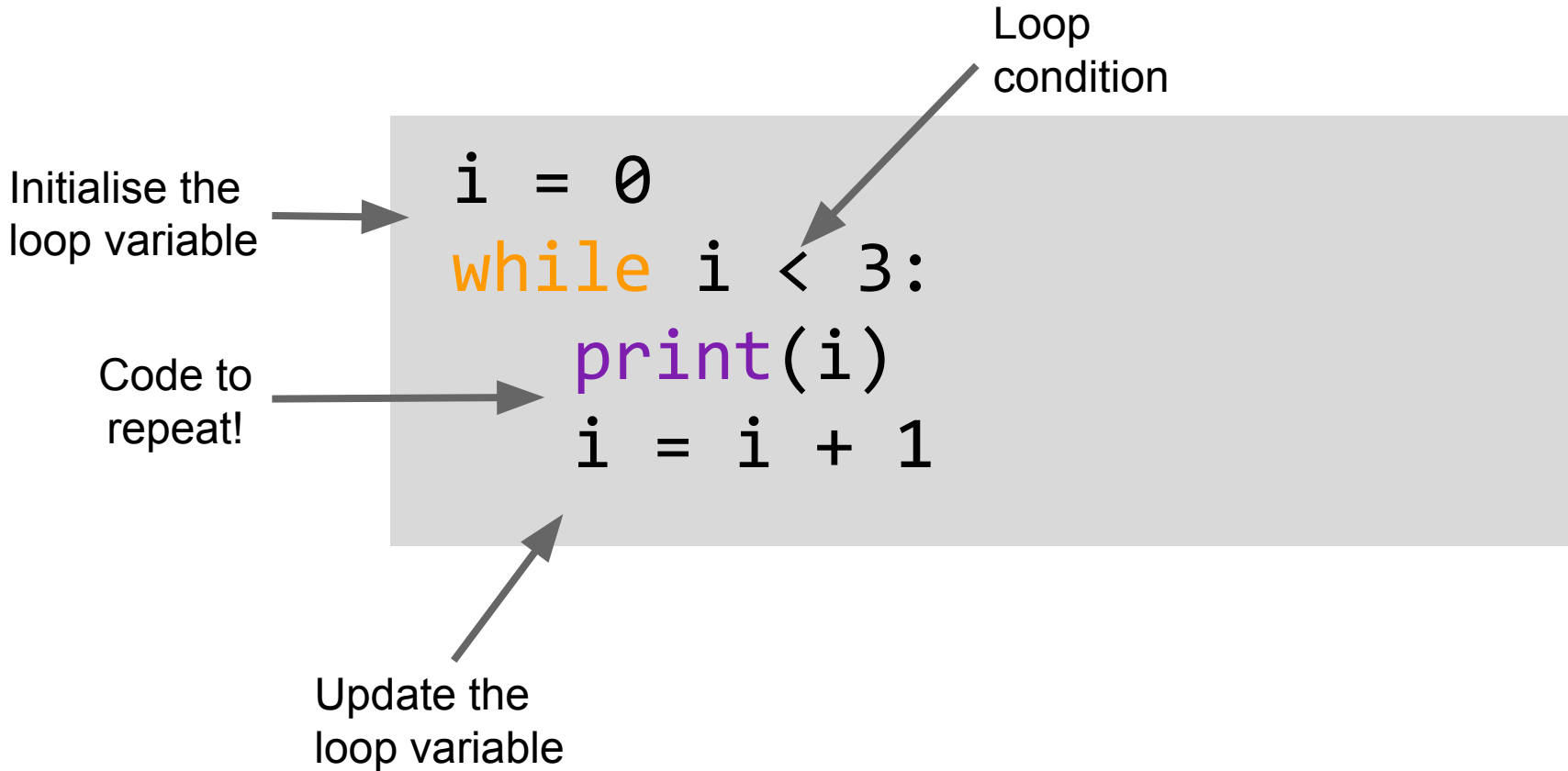
MY VARIABLES

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





# Introducing ... while loops!



# What happens when.....

What happens if we forget to update the loop variable?

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



# What happens when.....

What happens if we forget to update the loop variable?

