

# Welcome to the labs!

## Cryptography!



# Thank you to our Sponsors!

Platinum Sponsor:



# Who are the tutors?

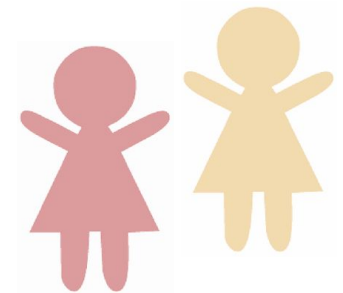
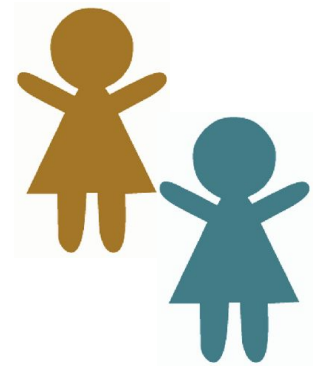


Who are you?



# Two Truths and a Lie

1. Get in a group of 3-5 people
2. Tell them three things about yourself:
  - a. Two of these things should be true
  - b. One of these things should be a lie!
3. The other group members have to guess which is the lie



# Log on

## Log on and jump on the GPN website

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

You can see:

- These **slides** (to take a look back on or go on ahead).
- A link to the EdStem course
- Helpful bits of text you can **copy and paste!**



Tell us you're here!

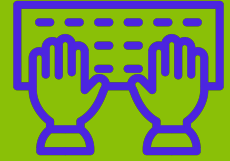
Click on the  
**Start of Day Survey**  
and fill it in now!

# Introduction to Edstem





# Signing up to Edstem



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

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

1. Follow this link: <https://edstem.org/au/join/qKyppB>
2. Type in your name and your personal email address
3. Click Create Account
4. Go to your email to verify your account
5. Create a password
6. It should then take you to the courses home page.
7. Click on the one we will be using for this project: —————>

Cryptography G  
Cryptography G

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


# Getting to the lessons

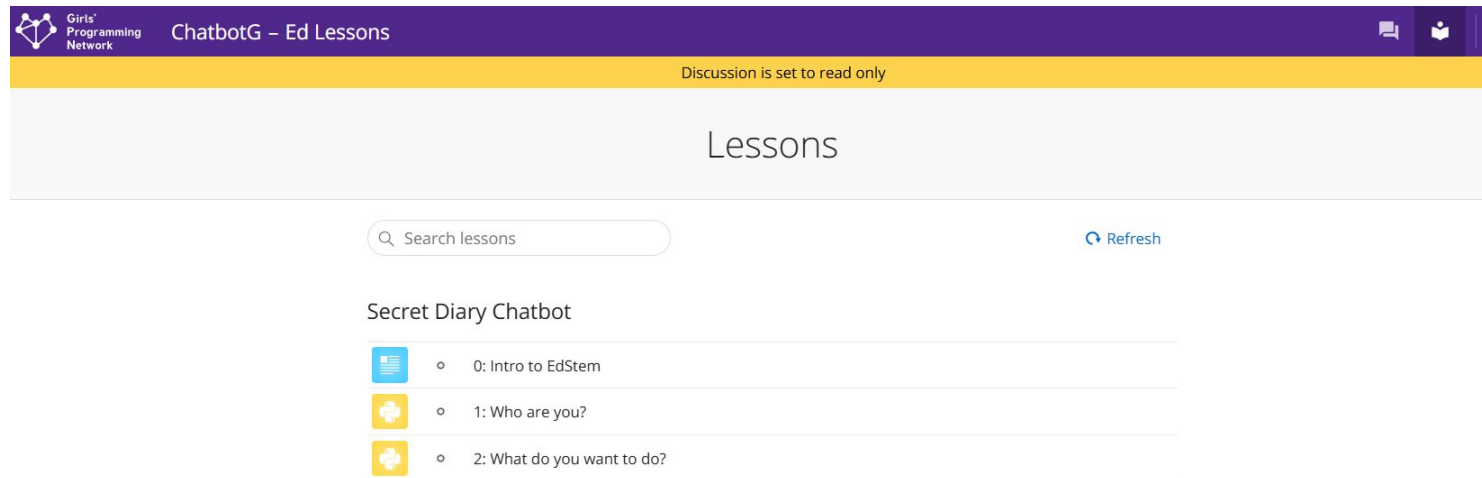
1. Once you are in the course, you'll be taken to a discussion page.
2. Click the button for the lessons page (top right - looks like a book)



# The set up of the workbook

## The main page:

- Heading at the top that tells you the project you are in
- List of “Chapters” - They have an icon 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

≡ 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

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.



# 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

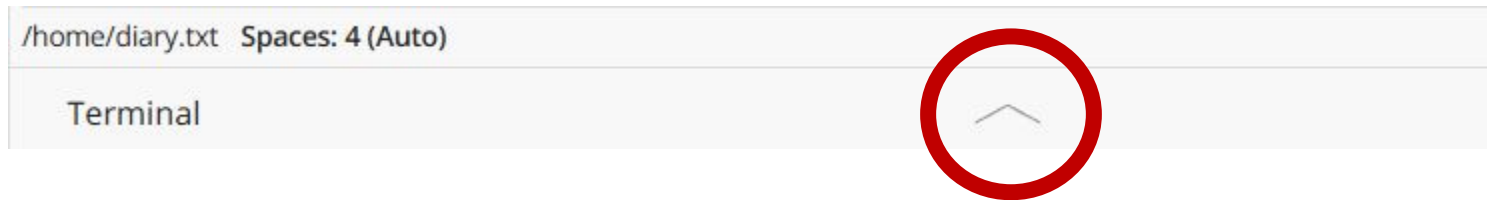
There are also  
Hints and  
Code Blocks to  
help you

The screenshot displays a web-based programming interface. On the left, a 'Description' panel titled 'Example Lesson Page' provides instructions: 'This is an example lesson page. This is where you're instructions and hints will go in each of your lessons.' It also includes a section 'To test out how to write code in this new online workbook...' with two steps: '1. First, print "Hello EdStem!"' and '2. Then run your code in the terminal. Remember you will need to enter the code `python chatbot.py`'. Below this is a 'Hint' box stating 'Hint: You can print using the code;' and a 'Run' button. At the bottom of the left panel is a code block with the text `1 print("Hello World")`. On the right, a code editor window titled 'chatbot.py' contains the code: `1 # Put your testing code here!!!` and `2 print("Hello EdStem!")`. The bottom of the interface shows a terminal window with the command `/home/chatbot.py` and the status 'Spaces: 4 (Auto)' and 'All changes saved'.



# Running your code...

1. Open the Terminal window below your code



2. Click button that says "Click here to activate the terminal".

Click here to activate the terminal

3. Your code should run automatically.
4. Click the button again to rerun your code.
5. You can resize the Terminal window.

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



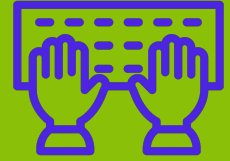
# Some shortcuts...

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**

Cryptography



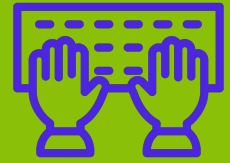
◦ 0. Intro to EdStem

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





# Project time!



You now know all about EdStem!

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

Remember the tutors will be around to help!



# Intro to Caesar Ciphers

Cryptography G  
Cryptography G

Let's get encrypting!



# What is a cipher?

A cipher is a way to write a message so that no one else can read it!

Unless they know the secret key!



# Examples of ciphers

If you've ever made up your own secret language or made notes to your friends so that other people can't read them, you've made a cipher!

For example:

**gnidoc evol i**

Can you figure out what this says?



# Examples of ciphers

If you've ever made up your own secret language or made notes to your friends so that other people can't read them, you've made a cipher!

For example:

**gnidoc evol i**

Can you figure out what this says?

It says **I love coding** backwards!



# Caesar Cipher

So what's a Caesar Cipher?

It's a cypher that Julius Caesar used in ancient Rome to send secret messages to his armies!

Let's learn how it works!



# Make a Cipher Wheel

- Cut out green circle
- Cut out purple circle
- Put small circle on top of big circle matching centres
- Secure together with centre split pin
- Spin inside circle of letters around



Caesar Cipher Wheel template in Workshop Material folder

# Shifting letters

A Caesar Cipher works by shifting letters in the alphabet so that they line up with new letters.

For example if we were to shift everything by 3 it would look like this:

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c

Line up the 'a' on both wheels and then turn the inside wheel 3 letters **anti-clockwise** so that you have your letters lining up like this!





