Welcome to the labs!

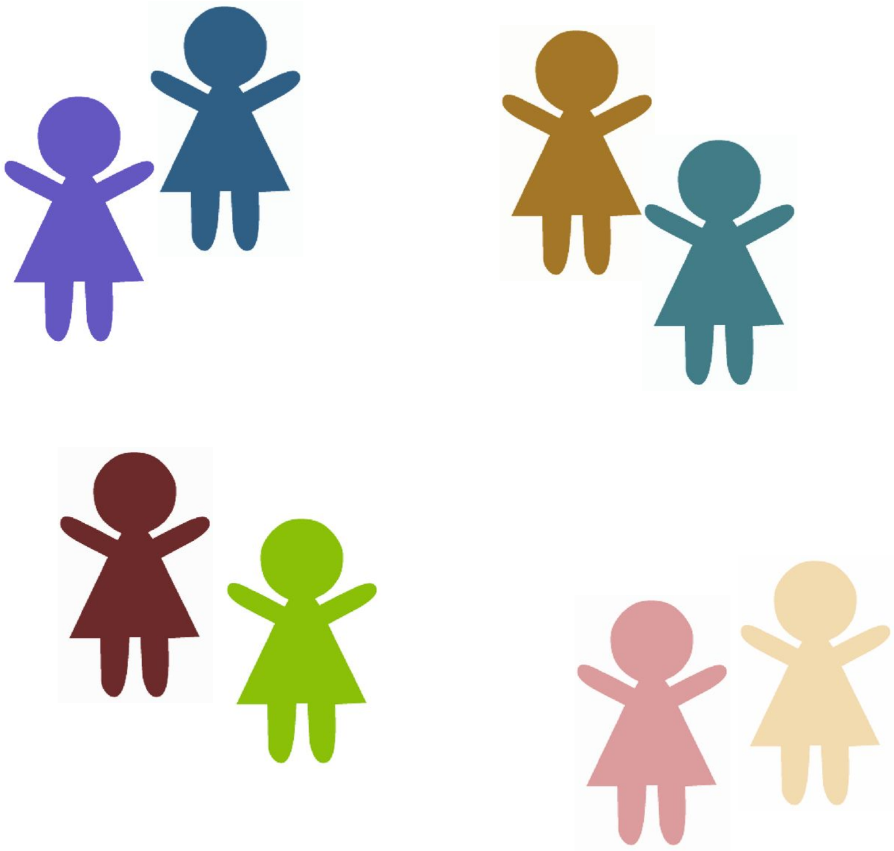
Cryptography

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

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!

Cryptography

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!

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**

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 6.1: 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.

☑ **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

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.

★ **BONUS 4.3: Do some extra!**

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

Intro to Caesar Ciphers

Let's get encoding!

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!

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

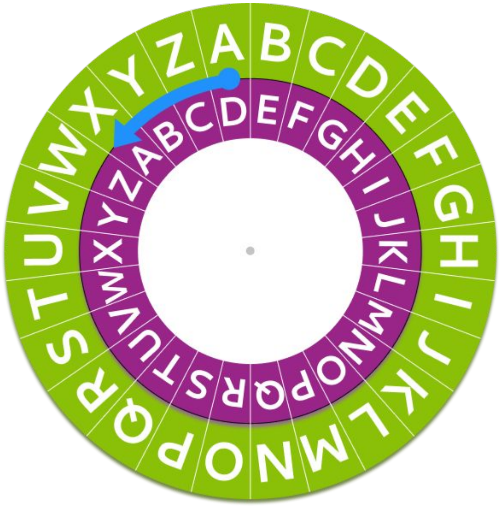
So what's a Caesar Cipher?

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

Let's learn how it works!

Cipher Wheels

You each have a cipher wheel that looks 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
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

You can spin the inside set of letters around and make them line up with different letters

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

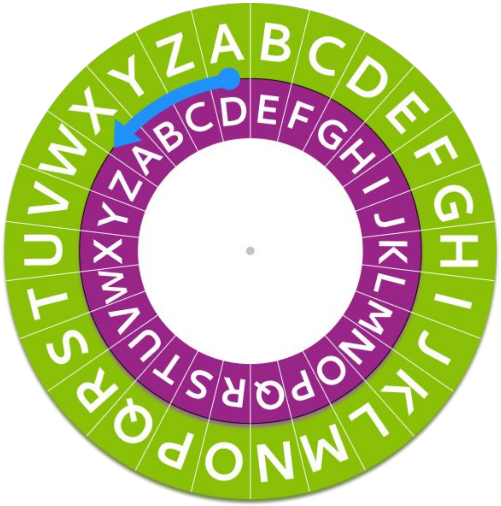
Try turning your purple wheel 3 letters **anti-clockwise** so that you have your letters lining up like this!

