

Welcome to the labs!

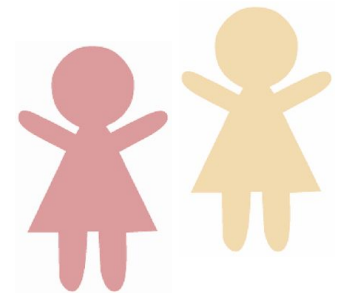
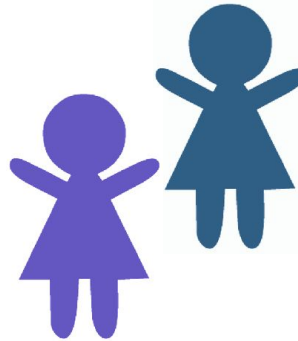
Tic Tac Toe

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

<http://bit.ly/gpn-2019-2>

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!

Tic Tac Toe

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!

What are we doing today?

```

| | | |
| | | |
What symbol are you? o
Where do you want to play? 5
| x | | |
| | o | |
| | | |
| | | |
What symbol are you? x
Where do you want to play? 3
| x | | x |
| | o | |
| | | |
| | | |
What symbol are you? o
Where do you want to play? |

```

How do we get there?

1. Starting the Game

- **Storing** the 9 squares of the Tic-Tac-Toe
- **Printing** the board out

2. The First Turn

- **Asking** the player for their **symbol** and what **square** they want to play in
- **Updating** the stored data for the correct square and printing the board again

3. Lots of turns

- Repeating the first turn so it happens lots of times

4. Make the computer check the winner

- Make the computer look for lines

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!

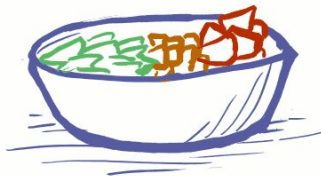
1) GET A LETTUCE HEAD, A CARROT, A TOMATO, A KNIFE, AND A BOWL



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



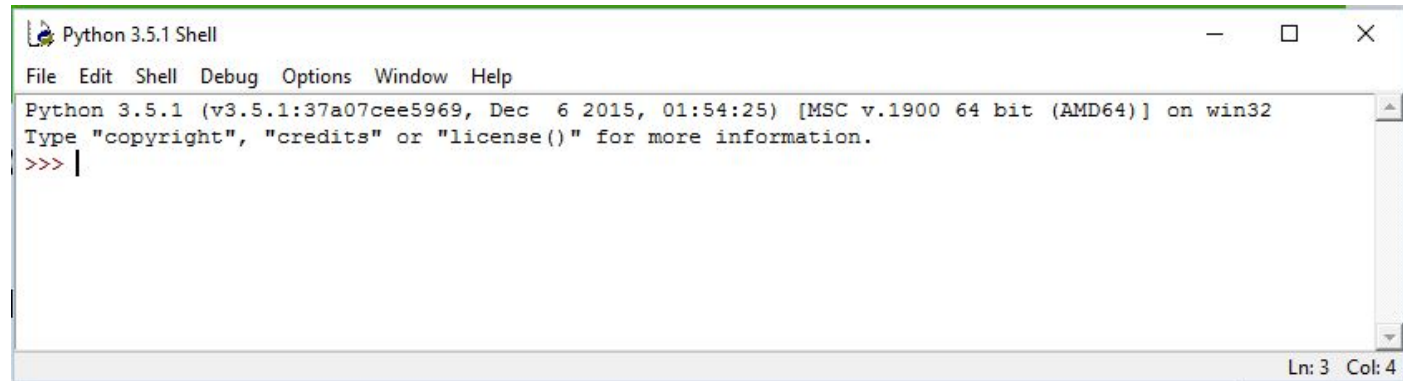
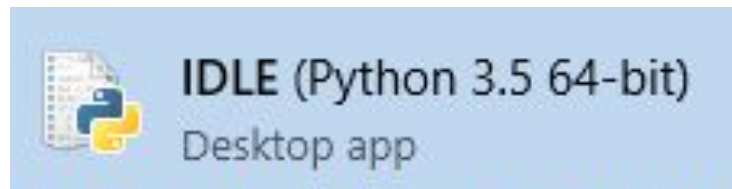
- Does exactly what you tell it
- Does it the same every time
- Doesn't need to sleep!
- Will work for hours on end!
- Get smarter when you tell it how

Intro to Python

Let's get coding!