# Encrypting

Now, let's encrypt **I love coding** using the wheel

For our Caesar Cipher we take each letter and replace it with the 'shifted' letter

So, let's start with the letter 'i'  
What new letter should we use to replace it?



>>> Find letter *i* on the **outside** wheel and replace it with its matching letter on the **inside** wheel = the letter 'h'



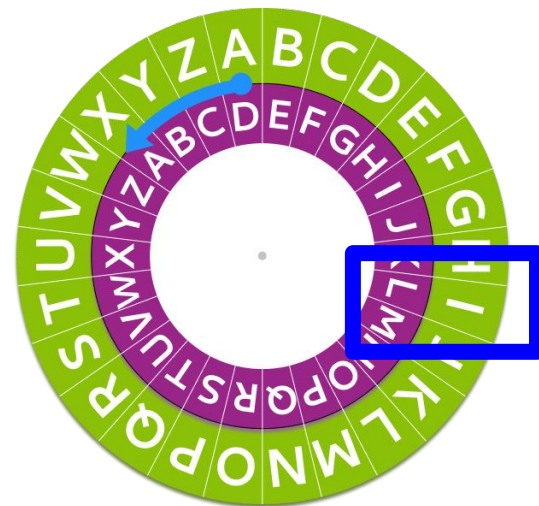
# Encrypting

Now, let's encrypt **I love coding** using the wheel

For our Caesar Cipher we take each letter and replace it with the 'shifted' letter

So, let's start with the letter 'i'

What new letter should we use to replace it?



>>> Find letter *i* on the **outside** wheel and replace it with its matching letter on the **inside** wheel = the letter 'j'



# Writing the whole message!

Let's do the rest of the message together

**I love coding**

I	Is replaced with	
o	Is replaced with	
v	Is replaced with	
e	Is replaced with	
c	Is replaced with	
o	Is replaced with	
d	Is replaced with	
i	Is replaced with	
n	Is replaced with	
g	Is replaced with	



# Writing the whole message!

Let's do the rest of the message together

**I love coding**

I	Is replaced with	
o	Is replaced with	
v	Is replaced with	
e	Is replaced with	
c	Is replaced with	
o	Is replaced with	
d	Is replaced with	
i	Is replaced with	
n	Is replaced with	
g	Is replaced with	

o



# Writing the whole message!

Let's do the rest of the message together

**I love coding**

I	Is replaced with	o
o	Is replaced with	r
v	Is replaced with	
e	Is replaced with	
c	Is replaced with	
o	Is replaced with	
d	Is replaced with	
i	Is replaced with	
n	Is replaced with	
g	Is replaced with	



# Writing the whole message!

Let's do the rest of the message together

**I love coding**

I	Is replaced with	o
o	Is replaced with	r
v	Is replaced with	y
e	Is replaced with	
c	Is replaced with	
o	Is replaced with	
d	Is replaced with	
i	Is replaced with	
n	Is replaced with	
g	Is replaced with	



# Writing the whole message!

Let's do the rest of the message together

**I love coding**

I	Is replaced with	o
o	Is replaced with	r
v	Is replaced with	y
e	Is replaced with	h
c	Is replaced with	
o	Is replaced with	
d	Is replaced with	
i	Is replaced with	
n	Is replaced with	
g	Is replaced with	



# Writing the whole message!

Let's do the rest of the message together

**I love coding**

I	Is replaced with	o
o	Is replaced with	r
v	Is replaced with	y
e	Is replaced with	h
c	Is replaced with	f
o	Is replaced with	
d	Is replaced with	
i	Is replaced with	
n	Is replaced with	
g	Is replaced with	





# Writing the whole message!

Let's do the rest of the message together

**I love coding**

I	Is replaced with	o
o	Is replaced with	r
v	Is replaced with	y
e	Is replaced with	h
c	Is replaced with	f
o	Is replaced with	r
d	Is replaced with	
i	Is replaced with	
n	Is replaced with	
g	Is replaced with	



# Writing the whole message!

Let's do the rest of the message together

**I love coding**

I	Is replaced with	o
o	Is replaced with	r
v	Is replaced with	y
e	Is replaced with	h
c	Is replaced with	f
o	Is replaced with	r
d	Is replaced with	g
i	Is replaced with	
n	Is replaced with	
g	Is replaced with	



# Writing the whole message!

Let's do the rest of the message together

**I love coding**

I	Is replaced with	o
o	Is replaced with	r
v	Is replaced with	y
e	Is replaced with	h
c	Is replaced with	f
o	Is replaced with	r
d	Is replaced with	g
i	Is replaced with	I
n	Is replaced with	
g	Is replaced with	



# Writing the whole message!

Let's do the rest of the message together

**I love coding**

I	Is replaced with	o
o	Is replaced with	r
v	Is replaced with	y
e	Is replaced with	h
c	Is replaced with	f
o	Is replaced with	r
d	Is replaced with	g
i	Is replaced with	l
n	Is replaced with	q
g	Is replaced with	



# Writing the whole message!

Let's do the rest of the message together

**I love coding**

I	Is replaced with	o
o	Is replaced with	r
v	Is replaced with	y
e	Is replaced with	h
c	Is replaced with	f
o	Is replaced with	r
d	Is replaced with	g
i	Is replaced with	l
n	Is replaced with	q
g	Is replaced with	j



# Secret Message

**So our secret encrypted message is**  
**L oryh frglqj**

That's a lot harder to figure out than it just being  
backwards!

Encrypt your own name!

Using a key of minus 1 (so A=Z) (Jessica = ldr rhbz)

Write your name on the blank tag in name badge!



# Decrypting

Writing secret messages isn't any fun if you can't figure out what they say!

**Luckily you can also use your cipher wheel to *decrypt* a secret message.**

How do you think we can do that?

What information do we need to know in order to decrypt a secret message?



# It's the key!

To decrypt a secret message **we need to know** the amount that we shifted the wheel when we encrypted it. That number is called **the key**!

**Once we know the key we can just turn our wheel and read the wheel from the inside out!**

*Find the letter on the **inside** wheel and replace it with it's matching letter on the **outside** wheel*





# Let's check it works!

l	Is replaced with
o	Is replaced with
r	Is replaced with
y	Is replaced with
h	Is replaced with
f	Is replaced with
r	Is replaced with
g	Is replaced with
l	Is replaced with
q	Is replaced with
j	Is replaced with



# Let's check it works!

l	Is replaced with
o	Is replaced with
r	Is replaced with
y	Is replaced with
h	Is replaced with
f	Is replaced with
r	Is replaced with
g	Is replaced with
l	Is replaced with
q	Is replaced with
j	Is replaced with

i



# Let's check it works!

l	Is replaced with
o	Is replaced with
r	Is replaced with
y	Is replaced with
h	Is replaced with
f	Is replaced with
r	Is replaced with
g	Is replaced with
l	Is replaced with
q	Is replaced with
j	Is replaced with

i  
l



# Let's check it works!

l	Is replaced with
o	Is replaced with
r	Is replaced with
y	Is replaced with
h	Is replaced with
f	Is replaced with
r	Is replaced with
g	Is replaced with
l	Is replaced with
q	Is replaced with
j	Is replaced with

i  
l  
o



# Let's check it works!

l	Is replaced with
o	Is replaced with
r	Is replaced with
y	Is replaced with
h	Is replaced with
f	Is replaced with
r	Is replaced with
g	Is replaced with
l	Is replaced with
q	Is replaced with
j	Is replaced with

i
l
o
v



# Let's check it works!

l	Is replaced with
o	Is replaced with
r	Is replaced with
y	Is replaced with
h	Is replaced with
f	Is replaced with
r	Is replaced with
g	Is replaced with
l	Is replaced with
q	Is replaced with
j	Is replaced with

i
l
o
v
e



# Let's check it works!

l	Is replaced with
o	Is replaced with
r	Is replaced with
y	Is replaced with
h	Is replaced with
f	Is replaced with
r	Is replaced with
g	Is replaced with
l	Is replaced with
q	Is replaced with
j	Is replaced with

i
l
o
v
e
c



# Let's check it works!

l	Is replaced with
o	Is replaced with
r	Is replaced with
y	Is replaced with
h	Is replaced with
f	Is replaced with
r	Is replaced with
g	Is replaced with
l	Is replaced with
q	Is replaced with
j	Is replaced with

i
l
o
v
e
c
o





# Let's check it works!

l	Is replaced with
o	Is replaced with
r	Is replaced with
y	Is replaced with
h	Is replaced with
f	Is replaced with
r	Is replaced with
g	Is replaced with
l	Is replaced with
q	Is replaced with
j	Is replaced with

i
l
o
v
e
c
o
d



# Let's check it works!

l	Is replaced with
o	Is replaced with
r	Is replaced with
y	Is replaced with
h	Is replaced with
f	Is replaced with
r	Is replaced with
g	Is replaced with
l	Is replaced with
q	Is replaced with
j	Is replaced with

i
l
o
v
e
c
o
d
i



# Let's check it works!

l	Is replaced with
o	Is replaced with
r	Is replaced with
y	Is replaced with
h	Is replaced with
f	Is replaced with
r	Is replaced with
g	Is replaced with
l	Is replaced with
q	Is replaced with
j	Is replaced with

i
l
o
v
e
c
o
d
i
n



# Let's check it works!

l	Is replaced with
o	Is replaced with
r	Is replaced with
y	Is replaced with
h	Is replaced with
f	Is replaced with
r	Is replaced with
g	Is replaced with
l	Is replaced with
q	Is replaced with
j	Is replaced with