Making the secret message

Now, let’s write a secret message!

I love coding

For our Caesar cipher we take each letter and replace it with the letter that has been shifted


Y Z A B C D E F G H I J K L M N O P Q R S T U V W X
C 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

So, let’s start with the letter i

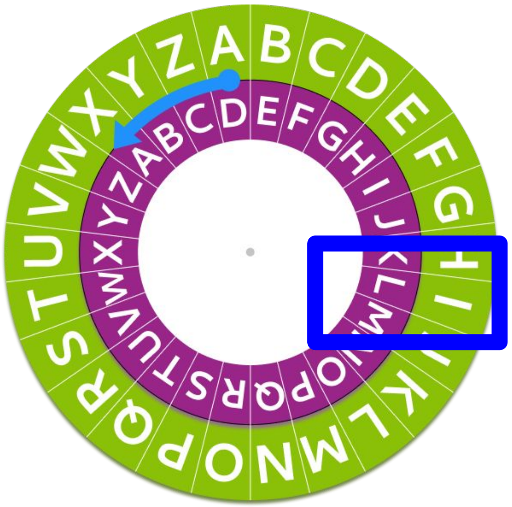
What new letter should we use to replace it?

Making the secret message

Now, let's write a secret message!

I love coding

For our Caesar cipher we take each letter and replace it with the letter that has been shifted



So, let's start with the letter i

What new letter should we use to replace it?

The letter L

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	
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	
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	l
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!

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 the *other* way (clockwise) to decrypt the message!

Let's check that it works with: L oryh frglqj

Remember that the key is 3!

Turn it back!

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	

Turn it back!

l	Is replaced with	i
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	

Turn it back!

l	Is replaced with	i
o	Is replaced with	l
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	

Turn it back!

l	Is replaced with	**i**
o	Is replaced with	**l**
r	Is replaced with	**o**
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	

Turn it back!

l	Is replaced with	i
o	Is replaced with	l
r	Is replaced with	o
y	Is replaced with	v
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	

Turn it back!

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

Turn it back!

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

Fun fact!

Turning the wheel **backwards**

is the same as

reading your wheel **inside out**!

Your Turn!

Now you try on your own!

Try doing Part 0 of the workbook using your cipher wheels!

Your tutors are here to help you if you get stuck

Intro to Programming

What is programming?

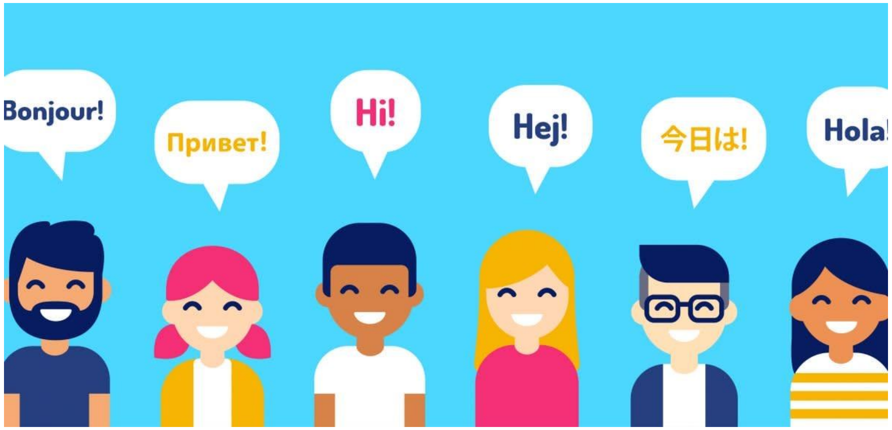
Programming is not a bunch of crazy numbers!

It's giving computers a set of instructions!

A special language

Humans have languages like English, French, Spanish, Mandarin

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



https://images.saymedia-content.com/.image/t_share/MTc0MTAyNzI3ODUxMjU1MjQx/how-to-easily-learn-a-language.jpg

Problem solving

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

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

People are smart, computers are dumb!

