

# Welcome to the Labs!

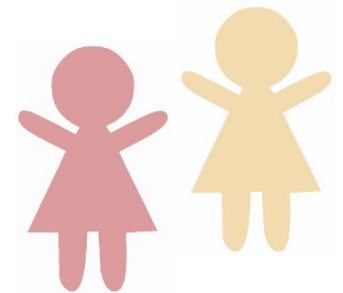
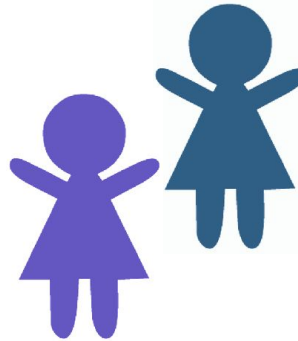
## Tic Tac Toe

# 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 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!

Workshop Name Here

# 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!

# Intro to Programming

# What is programming?



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

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



# A Special Language

A language to talk  
to dogs!



Programming is a  
language to talk to  
computers

# People are smart! Computers are dumb!

## *SALAD INSTRUCTIONS*

Programming is like a recipe!

Computers do EXACTLY what you say, every time.

Which is great if you give them a good recipe!

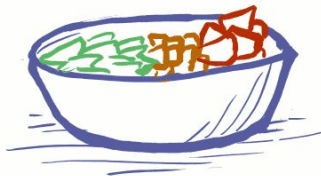
1) GET A LETTUCE HEAD, A CARROT, A TOMATO, A KNIFE, AND A BOWL



2) USE THE KNIFE TO CUT UP THE LETTUCE HEAD, CARROT, AND TOMATO



3) PUT THE LETTUCE, CARROT AND TOMATO IN THE BOWL



4) MIX THE CONTENTS OF THE BOWL



# People are smart! Computers are dumb!

But if you get it out of order....

A computer wouldn't know this recipe was wrong!

## *SALAD INSTRUCTIONS*

1) GET A LETTUCE HEAD, A CARROT, A TOMATO, A KNIFE, AND A BOWL



3) PUT THE LETTUCE, CARROT AND TOMATO IN THE BOWL



2) USE THE KNIFE TO CUT UP THE LETTUCE HEAD, CARROT, AND TOMATO



4) MIX THE CONTENTS OF THE BOWL



# People are smart! Computers are dumb!

Computers are bad at filling in the gaps!

A computer wouldn't know something was missing, it would just freak out!

## *SALAD INSTRUCTIONS*



# Everyone/thing has strengths!



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



- Does exactly what you tell it
- Does it the same every time
- Doesn't need to sleep
- Will work for hours on end
- Doesn't get bored
- Really really fast
- Get smarter when you tell it how