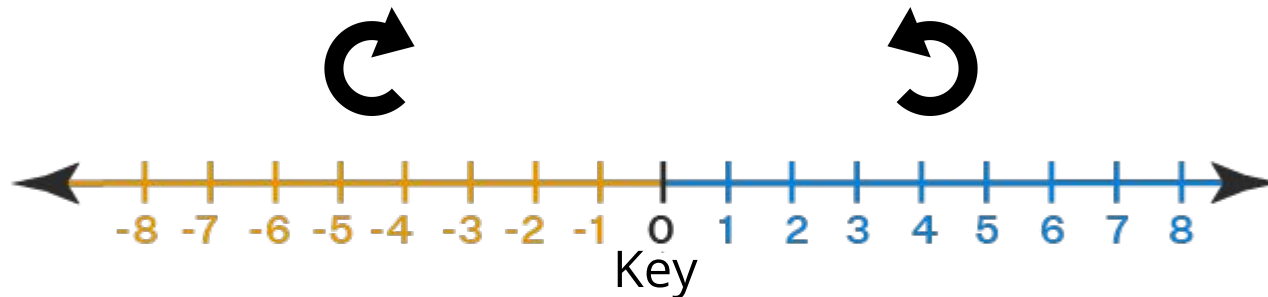
i
l
o
v
e
c
o
d
i
n
g



# Another way to decrypt



- Another way to decrypt a message is to change the key value to become the negative of the encryption key value
- We will use this method in our code
- This is because to decrypt a message we need to shift the alphabet the opposite way.
- A negative key value means you turn your inner purple wheel to the right (clockwise)



# Your Turn!

**Try doing Lessons 1-3  
using your Caesar Cipher wheels!**

Your tutors are here to help you if you get  
stuck



# Strings, Ints & Modulo



# Strings!

Strings are a sequence of characters in python.

Strings are created by enclosing characters inside  
**"quotes"**

>>> `alphabet = 'abcdefghijklmnopqrstuvwxyz'` creates a string variable that contains the letters of the alphabet

We can add strings together

>>> `"abc" + "def" = "abcdef"`





# Strings

We can get individual letters from a **string** using indexes.

```
>>> yum = "chocolate"
```

```
>>> yum[0]
```

```
>>> yum[5]
```

```
>>> yum[-1]
```

```
>>> yum[500]
```



# Strings

We can get individual letters from a **string** using indexes.

```
>>> yum = "chocolate"
```

```
>>> yum[0]
```

```
'c'
```

Computers start counting from 0, not 1!

```
>>> yum[5]
```

```
>>> yum[-1]
```

```
>>> yum[500]
```



# Strings

We can get individual letters from a **string** using indexes.

```
>>> yum = "chocolate"
```

```
>>> yum[0]
```

```
'c'
```

Computers start counting from 0, not 1!

```
>>> yum[5]
```

```
'l'
```

```
>>> yum[-1]
```

```
>>> yum[500]
```



# Strings

We can get individual letters from a **string** using indexes.

```
>>> yum = "chocolate"
```

```
>>> yum[0]
```

```
'c'
```

Computers start counting from 0, not 1!

```
>>> yum[5]
```

```
'l'
```

```
>>> yum[-1]
```

```
'e'
```

```
>>> yum[500]
```



# Strings

We can get individual letters from a **string** using indexes.

```
>>> yum = "chocolate"
```

```
>>> yum[0]
```

```
'c'
```

Computers start counting from 0, not 1!

```
>>> yum[5]
```

```
'l'
```

```
>>> yum[-1]
```

```
'e'
```

```
>>> yum[500]
```

```
IndexError: string index out of range
```



# Searching Strings

If we want to find where a letter is in a **string**, we look it up using **index()**

```
>>> yum = "chocolate"
```

```
>>> yum.index('h')
```

```
>>> yum.index('o')
```

```
>>> yum.index('z')
```



# Searching Strings

If we want to find where a letter is in a **string**, we look it up using **index()**

```
>>> yum = "chocolate"
```

```
>>> yum.index('h')
```

```
1
```

```
>>> yum.index('o')
```

```
>>> yum.index('z')
```



# Searching Strings

If we want to find where a letter is in a **string**, we look it up using **index()**

```
>>> yum = "chocolate"
```

```
>>> yum.index('h')
```

```
1
```

```
>>> yum.index('o')
```

```
2
```

Only the index of the first 'o' is returned!

```
>>> yum.index('z')
```





# Searching Strings

If we want to find where a letter is in a **string**, we look it up using **index()**

```
>>> yum = "chocolate"
```

```
>>> yum.index('h')
```

```
1
```

```
>>> yum.index('o')
```

```
2
```

Only the index of the first 'o' is returned!

```
>>> yum.index('z')
```

```
ValueError: substring not found
```



# Test if character in string

We can test if a character is in a string!

```
>>> yum = "chocolate"  
>>> if 'a' in yum:
```



# Maths on Indexes!

We can use any sort of **int** as an index, including the result of an expression or maths equation!

```
>>> yum = "chocolate"
```

```
>>> len(yum)
```

```
>>> yum[9 - 1]
```



# Maths on Indexes!

We can use any sort of **int** as an index, including the result of an expression or maths equation!

```
>>> yum = "chocolate"
```

```
>>> len(yum)
```

```
9
```

```
>>> yum[9 - 1]
```



# Maths on Indexes!

We can use any sort of **int** as an index, including the result of an expression or maths equation!

```
>>> yum = "chocolate"
```

```
>>> len(yum)
```

```
9
```

```
>>> yum[9 - 1]
```

```
'e'
```



# Modulo %

Modulo % is a maths operation

% gives the **remainder** of a division

You'll need to use it in your code!

- $10 \% 8 = 2$  (10 divided by 8 is 1 with remainder 2)
- $20 \% 7 = 6$  (20 divided by 7 is 2 with remainder 6)
- $5 \% 6 = 5$  (5 divided by 6 is 0 with remainder 5)

# Project time!

You now know all about strings, ints and modulo!

**Let's put what we learnt into our project**  
**Try Lessons 3 - 6**

The tutors will be around to help!



# Intro to Vigenere Ciphers

Cryptography P  
Cryptography





# Caesar Cipher

So now you know what a Caesar Cipher is, let's look at a more complicated cipher!

A Caesar Cipher uses just 1 key to encrypt and decrypt the message, a Vigenere cypher uses a whole word as the key!



# The keyword

Let's see how it uses a whole word by doing an example together!

Let's use the keyword  
**pizza**



# Splitting it into keys

Now we take the keyword and we split it into a bunch of keys!

Each letter of the alphabet equals a different number  
(a=0, b=1, c=2 etc.)

Now we change our keyword into a bunch of different keys  
by replacing each letter with its number in the alphabet

**p**

**i**

**z**

**z**

**a**



# Splitting it into keys

Now we take the keyword and we split it into a bunch of keys!

Each letter of the alphabet equals a different number  
(a=0, b=1, c=2 etc.)

Now we change our keyword into a bunch of different keys  
by replacing each letter with its number in the alphabet

**p**

**i**

**z**

**z**

**a**

15



# Splitting it into keys

Now we take the keyword and we split it into a bunch of keys!

Each letter of the alphabet equals a different number  
(a=0, b=1, c=2 etc.)

Now we change our keyword into a bunch of different keys  
by replacing each letter with its number in the alphabet

**p**

15

**i**

8

**z**

**z**

**a**



# Splitting it into keys

Now we take the keyword and we split it into a bunch of keys!