Where do we program? In IDLE

Click the start button and type IDLE!



Make 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*

Write some code!!

Type this into the window
Then press enter!

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

Did it print:

hello world

???

Tell me more!

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

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

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

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


Python the calculator!

Try writing some maths into python!

```
>>> 1 + 5
```

```
>>> 2 - 7
```

```
>>> 2 * 8
```

```
>>> 12/3
```

A calculator for words!

What do you think these bits of code do?

Try them and see!

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

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

Strings!

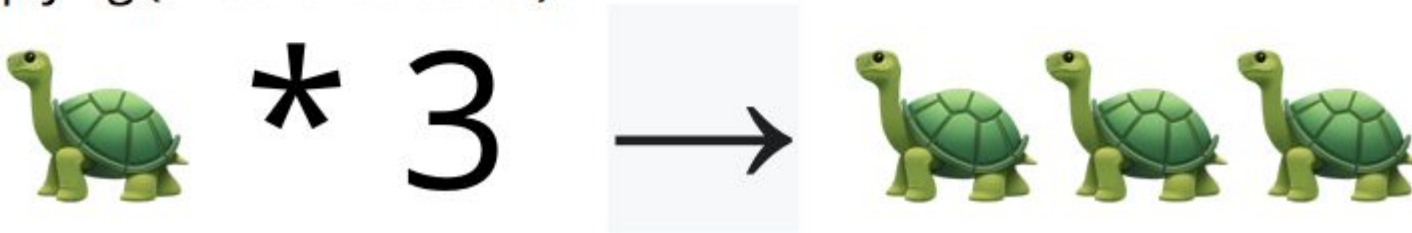
Strings are things with "quotes"

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!"
```

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

Strings and Ints!

Integers are numbers in python.

We can do maths with integers but not strings

```
>>> 5 + "5"
```

We can turn an integer into a string using int()

```
>>> 5 + int("5")
```

Similarly, we turn an int into a string using str()