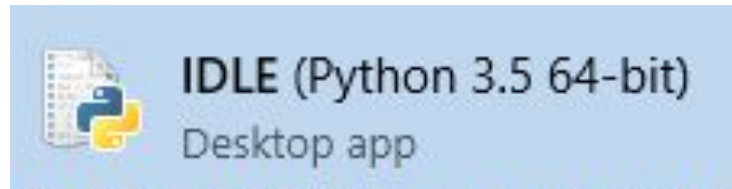


# Intro to Python

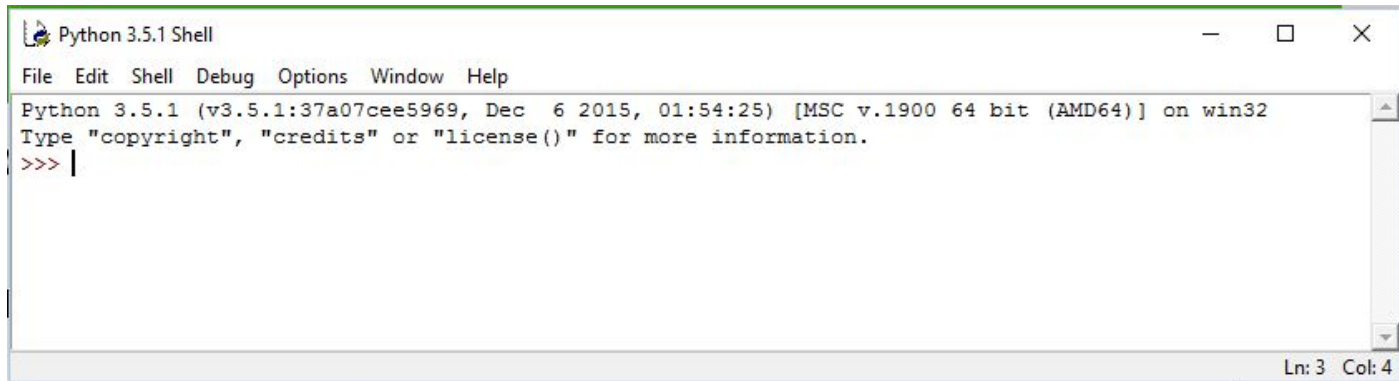
Let's get coding!

# Where do we program? In IDLE

Click the start button and type IDLE!



Make sure the first number after “Python” is 3!



# Make a mistake!

Type by **button mashing** the keyboard!

Then press enter!

```
asdf asdjlkj;pa j;k4uroei
```

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

# Mistakes are great!

*SyntaxError:  
Invalid Syntax*

**Good work you made an error!**

*ImportError:  
No module  
named humour*

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



*KeyError:  
'Hairy Potter'*

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

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



# We can learn from our mistakes!

Error messages help us fix our mistakes!  
We read error messages from bottom to top

Traceback (most recent call last):

File "C:/Users/Madeleine/Desktop/tmp.py", line 9, in <module>  
 print("I have " + 5 + " apples")

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

1. What went wrong

2. What code didn't work

3. Where that code is

# Adding a comment!

Sometimes we want to write things in our file that the computer doesn't look at. We can use **comments** for that!

Sometimes we want to write a note for a people to read

```
# This code was written by Vivian
```

And sometimes we want to not run some code (but don't want to delete it!)

```
# print("Goodbye world!")
```

# Write some code!!

Watch a Tutor type this into the window  
Then press enter!

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

Did it print:

hello world

???

# A calculator for words!

What do you think these bits of code do?

```
>>> "cat" + "dog"
```

```
>>> "tortoise" * 3
```



# A calculator for words!

What do you think these bits of code do?

```
>>> "cat" + "dog"
```

```
catdog
```

```
>>> "tortoise" * 3
```

# A calculator for words!

What do you think these bits of code do?

```
>>> "cat" + "dog"
```

```
catdog
```

```
>>> "tortoise" * 3
```

```
tortoisetortoisetortoise
```

# Strings!

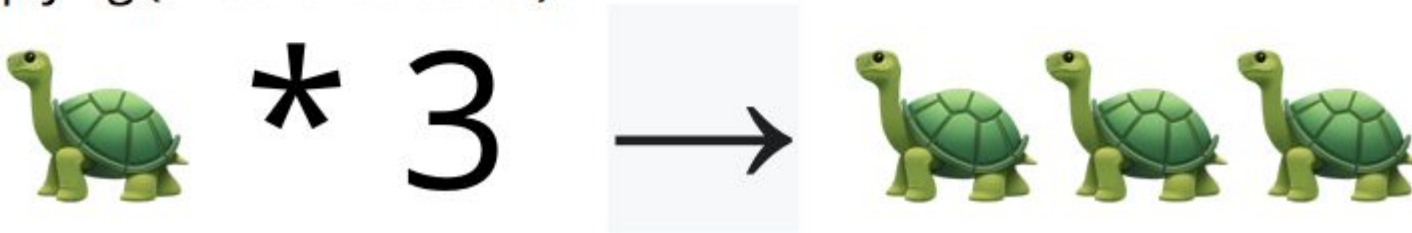
Strings are things with "quotes"

To python they are essentially just a bunch of pictures!