Each letter of the alphabet equals a different number  
(a=0, b=1, c=2 etc.)

Now we change our keyword into a bunch of different keys  
by replacing each letter with its number in the alphabet

**p**

15

**i**

8

**z**

25

**z**

**a**



# Splitting it into keys

Now we take the keyword and we split it into a bunch of keys!

Each letter of the alphabet equals a different number  
(a=0, b=1, c=2 etc.)

Now we change our keyword into a bunch of different keys  
by replacing each letter with its number in the alphabet

**p**

15

**i**

8

**z**

25

**z**

25

**a**



# Splitting it into keys

Now we take the keyword and we split it into a bunch of keys!

Each letter of the alphabet equals a different number  
(a=0, b=1, c=2 etc.)

Now we change our keyword into a bunch of different keys  
by replacing each letter with its number in the alphabet

**p**

15

**i**

8

**z**

25

**z**

25

**a**

0





# Loop the word

Let's try encrypting a message with our keyword using a Vigenere cipher now!

**I love coding**

Each letter in our message will line up with a letter in our keyword and we will keep looping the keyword like this:

i	l	o	v	e	c	o	d	i	n	g
p	i	z	z	a	p	i	z	z	a	p



# Using the numbers

Now we replace each letter of our keyword with the numbers that we worked out before:

i	l	o	v	e	c	o	d	i	n	g
15	8	25	25	0	15	8	25	25	0	15

Next we just shift each letter in our message like we do with a Caesar Cipher but with the key that it lines up with.

What key does the letter C use?



# Using the numbers

Now we replace each letter of our keyword with the numbers that we worked out before:

i	l	o	v	e	c	o	d	i	n	g
15	8	25	25	0	15	8	25	25	0	15

Next we just shift each letter in our message like we do with a Caesar Cipher but with the key that it lines up with.

What key does the letter C use? 15



# Making the secret message

i	Using key: <b>15</b>	Is replaced with
l	Using key: <b>8</b>	Is replaced with
o	Using key: <b>25</b>	Is replaced with
v	Using key: <b>25</b>	Is replaced with
e	Using key: <b>0</b>	Is replaced with
c	Using key: <b>15</b>	Is replaced with
o	Using key: <b>8</b>	Is replaced with
d	Using key: <b>25</b>	Is replaced with
i	Using key: <b>25</b>	Is replaced with
n	Using key: <b>0</b>	Is replaced with
g	Using key: <b>15</b>	Is replaced with



# Making the secret message

i	Using key: <b>15</b>	Is replaced with	x
l	Using key: <b>8</b>	Is replaced with	
o	Using key: <b>25</b>	Is replaced with	
v	Using key: <b>25</b>	Is replaced with	
e	Using key: <b>0</b>	Is replaced with	
c	Using key: <b>15</b>	Is replaced with	
o	Using key: <b>8</b>	Is replaced with	
d	Using key: <b>25</b>	Is replaced with	
i	Using key: <b>25</b>	Is replaced with	
n	Using key: <b>0</b>	Is replaced with	
g	Using key: <b>15</b>	Is replaced with	



# Making the secret message

i	Using key: <b>15</b>	Is replaced with	x
l	Using key: <b>8</b>	Is replaced with	t
o	Using key: <b>25</b>	Is replaced with	
v	Using key: <b>25</b>	Is replaced with	
e	Using key: <b>0</b>	Is replaced with	
c	Using key: <b>15</b>	Is replaced with	
o	Using key: <b>8</b>	Is replaced with	
d	Using key: <b>25</b>	Is replaced with	
i	Using key: <b>25</b>	Is replaced with	
n	Using key: <b>0</b>	Is replaced with	
g	Using key: <b>15</b>	Is replaced with	



# Making the secret message

i	Using key: <b>15</b>	Is replaced with	x
l	Using key: <b>8</b>	Is replaced with	t
o	Using key: <b>25</b>	Is replaced with	n
v	Using key: <b>25</b>	Is replaced with	
e	Using key: <b>0</b>	Is replaced with	
c	Using key: <b>15</b>	Is replaced with	
o	Using key: <b>8</b>	Is replaced with	
d	Using key: <b>25</b>	Is replaced with	
i	Using key: <b>25</b>	Is replaced with	
n	Using key: <b>0</b>	Is replaced with	
g	Using key: <b>15</b>	Is replaced with	



# Making the secret message

i	Using key: <b>15</b>	Is replaced with	x
l	Using key: <b>8</b>	Is replaced with	t
o	Using key: <b>25</b>	Is replaced with	n
v	Using key: <b>25</b>	Is replaced with	u
e	Using key: <b>0</b>	Is replaced with	
c	Using key: <b>15</b>	Is replaced with	
o	Using key: <b>8</b>	Is replaced with	
d	Using key: <b>25</b>	Is replaced with	
i	Using key: <b>25</b>	Is replaced with	
n	Using key: <b>0</b>	Is replaced with	
g	Using key: <b>15</b>	Is replaced with	





# Making the secret message

i	Using key: <b>15</b>	Is replaced with	x
l	Using key: <b>8</b>	Is replaced with	t
o	Using key: <b>25</b>	Is replaced with	n
v	Using key: <b>25</b>	Is replaced with	u
e	Using key: <b>0</b>	Is replaced with	e
c	Using key: <b>15</b>	Is replaced with	
o	Using key: <b>8</b>	Is replaced with	
d	Using key: <b>25</b>	Is replaced with	
i	Using key: <b>25</b>	Is replaced with	
n	Using key: <b>0</b>	Is replaced with	
g	Using key: <b>15</b>	Is replaced with	



# Making the secret message

i	Using key: <b>15</b>	Is replaced with	x
l	Using key: <b>8</b>	Is replaced with	t
o	Using key: <b>25</b>	Is replaced with	n
v	Using key: <b>25</b>	Is replaced with	u
e	Using key: <b>0</b>	Is replaced with	e
c	Using key: <b>15</b>	Is replaced with	r
o	Using key: <b>8</b>	Is replaced with	
d	Using key: <b>25</b>	Is replaced with	
i	Using key: <b>25</b>	Is replaced with	
n	Using key: <b>0</b>	Is replaced with	
g	Using key: <b>15</b>	Is replaced with	



# Making the secret message

i	Using key: <b>15</b>	Is replaced with	x
l	Using key: <b>8</b>	Is replaced with	t
o	Using key: <b>25</b>	Is replaced with	n
v	Using key: <b>25</b>	Is replaced with	u
e	Using key: <b>0</b>	Is replaced with	e
c	Using key: <b>15</b>	Is replaced with	r
o	Using key: <b>8</b>	Is replaced with	w
d	Using key: <b>25</b>	Is replaced with	
i	Using key: <b>25</b>	Is replaced with	
n	Using key: <b>0</b>	Is replaced with	
g	Using key: <b>15</b>	Is replaced with	



# Making the secret message

i	Using key: <b>15</b>	Is replaced with	x
l	Using key: <b>8</b>	Is replaced with	t
o	Using key: <b>25</b>	Is replaced with	n
v	Using key: <b>25</b>	Is replaced with	u
e	Using key: <b>0</b>	Is replaced with	e
c	Using key: <b>15</b>	Is replaced with	r
o	Using key: <b>8</b>	Is replaced with	w
d	Using key: <b>25</b>	Is replaced with	c
i	Using key: <b>25</b>	Is replaced with	
n	Using key: <b>0</b>	Is replaced with	
g	Using key: <b>15</b>	Is replaced with	



# Making the secret message

i	Using key: <b>15</b>	Is replaced with	x
l	Using key: <b>8</b>	Is replaced with	t
o	Using key: <b>25</b>	Is replaced with	n
v	Using key: <b>25</b>	Is replaced with	u
e	Using key: <b>0</b>	Is replaced with	e
c	Using key: <b>15</b>	Is replaced with	r
o	Using key: <b>8</b>	Is replaced with	w
d	Using key: <b>25</b>	Is replaced with	c
i	Using key: <b>25</b>	Is replaced with	h
n	Using key: <b>0</b>	Is replaced with	
g	Using key: <b>15</b>	Is replaced with	



# Making the secret message

i	Using key: <b>15</b>	Is replaced with	x
l	Using key: <b>8</b>	Is replaced with	t
o	Using key: <b>25</b>	Is replaced with	n
v	Using key: <b>25</b>	Is replaced with	u
e	Using key: <b>0</b>	Is replaced with	e
c	Using key: <b>15</b>	Is replaced with	r
o	Using key: <b>8</b>	Is replaced with	w
d	Using key: <b>25</b>	Is replaced with	c
i	Using key: <b>25</b>	Is replaced with	h
n	Using key: <b>0</b>	Is replaced with	n
g	Using key: <b>15</b>	Is replaced with	