```
>>> str(5) + "5"
```

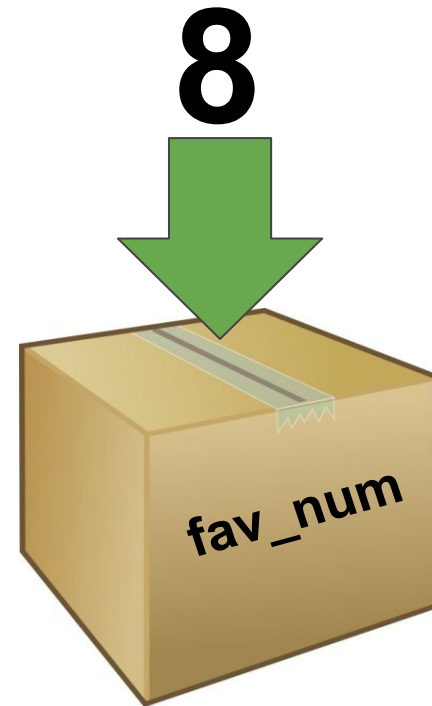
No Storing is Boring!

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

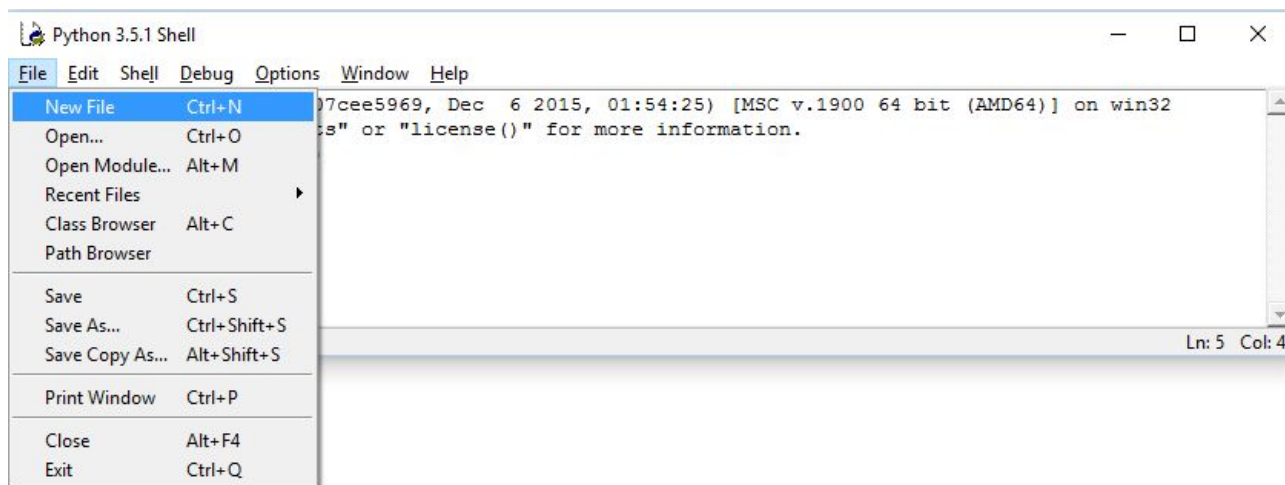
We'll come back to this later!

But writing 8 is
much shorter than
writing fav_num???



Coding in a file!

Code in a file is code we can run multiple times! Make a reusable “hello world”!



1. Make a new file called hello.py, like the picture
2. Put your `print('hello world')` code in it
3. Run your file using the F5 key

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 hello.py file
2. Run your code to make sure it doesn't do anything extra!

Project time!

You now know all about printing and variables!

Let's put what we learnt into our project
Try to do Part 0 and Part 1

The tutors will be around to help!

Inputs and Variables

Variables

Instead of writing the number 8, we can write fav_num.



$$\text{fav_num} - 6 \\ \Rightarrow 2$$

$$\text{fav_num} + 21 \\ \Rightarrow 29$$

$$\text{fav_num} * 2 \\ \Rightarrow 16$$

But writing 8 is
much shorter than
writing fav_num???

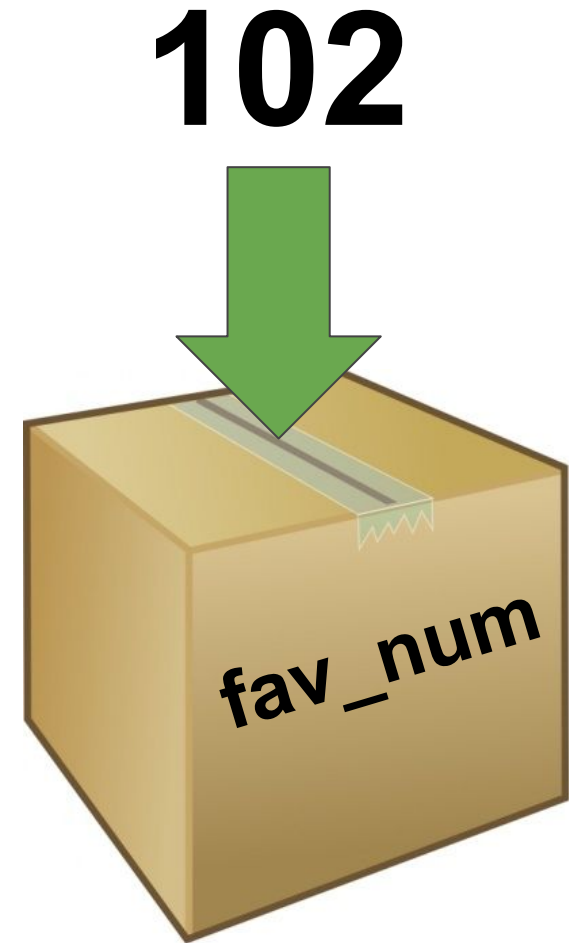


Variables

**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



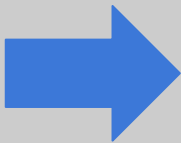
4
Changes

8 - 6

8 * 2

8 + 21

8 / 2



102 - 6

102 * 2

102 + 21

102 / 2



1
Change