Adding :



Multiplying (3 lots of tortoise!):



# Strings!

Strings can have any letters in them, even just spaces!

```
"Hello, world!"
```

```
"bla bla bla"
```

```
":)"
```

```
" "
```

```
'I can use single quotes too!'
```

```
"~\_(\ツ)\_/~"
```

```
"asdfghjklqwertyuiopzxcvbnm"
```

```
"DOGS ARE AWESOME!"
```

```
"!@#$%^&*()_+--[|\\:;'<>,./?"
```

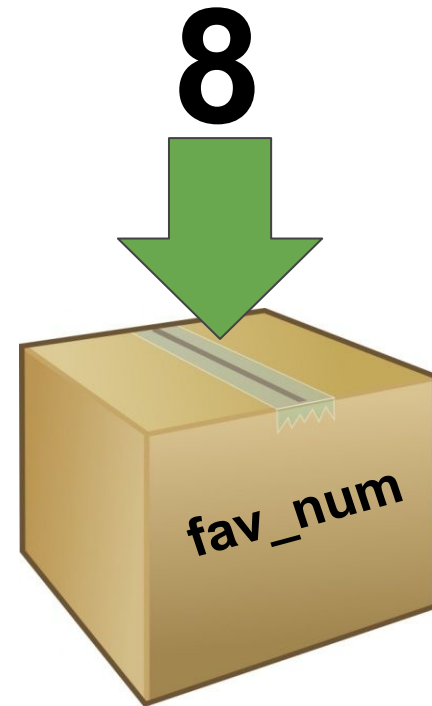
# No Storing is Boring!

**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!**



# Variables

Instead of writing the number 8, we can write fav\_num.



$$\text{fav\_num} - 6 \\ \Rightarrow 2$$

$$\text{fav\_num} + 21 \\ \Rightarrow 29$$

$$\text{fav\_num} * 2 \\ \Rightarrow 16$$

$$\text{fav\_num} / 2 \\ \Rightarrow 4$$



# Variables

Instead of writing the number 8, we can write fav\_num.



fav\_num - 6  
**=> 2**

fav\_num + 21  
**=> 29**

fav\_num \* 2  
**=> 16**

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

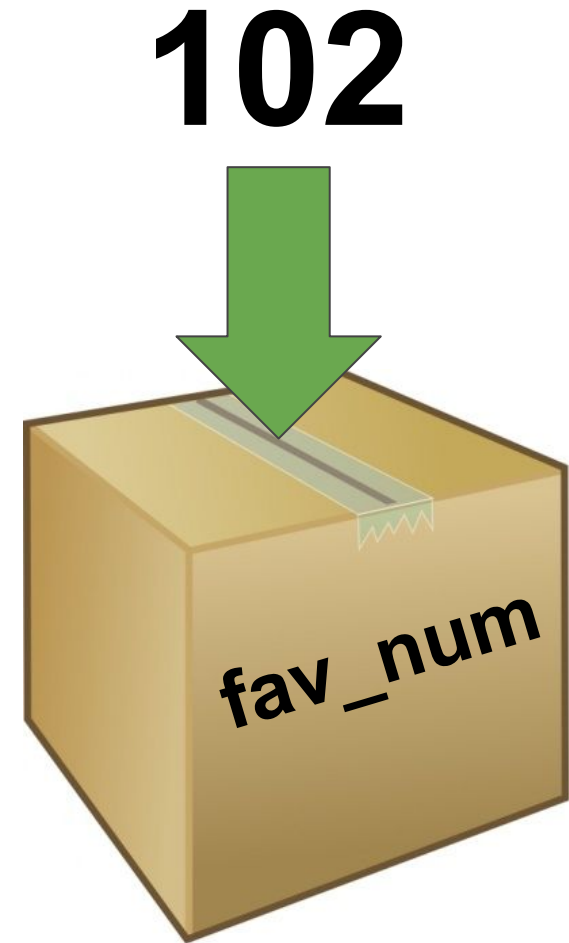


# Variables

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

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

Try changing fav\_num to  
**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**