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

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

Everyone/thing has strengths!

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

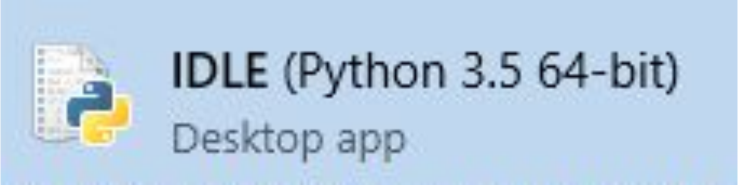
- Incomplete instructions are not okay
- Improves when you tell it how to

Intro to Python

Let’s get coding!

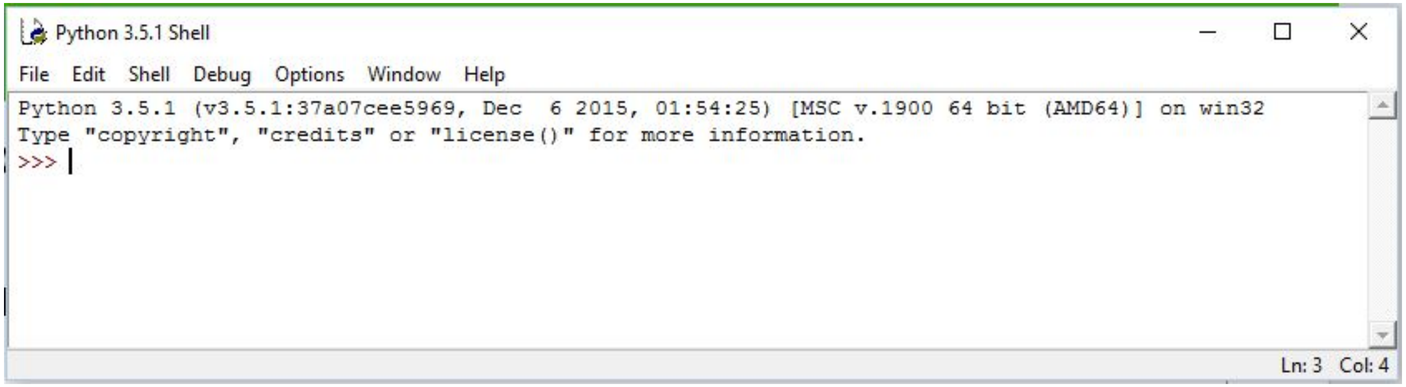
Where do we program? In IDLE

Click the start button and type IDLE!



IDLE (Python 3.5 64-bit)

Desktop app



```
Python 3.5.1 (v3.5.1:37a07cee5969, Dec  6 2015, 01:54:25) [MSC v.1900 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> 
```

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?

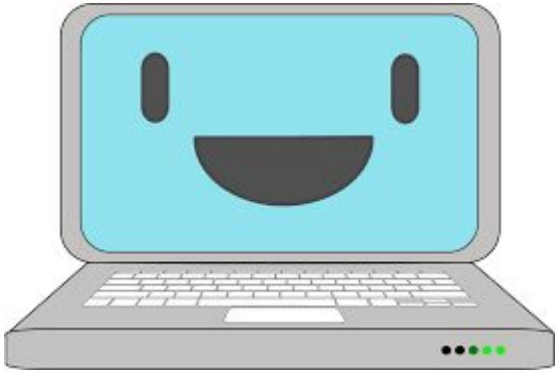
Mistakes are great!

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
```

???

Data types

In programming, we have special names for the following:

Number → Integer

Letter → Character

Word → String

Let’s look at some examples

Characters - not always letters

What do all of these have in common?

“A”   ‘6’   “f”   ‘$’

Anything that only takes up only one space and is surrounded by ‘single’ or “double” quotes, is considered a **character** by the computer.

```
type("5") = char
type(5) = int
```

You can check the data type Python sees it as using the type() command

Strings

Strings are a group of more than one **character** put together and surrounded with `"quotes"`

All of these are strings:

```
"Dog"
          "my name is"
                         "123 hahaha"
                                         "$%#^&@(){}[]"
```

A calculator for words!?

What do you think these bits of code do?

Try them and see!

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

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

Calculator for... words!?

What do you think these bits of code do?

Try them and see!

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

>>> "tortoise" * 3
```

Calculator for… words!?

What do you think these bits of code do?

Try them and see!