```
int fav_num = 8
```

```
fav_num - 6
```

```
fav_num * 2
```

```
fav_num + 21
```

```
fav_num / 2
```



```
int 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?

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?

Variables

Your turn!

Can you guess what each `print` will do?

Type the code into IDLE to check your guesses

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

>>> print(x + x)

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

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

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!

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!

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!
```


Project time!

You now know all about variables!

Let's put what we learnt into our project
Try to do Part 2

The tutors will be around to help!

If Statements

Conditions!

Conditions let us make decision.

First we test if the condition is met!

Then maybe we'll do the thing



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

Try typing these into IDLE!

`5 < 10`

`3 + 2 == 5`

`5 != 5`

`"Dog" == "dog"`

`"D" in "Dog"`

`"Q" not in "Cat"`

Booleans (True and False)

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

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

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

```
>>> phone_book = {"Maddie": 111, "Lucy": 222, "Julia": 333}  
>>> "Maddie" in phone_book  
>>> "Gabe" in phone_book  
>>> 333 in phone_book
```

Booleans (True and False)

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

True

"A" in "AEIOU"

False

"Z" in "AEIOU"

False

"a" in "AEIOU"

False

"banana" in animals

True

"cat" in animals

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

```
>>> phone_book = {"Maddie": 111, "Lucy": 222, "Julia": 333}
```

True

"Maddie" in phone_book

False

"Gabe" in phone_book

False

333 in phone_book

It only checks in the keys!

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

What do you think happens?

```
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?