# No variables VS using variables



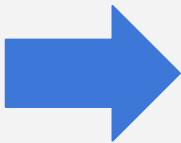
4  
Changes

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
```

# What can we store?

We can put any value in a variable:

```
apples = 5 + 5
print(apples)
apples = apples - 1
print(apples)
apples = "Delicious"
print(apples)
```

What will this output?

# What can we store?

We can put any value in a variable:

```
apples = 5 + 5
print(apples)
apples = apples - 1
print(apples)
apples = "Delicious"
print(apples)
```

10

9

Delicious

# 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
```

# Switcharoo - Making copies!

Set some variables!

```
>>> x = 3
```

```
>>> y = x
```

```
>>> x = 5
```

What do x and y contain now?

Let's find out together!

# Switcharoo - Making copies!

Set some variables!

```
>>> x = 3
```

```
>>> y = x
```

```
>>> x = 5
```

What do x and y contain now?

```
>>> x
```

```
5
```

```
>>> y
```

```
3
```

y hasn't changed  
because it has a  
copy of x in it!

# 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
```

# Project time!

You now know all about the building blocks  
of Python!

**Let's put what we learnt into our project**  
**Try to do the next Part!**

The tutors will be around to help!

# Inputs



# Asking a question!

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

**Let's get the computer to ask us a question!**

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

What do you think happens?

# Asking a question!

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

**Let's get the computer to ask us 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?

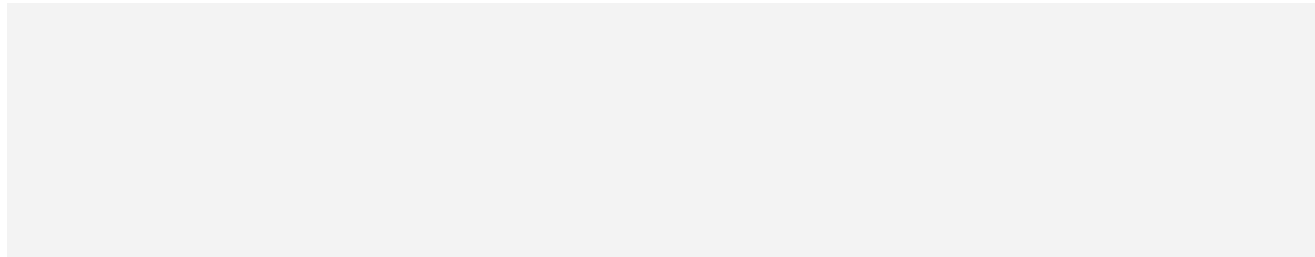
What is your name? Maddie  
Hello Maddie

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?



```
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?

```
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?

```
flavour = input('What cake do you like? ')\nprint(flavour + ' cake for you!')
```

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

# Project time!

You now know all about input!

**Let's put what we learnt into our project**  
**Try to do the next Parts!**

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>	<b>True</b>	<code>"Dog" == "dog"</code>
<code>3 + 2 == 5</code>	<b>True</b>	<code>"D" in "Dog"</code>
<code>5 != 5</code>	<b>False</b>	<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

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



# Conditions

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

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

# Conditions

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

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

That's the  
condition!

# Conditions

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

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

That's the  
condition!

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

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

# Conditions

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

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

Put in the  
answer to  
the question

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

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

# Conditions

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

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

What do you think happens?

```
>>>
```

# Conditions

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

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

What do you think happens?

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

# Conditions

How about a different number???

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



# Conditions

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

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

Put in the  
answer to  
the question

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

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



# Conditions

How about a different number???

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



What do you think happens?

```
>>>
```

# Conditions

How about a different number???

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



What do you think happens?