# Making the secret message

i	Using key: <b>15</b>	Is replaced with	x
l	Using key: <b>8</b>	Is replaced with	t
o	Using key: <b>25</b>	Is replaced with	n
v	Using key: <b>25</b>	Is replaced with	u
e	Using key: <b>0</b>	Is replaced with	e
c	Using key: <b>15</b>	Is replaced with	r
o	Using key: <b>8</b>	Is replaced with	w
d	Using key: <b>25</b>	Is replaced with	c
i	Using key: <b>25</b>	Is replaced with	h
n	Using key: <b>0</b>	Is replaced with	n
g	Using key: <b>15</b>	Is replaced with	v



# Secret Message

So our secret encrypted message is **x tne rwchnv**

To decrypt it you do the same thing with each letter and key that you did to decrypt in the Caesar cipher

- change the key value to become the negative of the encryption key value
- turn the wheel backwards (clockwise) to undo the encryption and get the secret message
- this shifts the alphabet the opposite way to what we did to encrypt the message





# Turn it back!

x	Using key: <b>15</b>	Is replaced with
t	Using key: <b>8</b>	Is replaced with
n	Using key: <b>25</b>	Is replaced with
u	Using key: <b>25</b>	Is replaced with
e	Using key: <b>0</b>	Is replaced with
r	Using key: <b>15</b>	Is replaced with
w	Using key: <b>8</b>	Is replaced with
c	Using key: <b>25</b>	Is replaced with
h	Using key: <b>25</b>	Is replaced with
n	Using key: <b>0</b>	Is replaced with
v	Using key: <b>15</b>	Is replaced with



# Turn it back!

x	Using key: <b>15</b>	Is replaced with	i
t	Using key: <b>8</b>	Is replaced with	
n	Using key: <b>25</b>	Is replaced with	
u	Using key: <b>25</b>	Is replaced with	
e	Using key: <b>0</b>	Is replaced with	
r	Using key: <b>15</b>	Is replaced with	
w	Using key: <b>8</b>	Is replaced with	
c	Using key: <b>25</b>	Is replaced with	
h	Using key: <b>25</b>	Is replaced with	
n	Using key: <b>0</b>	Is replaced with	
v	Using key: <b>15</b>	Is replaced with	



# Turn it back!

x	Using key: <b>15</b>	Is replaced with	i
t	Using key: <b>8</b>	Is replaced with	l
n	Using key: <b>25</b>	Is replaced with	
u	Using key: <b>25</b>	Is replaced with	
e	Using key: <b>0</b>	Is replaced with	
r	Using key: <b>15</b>	Is replaced with	
w	Using key: <b>8</b>	Is replaced with	
c	Using key: <b>25</b>	Is replaced with	
h	Using key: <b>25</b>	Is replaced with	
n	Using key: <b>0</b>	Is replaced with	
v	Using key: <b>15</b>	Is replaced with	



# Turn it back!

x	Using key: <b>15</b>	Is replaced with	i
t	Using key: <b>8</b>	Is replaced with	l
n	Using key: <b>25</b>	Is replaced with	o
u	Using key: <b>25</b>	Is replaced with	
e	Using key: <b>0</b>	Is replaced with	
r	Using key: <b>15</b>	Is replaced with	
w	Using key: <b>8</b>	Is replaced with	
c	Using key: <b>25</b>	Is replaced with	
h	Using key: <b>25</b>	Is replaced with	
n	Using key: <b>0</b>	Is replaced with	
v	Using key: <b>15</b>	Is replaced with	



# Turn it back!

x	Using key: <b>15</b>	Is replaced with	i
t	Using key: <b>8</b>	Is replaced with	l
n	Using key: <b>25</b>	Is replaced with	o
u	Using key: <b>25</b>	Is replaced with	v
e	Using key: <b>0</b>	Is replaced with	
r	Using key: <b>15</b>	Is replaced with	
w	Using key: <b>8</b>	Is replaced with	
c	Using key: <b>25</b>	Is replaced with	
h	Using key: <b>25</b>	Is replaced with	
n	Using key: <b>0</b>	Is replaced with	
v	Using key: <b>15</b>	Is replaced with	



# Turn it back!

x	Using key: <b>15</b>	Is replaced with	i
t	Using key: <b>8</b>	Is replaced with	l
n	Using key: <b>25</b>	Is replaced with	o
u	Using key: <b>25</b>	Is replaced with	v
e	Using key: <b>0</b>	Is replaced with	e
r	Using key: <b>15</b>	Is replaced with	
w	Using key: <b>8</b>	Is replaced with	
c	Using key: <b>25</b>	Is replaced with	
h	Using key: <b>25</b>	Is replaced with	
n	Using key: <b>0</b>	Is replaced with	
v	Using key: <b>15</b>	Is replaced with	



# Turn it back!

x	Using key: <b>15</b>	Is replaced with	i
t	Using key: <b>8</b>	Is replaced with	l
n	Using key: <b>25</b>	Is replaced with	o
u	Using key: <b>25</b>	Is replaced with	v
e	Using key: <b>0</b>	Is replaced with	e
r	Using key: <b>15</b>	Is replaced with	c
w	Using key: <b>8</b>	Is replaced with	
c	Using key: <b>25</b>	Is replaced with	
h	Using key: <b>25</b>	Is replaced with	
n	Using key: <b>0</b>	Is replaced with	
v	Using key: <b>15</b>	Is replaced with	



# Turn it back!

x	Using key: <b>15</b>	Is replaced with	i
t	Using key: <b>8</b>	Is replaced with	l
n	Using key: <b>25</b>	Is replaced with	o
u	Using key: <b>25</b>	Is replaced with	v
e	Using key: <b>0</b>	Is replaced with	e
r	Using key: <b>15</b>	Is replaced with	c
w	Using key: <b>8</b>	Is replaced with	o
c	Using key: <b>25</b>	Is replaced with	
h	Using key: <b>25</b>	Is replaced with	
n	Using key: <b>0</b>	Is replaced with	
v	Using key: <b>15</b>	Is replaced with	





# Turn it back!

x	Using key: <b>15</b>	Is replaced with	i
t	Using key: <b>8</b>	Is replaced with	l
n	Using key: <b>25</b>	Is replaced with	o
u	Using key: <b>25</b>	Is replaced with	v
e	Using key: <b>0</b>	Is replaced with	e
r	Using key: <b>15</b>	Is replaced with	c
w	Using key: <b>8</b>	Is replaced with	o
c	Using key: <b>25</b>	Is replaced with	d
h	Using key: <b>25</b>	Is replaced with	
n	Using key: <b>0</b>	Is replaced with	
v	Using key: <b>15</b>	Is replaced with	



# Turn it back!

x	Using key: <b>15</b>	Is replaced with	i
t	Using key: <b>8</b>	Is replaced with	l
n	Using key: <b>25</b>	Is replaced with	o
u	Using key: <b>25</b>	Is replaced with	v
e	Using key: <b>0</b>	Is replaced with	e
r	Using key: <b>15</b>	Is replaced with	c
w	Using key: <b>8</b>	Is replaced with	o
c	Using key: <b>25</b>	Is replaced with	d
h	Using key: <b>25</b>	Is replaced with	i
n	Using key: <b>0</b>	Is replaced with	
v	Using key: <b>15</b>	Is replaced with	



# Turn it back!

x	Using key: <b>15</b>	Is replaced with	i
t	Using key: <b>8</b>	Is replaced with	l
n	Using key: <b>25</b>	Is replaced with	o
u	Using key: <b>25</b>	Is replaced with	v
e	Using key: <b>0</b>	Is replaced with	e
r	Using key: <b>15</b>	Is replaced with	c
w	Using key: <b>8</b>	Is replaced with	o
c	Using key: <b>25</b>	Is replaced with	d
h	Using key: <b>25</b>	Is replaced with	i
n	Using key: <b>0</b>	Is replaced with	n
v	Using key: <b>15</b>	Is replaced with	



# Turn it back!

x	Using key: <b>15</b>	Is replaced with	i
t	Using key: <b>8</b>	Is replaced with	l
n	Using key: <b>25</b>	Is replaced with	o
u	Using key: <b>25</b>	Is replaced with	v
e	Using key: <b>0</b>	Is replaced with	e
r	Using key: <b>15</b>	Is replaced with	c
w	Using key: <b>8</b>	Is replaced with	o
c	Using key: <b>25</b>	Is replaced with	d
h	Using key: <b>25</b>	Is replaced with	i
n	Using key: <b>0</b>	Is replaced with	n
v	Using key: <b>15</b>	Is replaced with	g



# Your Turn!

Now you try on your own!