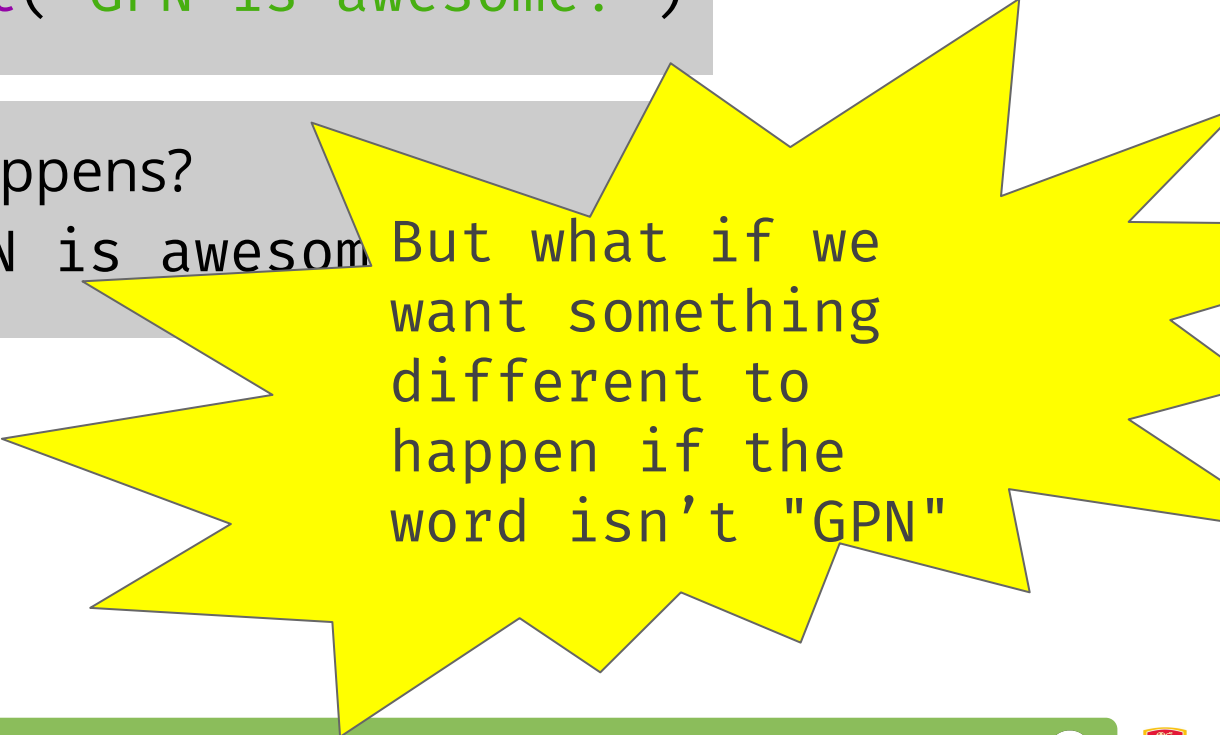
```
>>> GPN is awesome!
```

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

elif

Means we can
give specific
instructions for
other words

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

What happens?

Elif statements

elif

Means we can
give specific
instructions for
other words

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

What happens?

```
>>> YUMMM Chocolate!
```

Practice Time!

1. Create a new file, call it weather.py
2. Copy this code into your file

```
weather = input("What is the weather? ")  
if weather == "raining":
```

3. Add a third line to make it print a special message, but only if the user says "raining"
4. Run your code! Try typing in **raining**, try typing in **sunny**
5. BONUS! Add an else statement, to print a non-rainy message!

Practice Time!

1. Create a new file, call it weather.py
2. Copy this code into your file

```
weather = input("What is the weather? ")  
if weather == "raining":  
    print("Take an umbrella!")
```

3. Add a third line to make it print a special message, but only if the user says "raining"
4. Run your code! Try typing in **raining**, try typing in **sunny**
5. BONUS! Add an else statement, to print a non-rainy message!

Chaining it together

You can check many things are the same or different at the same time by chaining logic operators together.

```
roll1 = 6
roll2 = 6
roll3 = 6
if roll1 == roll2 == roll3:
    print("Three in a row, that's lucky!")
```

What happens?

```
>>>
```

Chaining it together

You can check many things are the same or different at the same time by chaining logic operators together.

```
roll1 = 6
roll2 = 6
roll3 = 6
if roll1 == roll2 == roll3:
    print("Three in a row, that's lucky!")
```

What happens?

```
>>> Three in a row, that's lucky!
```

Project Time!

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

See if you can do Part 3

The tutors will be around to help!

While Loops

Loops



We know how to do things on repeat!

Sometimes we want to do some code on repeat!

Introducing ... while loops!

What do you think this does?

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

Introducing ... while loops!

What do you think this does?

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

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

Introducing ... while loops!

Stepping through a while loop...

Introducing ... while loops!

One step at a time!

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

MY VARIABLES

i = 0

Set the
variable

Introducing ... while loops!

One step at a time!

0 is less
than 3!

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

MY VARIABLES

i = 0

Introducing ... while loops!

One step at a time!

Print!

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

```
i is 0
```

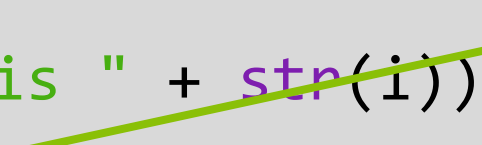
MY VARIABLES

```
i = 0
```


Introducing ... while loops!

One step at a time!

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



MY VARIABLES

~~i = 0~~
i = 1

UPDATE
TIME!

```
i is 0
```

Introducing ... while loops!

One step at a time!

Take it
from the
top!

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

```
i is 0
```

MY VARIABLES

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

Introducing ... while loops!

One step at a time!

i is less
than 3!

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

MY VARIABLES

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

```
i is 0
```

Introducing ... while loops!