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

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

Calculator for words and number?

If we can do calculations with numbers, and calculations with words, can we do calculations with words *and* numbers?

Try writing this!

```
>>> 1 + "1"
```

```
>>> "100" * 2
```

How do we deal with this problem? See next slide!

Type casting

We tell the computer exactly what type we want to use!

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

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

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

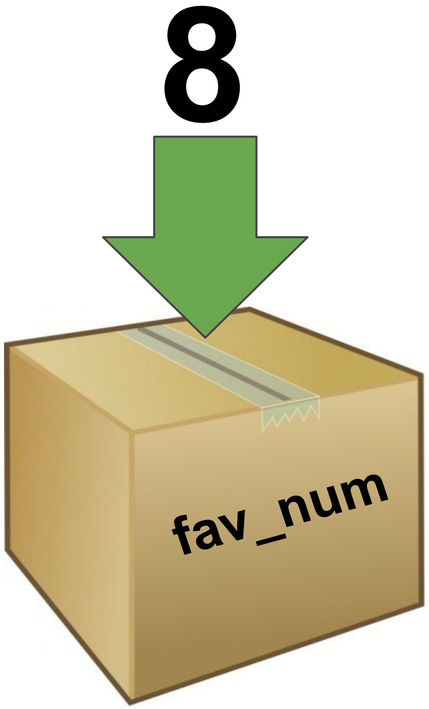
```
>>> str(5) + "5"
```

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.

```
fav_num - 6
    => 2
```

```
fav_num + 21
    => 29
```

```
fav_num * 2
   => 16
```

```
fav_num / 2
    => 4
```

Variables

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

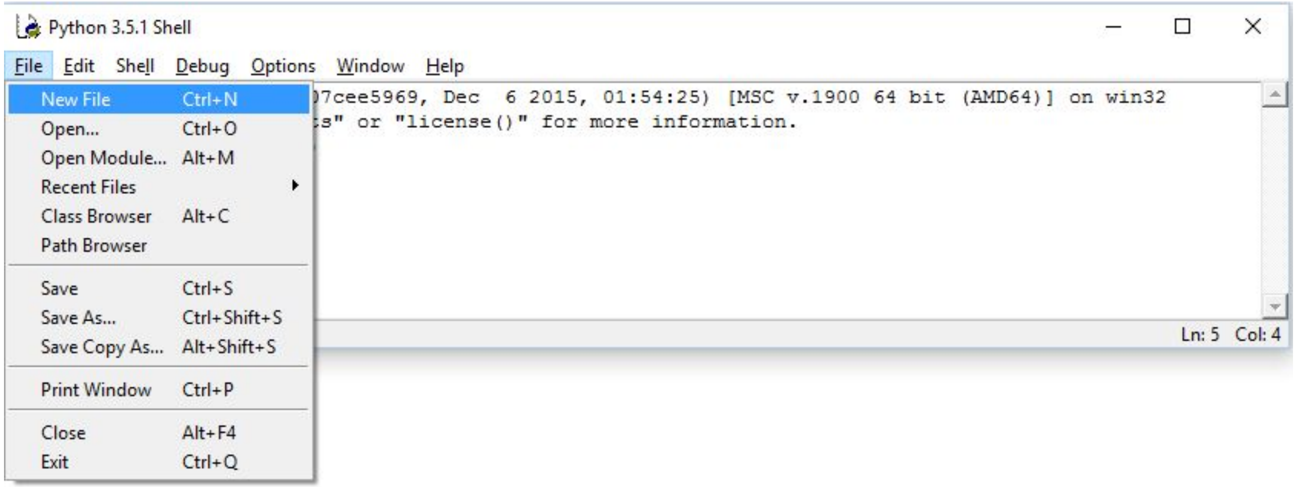
```
fav_num - 6
   => 2

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”!



```
)7cee5969, Dec  6 2015, 01:54:25) [MSC v.1900 64 bit (AMD64)] on win32
:s" or "license()" for more information.
```

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!

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? ')
print('Hello ' + my_name)
```

What do you think happens?