**Try doing Lesson 1 of the  
second workbook!**

<https://edstem.org/au/join/zCfRbq>

Your tutors are here to help you if you get  
stuck



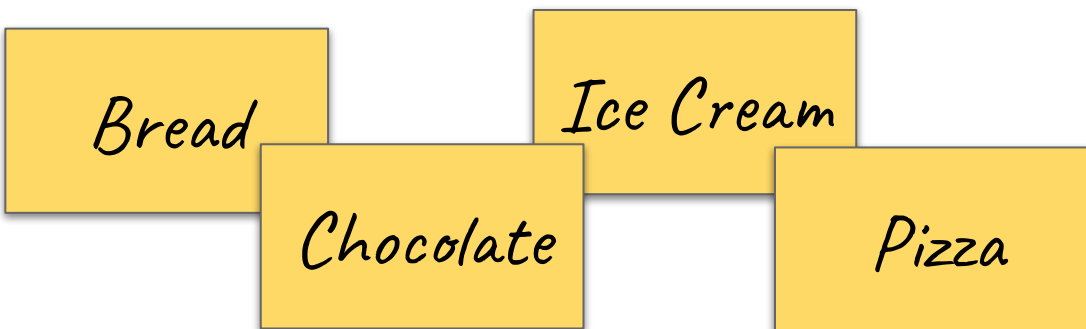
# Lists



# Lists

When we go shopping, we write down what we want to buy!

But we don't store it on lots of little pieces of paper!



We put it in one big shopping list!

- Bread
- Chocolate
- Ice Cream
- Pizza

# Lists

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

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

So much repetition!!

Instead we use a python list!

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





# You can put (almost) anything into a list

- You can have a list of **integers**

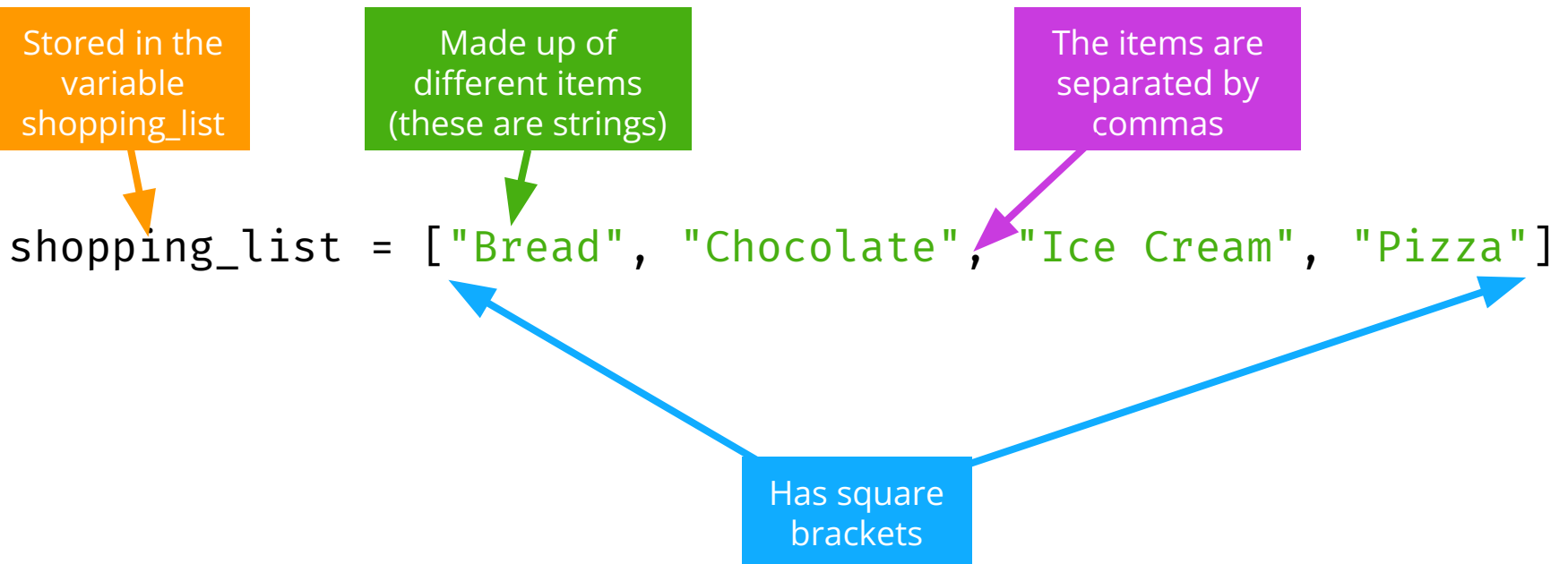
```
>>> primes = [1, 2, 3, 5, 11]
```

- You can have a **list** of **strings**

```
>>> mixture = ["one", "two", "three"]
```

- Every element of a list should be the same (eg integer, string). You should be able to treat every element of the **list** the same way.

# List anatomy



# Accessing Lists!

Make a list of your favourite things

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

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?



# Going Negative

Negative indices count backwards from the end of the **list**

```
>>> faves = ['books', 'butterfly', 'chocolate',  
             'skateboard']  
>>> faves[-1]  
'skateboard'
```

What would `faves[-3]` return?



# 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
```



# Updating items!

We can also update things in a list:

```
>>> faves = ['books', 'butterfly',  
'chocolate', 'skateboard']  
>>> faves[1]  
'butterfly'  
>>> faves[1] = 'kittens'  
>>> faves[1]  
'kittens'
```



# Updating items

What if we decided that we didn't like chocolate anymore, but loved lollipops?



What does this list look like now?





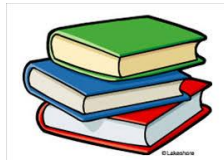
# Removing items!

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.remove('butterfly')
```

What does this list look like now?



# Adding items!

We can also add new items to the list!

What if we decided that we also liked programming?

```
>>> faves.append('programming')
```

What does this list look like now?



# What can you do with a list?

- Define an empty list to add to in your code

```
>>> songs = []
```

- Loop through a list

```
>>> odd_numbers = [1, 3, 5, 7]
```

```
>>> for i in odd_numbers:  
    print(i)
```



# Looping through a list

We can use a for statement to loop through a list

What if we wanted to print out all our favourites?

```
>>> for object in faves:  
    print('I like ' + object)  
  
'books'  
'lollipops'  
'skateboard'  
'programming'
```