One step at a time!

Print!

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

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


MY VARIABLES

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

Introducing ... while loops!

One step at a time!

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



MY VARIABLES

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

UPDATE
TIME!

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

Introducing ... while loops!

One step at a time!

Take it
from the
top!

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

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

MY VARIABLES

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

Introducing ... while loops!

One step at a time!

2 is less
than 3!

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

MY VARIABLES

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

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

Introducing ... while loops!

One step at a time!

Print!

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

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


MY VARIABLES

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


Introducing ... while loops!

One step at a time!

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



MY VARIABLES

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

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

UPDATE
TIME!

Introducing ... while loops!

One step at a time!

Take it
from the
top!

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

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

MY VARIABLES

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

Introducing ... while loops!

One step at a time!

3 IS NOT
less than
3!

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

We are
done
with this
loop!

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

MY VARIABLES

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

Introducing ... while loops!

Initialise the loop variable

Loop condition

Code to repeat

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

Update the loop variable

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?")
```

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 Part 4!

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 go outside?

You might do it like this:



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

But that starts to get messy quickly.

AND

Instead you can do it like this!

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

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

OR

What if you wanted to check if the weather was raining or snowing, and take an umbrella for either possibility.

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

This also doesn't look very nice. If you change what you're printing you have to change it twice.

OR

Instead, you can use OR

```
weather = "snowing"  
if weather == "raining" or weather == "snowing":  
    print("Take an umbrella!")
```

What happens?

```
>>>
```

OR

Instead, you can use OR

```
weather = "snowing"  
if weather == "raining" or weather == "snowing":  
    print("Take an umbrella!")
```

What happens?

```
>>> Take an umbrella!
```

More Complex Logic

You can use any comparison you like with **and** and **or**, such as `>` or `!=`

You can even use **or** to check different variables like this:

```
toys = 14
books = 20
if toys > 10 or books > 10:
    print("That's a lot!")
```

What happens?

```
>>>
```

More Complex Logic

You can use any comparison you like with **and** and **or**, such as `>` or `!=`

You can even use **or** to check different variables like this:

```
toys = 14
books = 20
if toys > 10 or books > 10:
    print("That's a lot!")
```

What happens?

```
>>> That's a lot!
```

Logic Operators

<code>a == b</code>	Checks if a equals b.
<code>a != b</code>	Checks if a is different to b.
<code>a >= b</code>	Checks if a is equal to or greater than b. Only works for numbers.
<code>a > b</code>	Checks if a is greater than b. Only works for numbers.
<code>a <= b</code>	Checks if a is equal to or less than b. Only works for numbers.
<code>a < b</code>	Checks if a is less than b. Only works for numbers.

Project time!

That's all very logical

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

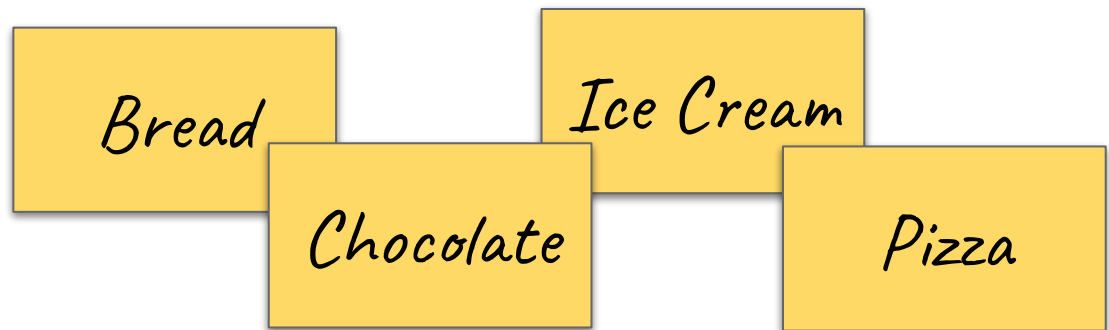
The tutors will be around to help!

Lists and Random

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!

-
- The diagram shows a single large yellow rectangular box containing a bulleted list of the same four shopping items. The items are listed in the same cursive font as in the previous diagram.
- 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"]
```