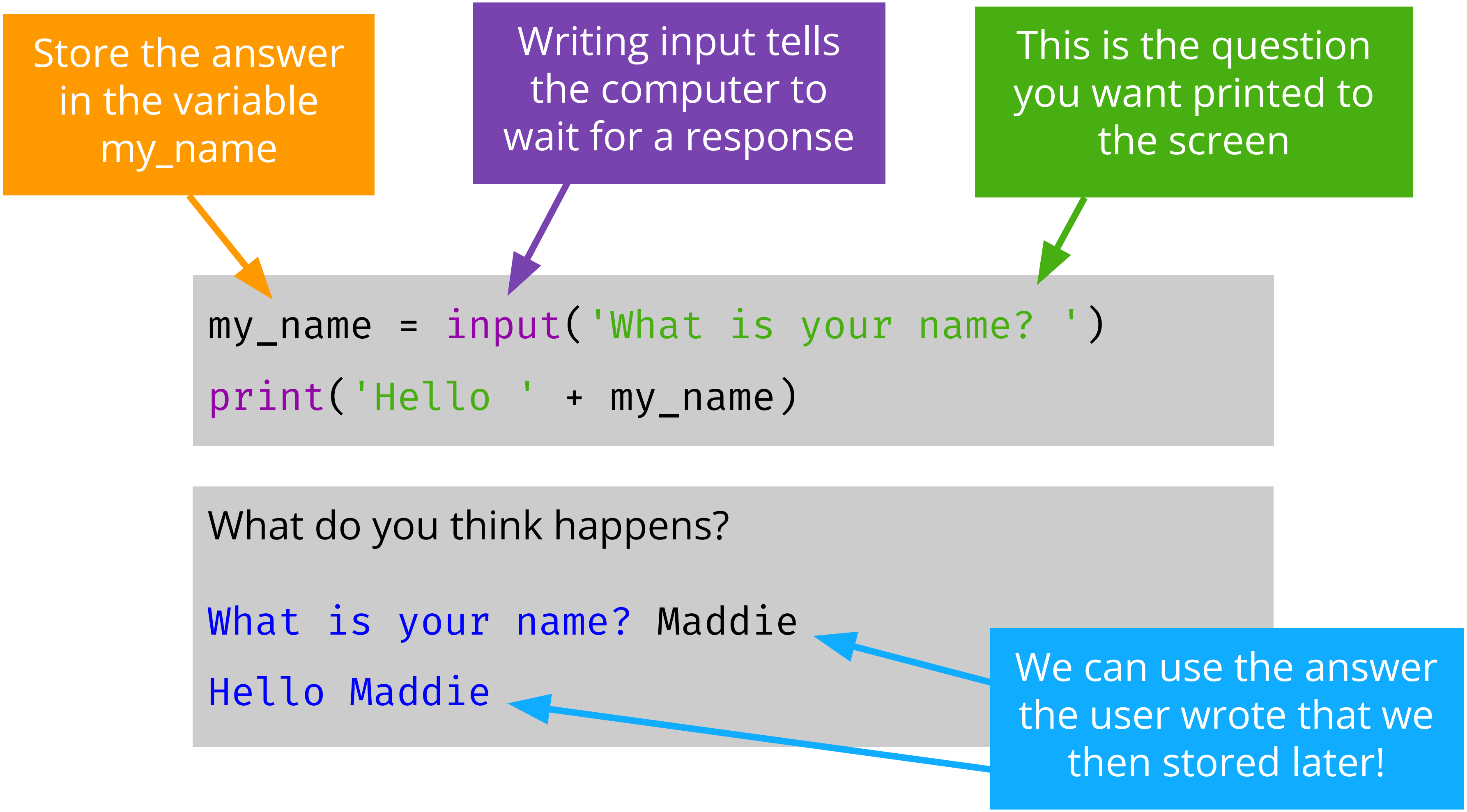
Asking a question!

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

What do you think happens?

```
What is your name? Maddie
Hello Maddie
```

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

Asking a question!

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

How would we print their answer?

Give it a try on your own computer first!

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

Asking a question!

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

How would we print their answer?

Give it a try on your own computer first!

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

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

Asking a question!

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

How would we print their answer?

Give it a try on your own computer first!