# 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]  
>>> first_pair  
["Alex", "Emily"]
```

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

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



# Project time!

You now know all about lists!

**Let's put what we learnt into our project.  
Try Lesson 2 of the second workbook!**

The tutors will be around to help!



# Functions!

Simpler, less repetition, easier to read code!

# How functions fit together!

Functions are like factories!

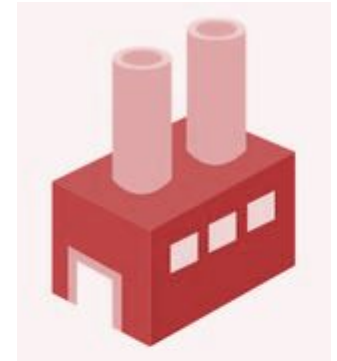
Timber Mill



Your main factory!



Metal Worker



Running a factory doesn't mean doing all the work yourself, you can get other factories to help you out!

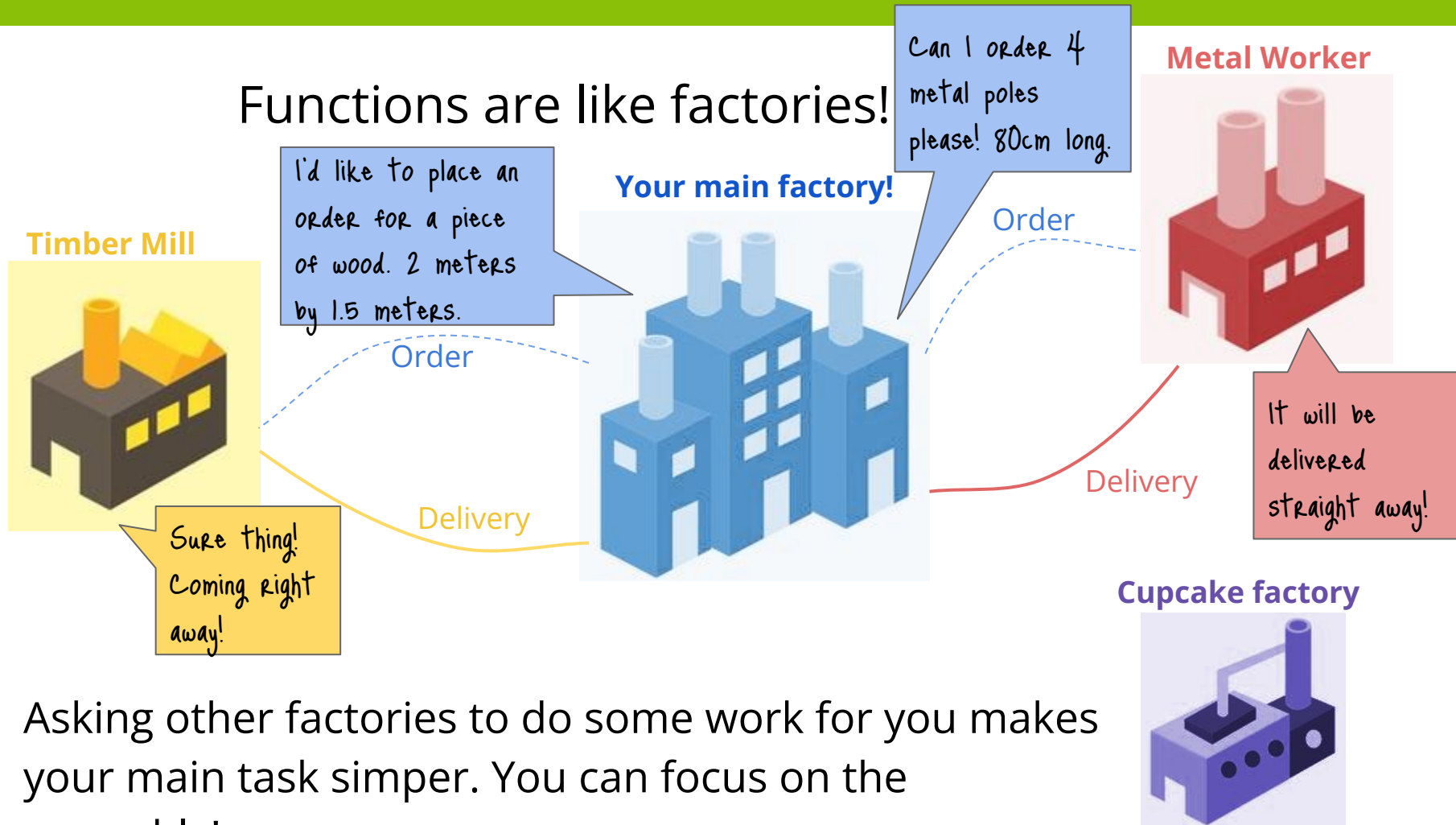
Cupcake factory





# How functions fit together!

Functions are like factories!



Asking other factories to do some work for you makes your main task simpler. You can focus on the assembly!



# How functions fit together!

Functions are like factories!

Timber Mill

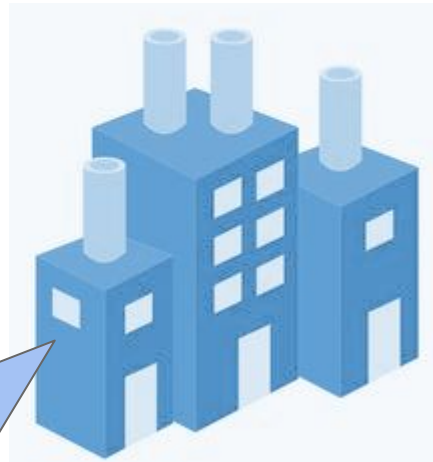


Look at this beautiful table  
I made!

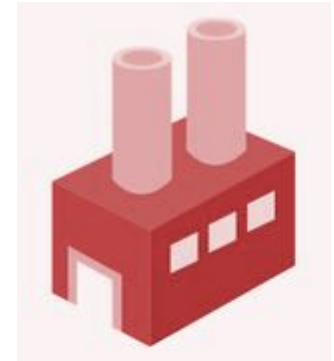


Outsourcing made it simple!

Your main factory!



Metal Worker



Cupcake factory



# How functions fit together!

## Your main code!



You can write a bunch of helpful functions to **simplify** your **main goal**!

You can **write** these **once** and then **use** them **lots** of times!  
They can be **anything** you like!

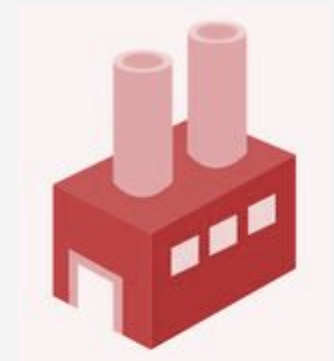
Helps with printing nicely



Uses stats to make decisions



Does calculations



# Don't reinvent the wheel

We're already familiar with some python in built functions like print and input!

**There's lots of functions python gives us to save us reinventing the wheel!**

For instance we can use len to get the length of a string, rather than having to write code to count every letter!

```
>>> len("Hello world")  
11
```

**Try these:**

```
>>> name = "Renee"  
>>> len(name)  
5  
  
>>> int("6")  
6  
  
>>> str(6)  
"6"
```



# Defining your own functions

Built in functions are great! But sometimes we want custom functions!

Defining our own functions means:

- We cut down on repeated code
- Nice function names makes our code clear and easy to read
- We can move bulky code out of the way



# Defining your own functions

Then you can use your function by calling it!

```
def cat_print():  
    print(""  
        #  
        #  
        #  
        ^..^ #####  
        =TT=      ;  
        #####  
        # #      # #  
        M M      M M """)
```

```
cat_print()  
cat_print()
```

Which will do this!

```
          #  
          #  
          #  
        ^..^ #####  
        =TT=      ;  
        #####  
        # #      # #  
        M M      M M  
          #  
          #  
        ^..^ #####  
        =TT=      ;  
        #####  
        # #      # #  
        M M      M M
```

# Defining your own functions

Then you can use your function by calling it!

```
def cat_print():  
    print("""
```

```
                #  
                #  
                #  
    ^..^ #####  
    =TT=      ;  
    #####  
    # #      # #  
    M M      M M  
                #  
                #  
    ^..^ #####  
    =TT=      ;  
    #####  
    # #      # #  
    M M      M M  
    """)
```

```
cat_print()  
cat_print()
```

When using a function in a **script** make sure you define the function first.

It doesn't matter if you call it from inside another function though!

Which will do this!

```
                #  
                #  
                #  
    ^..^ #####  
    =TT=      ;  
    #####  
    # #      # #  
    M M      M M  
                #  
                #  
    ^..^ #####  
    =TT=      ;  
    #####  
    # #      # #  
    M M      M M
```



# Pretty Word Printer

Create a new file and make a pretty word printer! It can print any word you like.

1. Define a function called `pretty_word_print`
2. Set a variable called `word`
3. Have the function print out some decorative marks as long as the word above and below the word like these examples:



```
~~~  
GPN  
~~~  
  
*****  
Hello World  
*****
```

4. Call your function in your file as many times as you like!



# Functions often need extra information

Functions are more useful if we can change what they do

We can do this by giving them arguments (aka parameters)

```
>>> def hello(person):  
    print('Hello, ' + person + ', how are you?')  
>>> hello('Alex')  
Hello, Alex, how are you?
```

Here, we give the hello() function a name

Any string will work

```
>>> hello('abcd')  
Hello, abcd, how are you?
```



# Functions can take multiple arguments

Often we want to work with multiple pieces of information.

You can actually have as many parameters as you like!

This function takes two numbers, adds them together and prints the result.

```
>>> def add(x, y):  
        print(x + y)  
>>> add(3, 4)  
7
```



# Arguments stay inside the function

The arguments are not able to be accessed outside of the function declaration.

```
>>> def hello(person):  
        print('Hello, ' + person + '!')  
>>> print(person)  
Traceback (most recent call last):  
File "<stdin>", line 1, in <module>  
NameError: name 'person' is not defined
```



# Variables stay inside the function

Neither are variables declared inside the function. They are **local variables**.