```
>>>
```

**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!
```

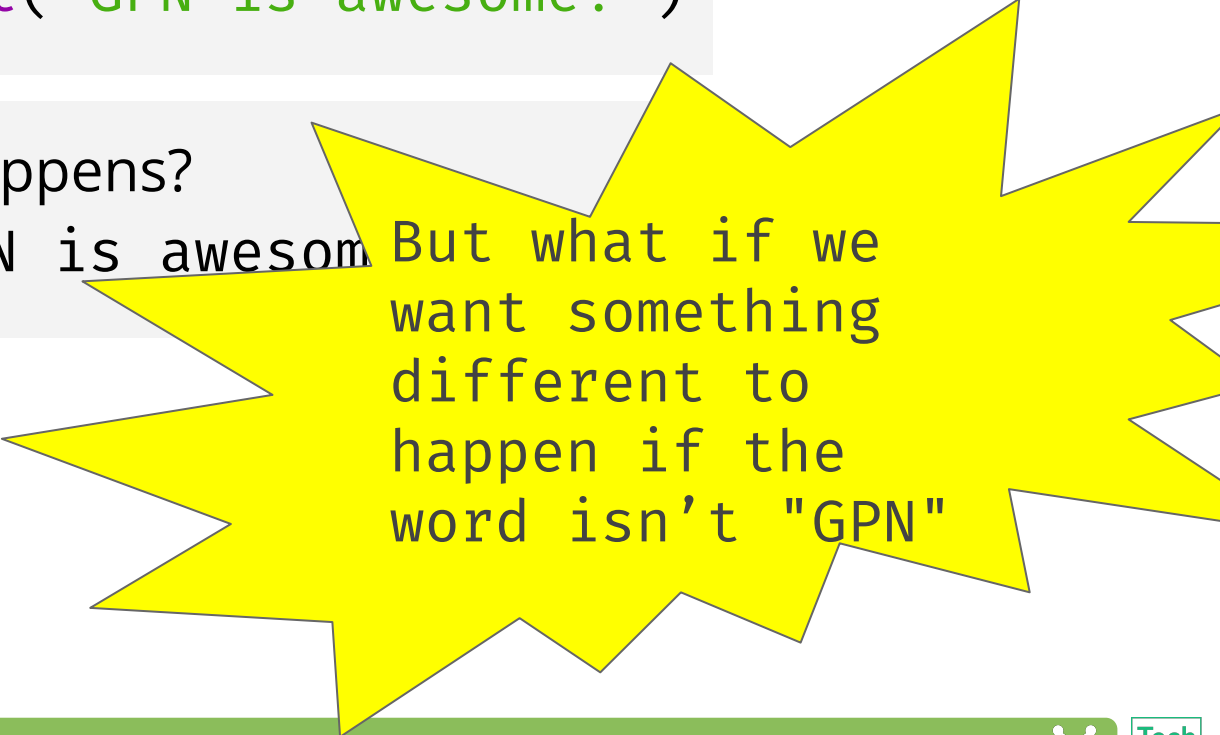


# If statements

```
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

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

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

What happens?

# Else statements

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

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

What happens?

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

# Elif statements

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

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

What happens?

# Elif statements

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

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

What happens?  
>>> YUMMM Chocolate!

# Project Time!

You now know all about **if**!

**See **if** you can do the next Part**

The tutors will be around to help!

# While Loops

# Loops



We know how to do things on repeat!

Sometimes we want to do some code on repeat!



# 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!

1 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
```

MY VARIABLES

~~i = 0~~  
~~i = 1~~  
~~i = 2~~  
i = 3

We are  
are done  
with this  
loop!

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

# Introducing ... while loops!

Initialise the loop variable

Loop condition

Code to repeat

Update the loop variable

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

# 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))
```

[illegible]

# Infinite loop!

Sometimes we want our loop to go forever!

So we set a condition that is always True!

**We can even just write True!**

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

# Infinite loop!

Sometimes we want our loop to go forever!

So we set a condition that is always True!

**We can even just write True!**

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

```
Are we there yet?  
Are we there yet?  
Are we there yet?  
Are we there yet?  
Are we there yet?  
Are we there yet?
```

# Give me a break!

But what if I wanna get out of a loop early?  
That's when we use the **break** keyword!