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

print(flavour + "cake for you"!)
```

```
What cake do you like? chocolate

chocolate cake for you!
```

Project time!

You now know all about printing and variables!

Let's put what we learnt into our project

Try to do Part 1 - Part 2

The tutors will be around to help!

More Strings and Ints

More maths!

There are many different ways we can divide numbers in python:

```
>>> 5 / 3

>>> 5 // 3

>>> 5 % 3
```

More maths!

There are many different ways we can divide numbers in python:

```
>>> 5 / 3
1.6666666666666667 (normal division)
>>> 5 // 3


>>> 5 % 3
```

More maths!

There are many different ways we can divide numbers in python:

```
>>> 5 / 3
1.6666666666666667 (normal division)
>>> 5 // 3
1 (division without remainder)
>>> 5 % 3
```

More maths!

There are many different ways we can divide numbers in python:

```
>>> 5 / 3
1.6666666666666667 (normal division)
>>> 5 // 3
1 (division without remainder)
>>> 5 % 3
2 (remainder from division)
```

Cutting Strings

We can get individual letters from a string using indexes.

```
>>> yum = "chocolate"
>>> yum[0]


>>> yum[5]


>>> yum[-1]


>>> yum[500]
```

Cutting Strings

We can get individual letters from a string using indexes.

```
>>> yum = "chocolate"
>>> yum[0]
'c'
>>> yum[5]


>>> yum[-1]


>>> yum[500]
```

Computers start counting from 0, not 1!

Cutting Strings

We can get individual letters from a string using indexes.

```
>>> yum = "chocolate"
>>> yum[0]
'c'
>>> yum[5]
'l'
>>> yum[-1]


>>> yum[500]
```

Computers start counting from 0, not 1!

Cutting Strings

We can get individual letters from a string using indexes.

```
>>> yum = "chocolate"
>>> yum[0]
'c'
>>> yum[5]
'l'
>>> yum[-1]
'e'
>>> yum[500]
```

Computers start counting from 0, not 1!

Cutting Strings

We can get individual letters from a string using indexes.

```
>>> yum = "chocolate"
>>> yum[0]
'c'
>>> yum[5]
'l'
>>> yum[-1]
'e'
>>> yum[500]
IndexError: string index out of range
```

Computers start counting from 0, not 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)


>>> yum[9 - 1]


>>> yum[10 % len(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)
9
>>> yum[9 - 1]


>>> yum[10 % len(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)
9
>>> yum[9 - 1]
'e'
>>> yum[10 % len(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)
9
>>> yum[9 - 1]
'e'
>>> yum[10 % len(yum)]
'h'
```

Notice how we used the remainder from dividing by the length to count again from the beginning of the word?

Project time!

You now know all about strings and ints!

Let’s put what we learnt into our project

Try to do Part 3

The tutors will be around to help!

For Loops

Looping through lists!

What would we do if we wanted to print out this list, one word at a time?

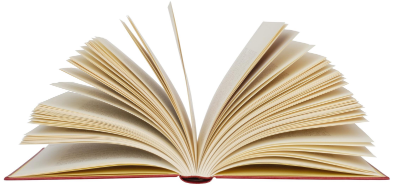
```
words = ['This', 'is', 'a', 'sentence']


print(words[0])
print(words[1])
print(words[2])
print(words[3])
```

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

For Loops

For loops allow you to do something for **each** item in a **group** of things

There are many real world examples, like:



For each page in this book:

Read

For each chip in this bag of chips:

Eat

Looping over a list of ints

We can loop through a list:

```
numbers = [1, 2, 3, 4]
for i in numbers:
    print(i)
```

What’s going to happen?

Looping over a list of ints

We can loop through a list:

```
numbers = [1, 2, 3, 4]
for i in numbers:
    print(i)
```

What’s going to happen?

```
>>> 1
>>> 2
>>> 3
>>> 4
```

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

Looping over a list of ints

Strings are lists of letters!

```python
word = "cat"
for i in word:
   print(i)
```

What’s going to happen?

Looping over a list of ints

Strings are lists of letters!

```
word = "cat"
for i in word:
   print(i)
```

What’s going to happen?

```
>>> c
>>> a
>>> t
```

Practice Time!

1. Make a new file called yummy.py

2. Copy in this list

```
>>> fruits = ['apple', 'banana', 'mango']
```

3. Add **2 lines of code** that makes your program print out this. Use a for loop!

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

HINT!

```
numbers = [1, 2, 3, 4]
for i in numbers:
    print(i)
```

How does it work??

Somehow it knows how to get one fruit out at a time!!

It's like it knows english!

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

But fruit is just a variable! We could call it anything! Like dog!

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

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

How does it work??

Everything in the list gets to have a turn at being the dog variable

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

Let's set dog to to the first thing in the list!
dog is now 'apple'!

How does it work??