```
>>> def add(x, y):  
    z = x + y  
    print(z)  
>>> add(3, 4)  
7  
>>> z  
Traceback (most recent call last):  
  File "<stdin>", line 1, in <module>  
NameError: name 'z' is not defined
```



# Global variables are not affected

Changing a variable in a function **only changes it *inside* the function.**

```
>>> z = 1
>>> def add(x, y):
        z = x + y
        print(z)
>>> add(3, 4)
7
```



# Global variables are not affected

Changing a variable in a function **only changes it *inside* the function.**

```
>>> z = 1
>>> def add(x, y):
        z = x + y
        print(z)
>>> add(3, 4)
7
```

What's the value of z now?

```
>>> print(z)
```



# Global variables are not affected

Changing a variable in a function **only changes it *inside* the function.**

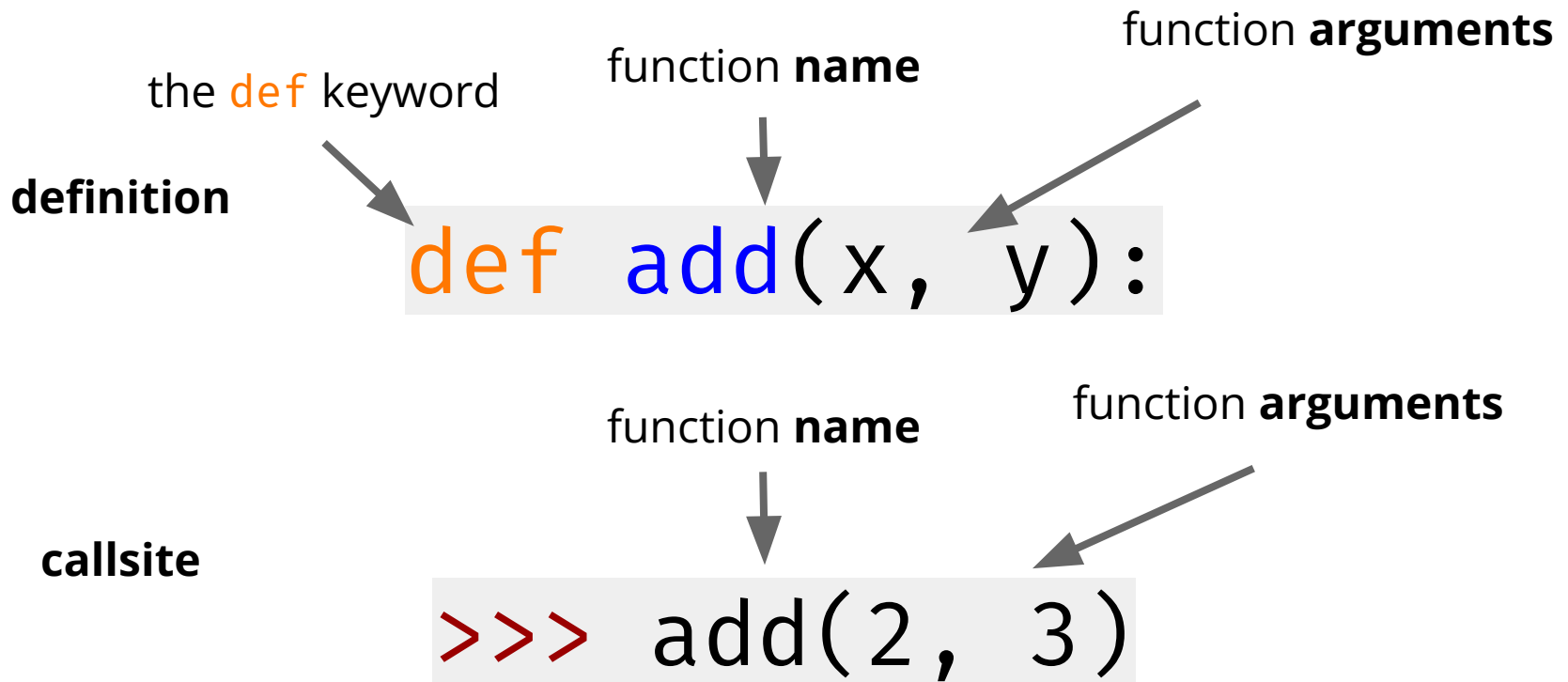
```
>>> z = 1
>>> def add(x, y):
        z = x + y
        print(z)
>>> add(3, 4)
7
```

What's the value of z now?

```
>>> print(z)
1
```



# Recap: A function signature





# Pretty Word Printer

At the moment our pretty word printer always prints the same word. Let's fix that!

## Edit your pretty word printer function:

1. Change your function so it takes in an argument called word
2. Remove the line where you set word as a variable, now we are passing in word
3. Change the places where you called your pretty word printer, so now you pass in a word as an argument (make sure you pass in a string).
4. Try calling your function multiple times, but with different words

**Calling your function with these arguments might look like this:**

```
pretty_word_print("Hi everyone")
pretty_word_print("Coding is cool")
```

```
*****
Hi everyone
*****
*****
Coding is cool
*****
```



# Giving something back

At the moment our function just does a thing, but it's not able to give anything back to the main program.

Currently, we can't use the result of `add()`

```
>>> def add(x, y):  
        print(x + y)  
>>> sum = add(1, 3)  
4  
>>> sum
```

sum has no value!



# Giving something back

Using `return` in a function immediately returns a result.

```
>>> def add(x, y):  
...     z = x + y  
...     return z  
  
>>> sum = add(1, 3)  
>>> sum  
4
```



# Giving something back

When a function returns something, the *control* is passed back to the main program, so no code after the `return` statement is run.

```
>>> def add(x, y):  
    print('before the return')  
    z = x + y  
    return z  
    print('after the return')  
>>> sum = add(1, 3)  
before the return  
>>> sum  
4
```

Here, the `print` statement after the `return` never gets run.



# Project time!

Now you know how to build function!

**Now try Lessons 3 - 6 of  
the second workbook!**

The tutors will be around to help!



# Sets & Files



# Sets

**Sets** are like **lists** without an order and without repetition. They're good when you only want to store one of each thing but don't care where they are.



Let's say you want to store your card hand in poker. The order of cards is not important; you only care if a card is in your hand or not! A **set** lets you look this up quickly.



# Sets

1. Create a set

```
>>> hand = set()
```

2. Add to the set

```
>>> hand.add('A hearts')
```

```
>>> hand.update(['7 diamonds', 'K clubs'])
```

```
>>> hand
```

```
{'K clubs', '7 diamonds', 'A hearts'}
```





# Sets

## 3. Check in set

```
>>> if '7 diamonds' in hand:
    print('Play card')
Play card
```

## 4. Remove from set

```
>>> hand.remove('A hearts')
>>> hand
{'K clubs', '7 diamonds'}
```



# Sets cannot contain things twice

5. Adding the same element again does not change the set

```
>>> hand
{'K clubs', '7 diamonds'}
>>> hand.add('K clubs')

>>> hand
{'K clubs', '7 diamonds'}
```

We cannot have two of the same card in our hand.

# 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!

```
my_file = open("test.txt")
```

You'll now be able to read the things in `my_file`

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:

```
f = open('missing.txt')
```

```
Traceback (most recent call last):
```

```
  File "<stdin>", line 1, in <module>
```

```
IOError: [Errno 2] No such file or  
directory: 'missing.txt'
```



# You can read a whole file into a string

```
>>> my_file = open('haiku.txt')
>>> my_string = f.read()
>>> my_string
'Wanna go outside.\nOh NO!
Help! I got outside!\nLet me
back inside!
```

```
>>> print(my_string)
Wanna go outside.
Oh NO! Help! I got outside!
Let me back inside!
```

haiku.txt

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



# You can also read in one line at a time

**You can use a for loop to only get 1 line at a time!**

```
my_file = open('haiku.txt')  
for line in my_file:  
    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!

```
for line in open('haiku.txt'):
    line = line.strip()
    print(line)
```

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

**No extra lines!**



# Using **with**!

**This is a special trick for opening files!**

```
with open("words.txt") as f:  
    for line in f:  
        print(line.strip())
```

**It automatically closes your file for you!**

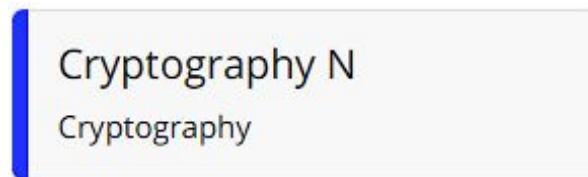
It's good when you are writing files in python!



# Project time!

Now you know how to use files and sets!

**Go file your knowledge into the third workbook!**



<https://edstem.org/au/join/aDq4eg>

The tutors will be around to help!