```
number = 0
while number != 42 :
    number = input("Guess a number: ")

    if number == "I give up":
        print("The number was 42")
        break

    number = int(number)
```

# Continuing on

How about if I wanna skip the rest of the loop body and loop again? We use `continue` for that!

```
number = 0
while number != 42 :
    number = input("Guess a number: ")

    if not number.isnumeric():
        print("That's not a number!")
        print("Try again")
        continue

    number = int(number)
```



# Project Time!

**while** we're here:

**Try to do the next Parts!**

The tutors will be around to help!

# Complex Logic

# 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!

That's all very logical

**Let's put what we learnt into our project**  
**Try to do Part 5 and the Extensions!**

The tutors will be around to help!

Random!

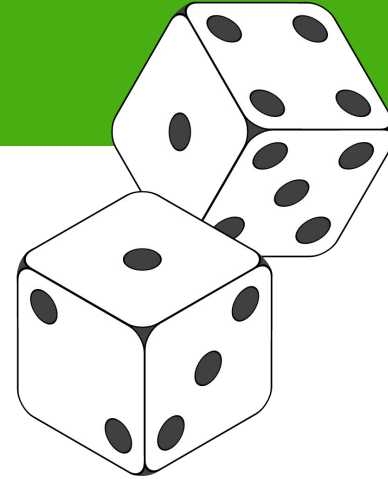


# That's so random!

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



Python lets us **import** common bits of code people use! We're going to use the **random** module!



We want the computer to be random sometimes!



# 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 = ["Bread", "Chocolate", "Ice Cream",  
                    "Pizza"]
```