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

i is 0

i is 0

i is 0

i is 0

i is 0

i is 0

i is 0

i is 0

i is 0

i is 0

i is 0

i is 0

i is 0

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?")
```



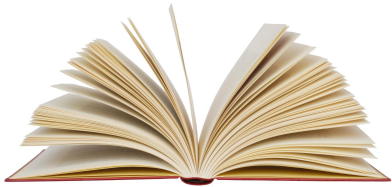
# For Loops



# For Loops

For loops allow you to do something **for a number of times or for each item in a group**

There are many real world examples, like:

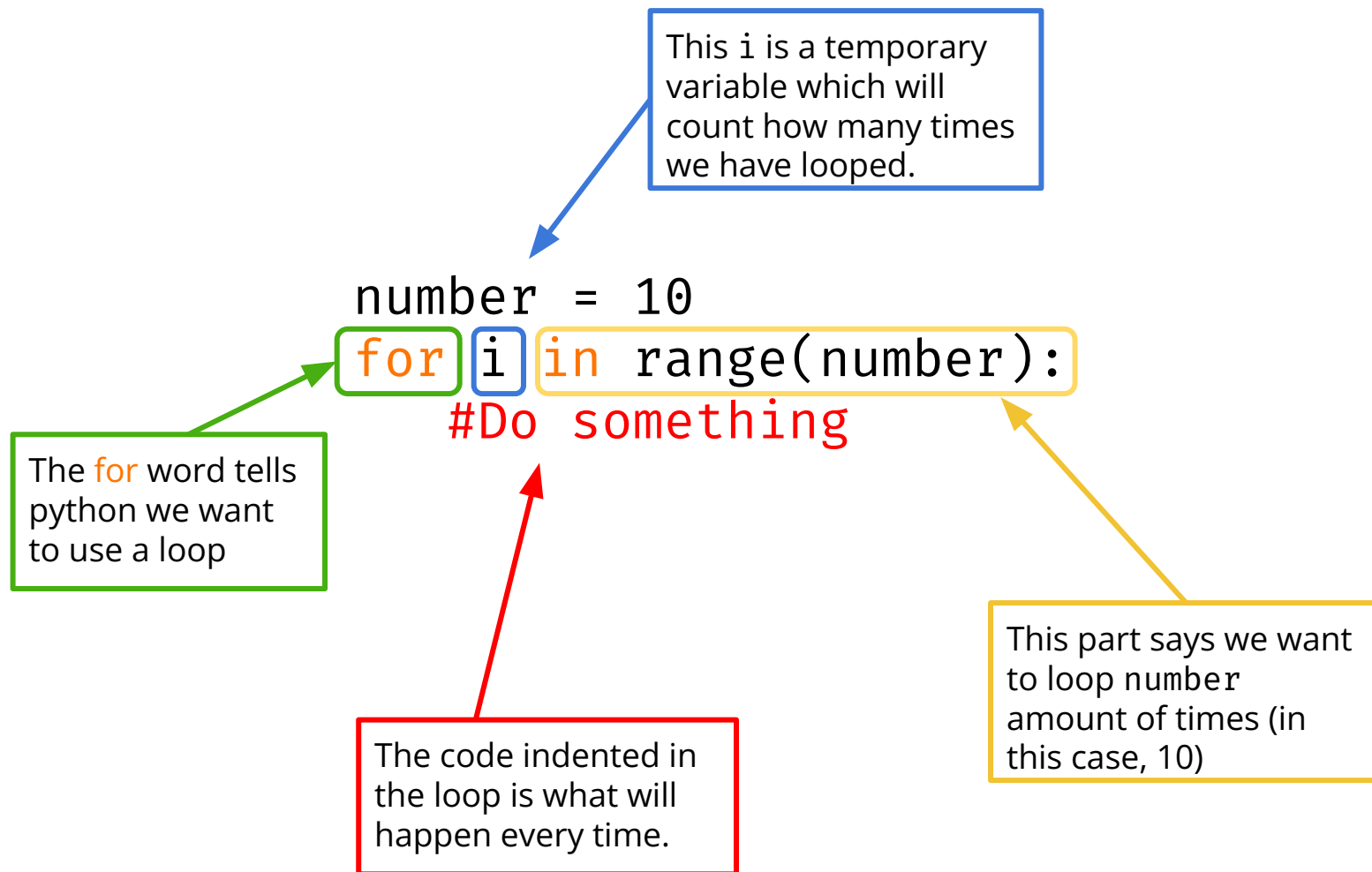


**For each page in this book:  
Read**



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

# For Loops



# Looping how many times?

**We can loop through a list:**

```
friends = 4  
for i in range(friends):  
    print("Hello friend!")
```

What's going to happen?





# Looping how many times?

## We can loop through a list:

```
friends = 4  
for i in range(friends):  
    print("Hello friend!")
```

What's going to happen?

We do what's in the for loop as many times as what is in the "range"

```
>>> Hello friend!  
>>> Hello friend!  
>>> Hello friend!  
>>> Hello friend!
```



# Asking a question with a number answer!

It's common to ask the user to enter a number

**Input** always gives us a string of text

We need to turn the **string** into a number before we can use it as a range in a for loop

We do this by using **int()**

```
no_of_turns = int(input("How many times: " ))  
for i in range(no_of_turns)  
    Do something
```



# Project Time!



**Now you know how to use a for WHILE  
and FOR loop!**

**Try to do Part 5**

**And then try the Extensions 6 - 9!**

The tutors will be around to help!



Tell us what you think!

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