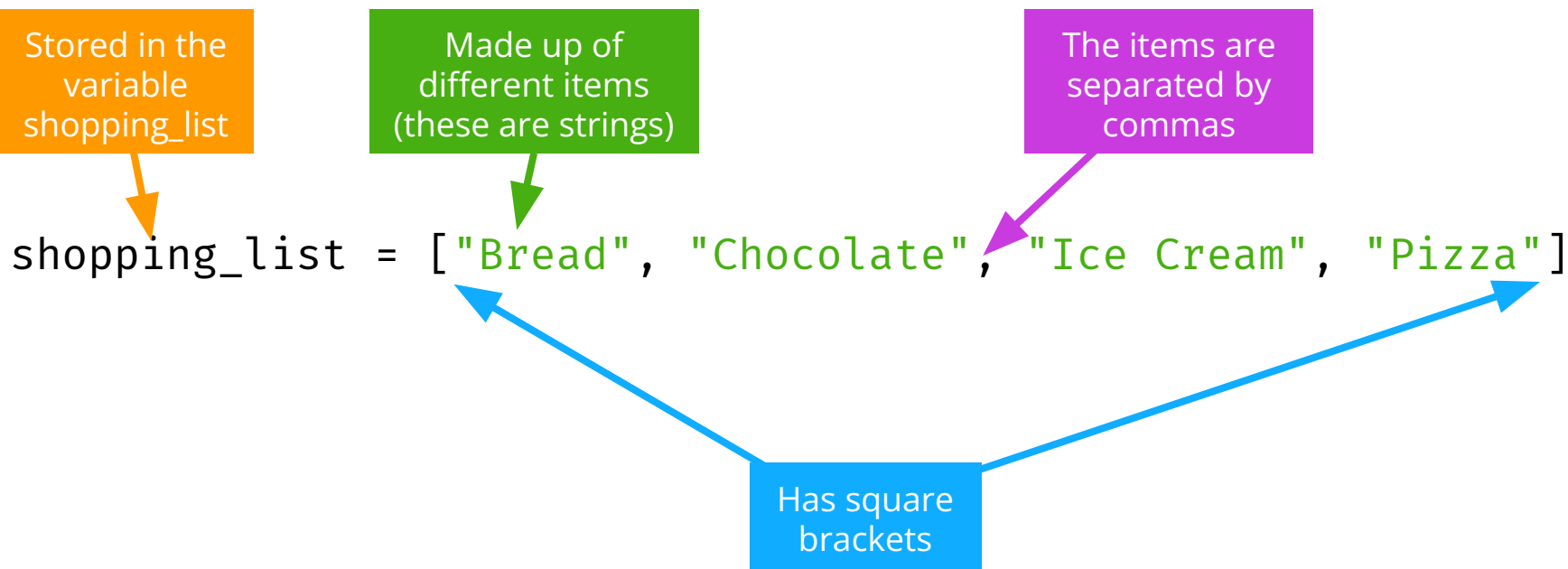
You can put (almost) anything into a list

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

```
>>> primes = [1, 2, 3, 5, 11]
```
- You can have **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

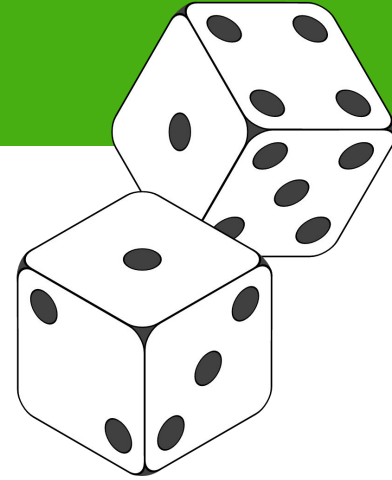


That's so random!

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 = ["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 try use it in our project!
Try to do Extension 9 onwards

The tutors will be around to help!