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 = ["Bread", "Chocolate", "Ice Cream",
                    "Pizza"]
>>> random_food = random.choice(shopping_list)
>>> print(random_food)
```



# Project Time!

Raaaaaaaaaandom! Can you handle that?

**Let's try use it in our project!**  
**Try to do the next Part**

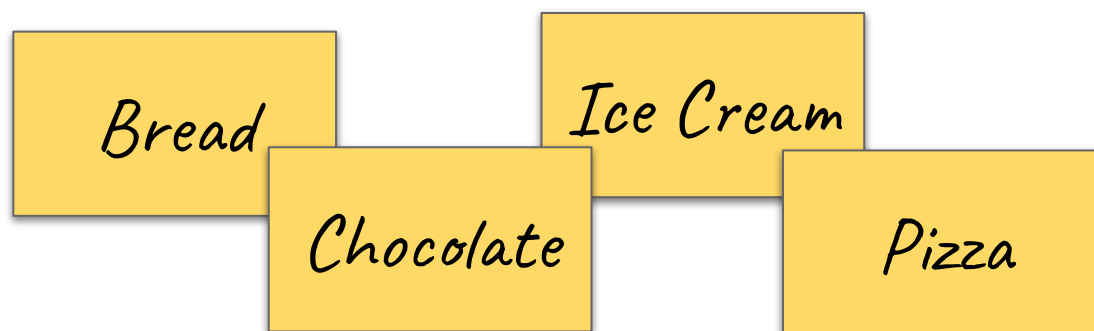
The tutors will be around to help!

# Lists

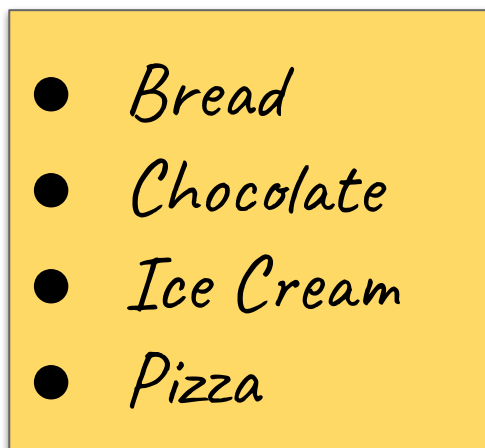
# Lists

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

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



We put it in one big shopping list!



# 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

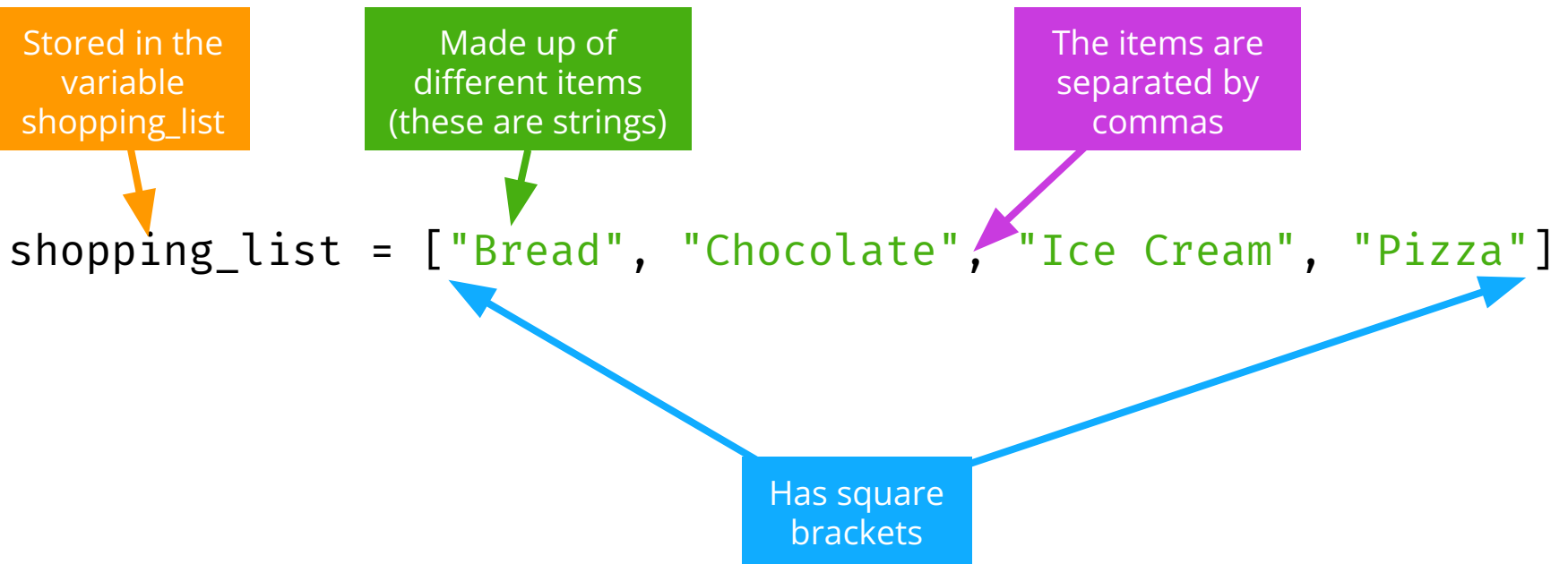
- You can have a list of **integers**  

```
>>> primes = [1, 2, 3, 5, 11]
```
- You can have **lists** with mixed **integers** and **strings**  

```
>>> mixture = [1, 'two', 3, 4, 'five']
```
- But this is almost never a good idea! You should be able to treat every element of the **list** the same way.



# List anatomy



# Removing items!

We can remove items from the list if they're no longer needed!

What if we decided that we didn't like butterflies anymore?

```
>>> faves
```

```
['books', 'butterfly', 'lollipops', 'skateboard']
```

```
>>> faves.remove('butterfly')
```

What does this list look like now?

# Removing items!

We can remove items from the list if they're no longer needed!

What if we decided that we didn't like butterflies anymore?

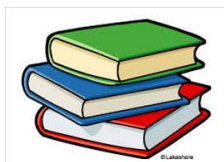
```
>>> faves
```

```
['books', 'butterfly', 'lollipops', 'skateboard']
```

```
>>> faves.remove('butterfly')
```

What does this list look like now?

```
['books', 'lollipops', 'skateboard']
```



# Project time!

You now know all about lists!

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

The tutors will be around to help!