Everything in the list gets to have a turn at being the dog variable

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

```
>>> Yummy apple
```

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

dog is now 'apple'!

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

How does it work??

Everything in the list gets to have a turn at being the dog variable

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

```
>>> Yummy apple
```

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

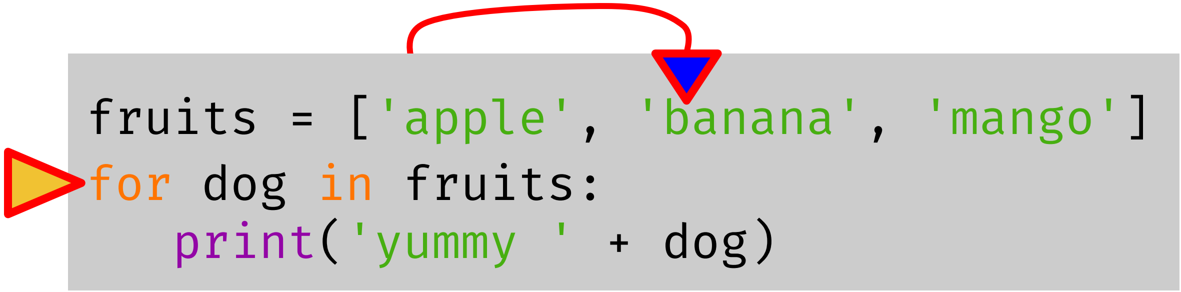
dog is now 'apple'!

`print('yummy ' + dog)`

We're at the end of the loop body, back to the top!

How does it work??

Everything in the list gets to have a turn at being the dog variable



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

```
>>> Yummy apple
```

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

Let’s set dog to to the next thing in the list!
dog is now 'banana'!

How does it work??

Everything in the list gets to have a turn at being the dog variable

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

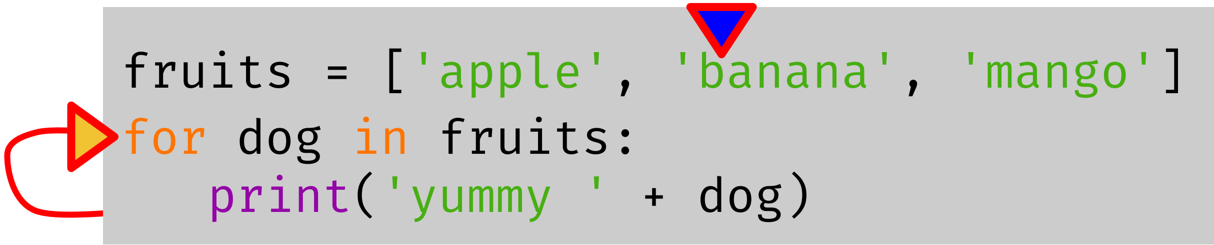
```
>>> Yummy apple
>>> Yummy banana
```

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

Let's set dog to to the next thing in the list!
dog is now 'banana'!
`print('yummy ' + dog)`

How does it work??

Everything in the list gets to have a turn at being the dog variable



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

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

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

Let's set dog to to the next thing in the list!
dog is now 'banana'!
print('yummy ' + dog)
Out of body, back to the top!

How does it work??

Everything in the list gets to have a turn at being the dog variable

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

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

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

Let's set dog to to the next thing in the list!
dog is now 'banana'!
print('yummy ' + dog)
Out of body, back to the top!

Let's set dog to to the next thing in the list!
dog is now 'mango'!

How does it work??

Everything in the list gets to have a turn at being the dog variable

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

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

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

Let's set dog to to the next thing in the list!
dog is now 'banana'!
`print('yummy ' + dog)`
Out of body, back to the top!

Let's set dog to to the next thing in the list!
dog is now 'mango'!
`print('yummy ' + dog)`

How does it work??

Everything in the list gets to have a turn at being the dog variable

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

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

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

Let's set dog to to the next thing in the list!
dog is now 'banana'!
`print('yummy ' + dog)`
Out of body, back to the top!

Let's set dog to to the next thing in the list!
dog is now 'mango'!
`print('yummy ' + dog)`
Out of body, and out of list!! We're done here!

Project Time!

Now you know how to use a for loop!

Try to do Part 4

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

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

<u>If it's raining</u> 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"
```

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

```
>>> 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")
```

<u>What do you think happens?</u>

```
>>>
```

Conditions

How about a different number???

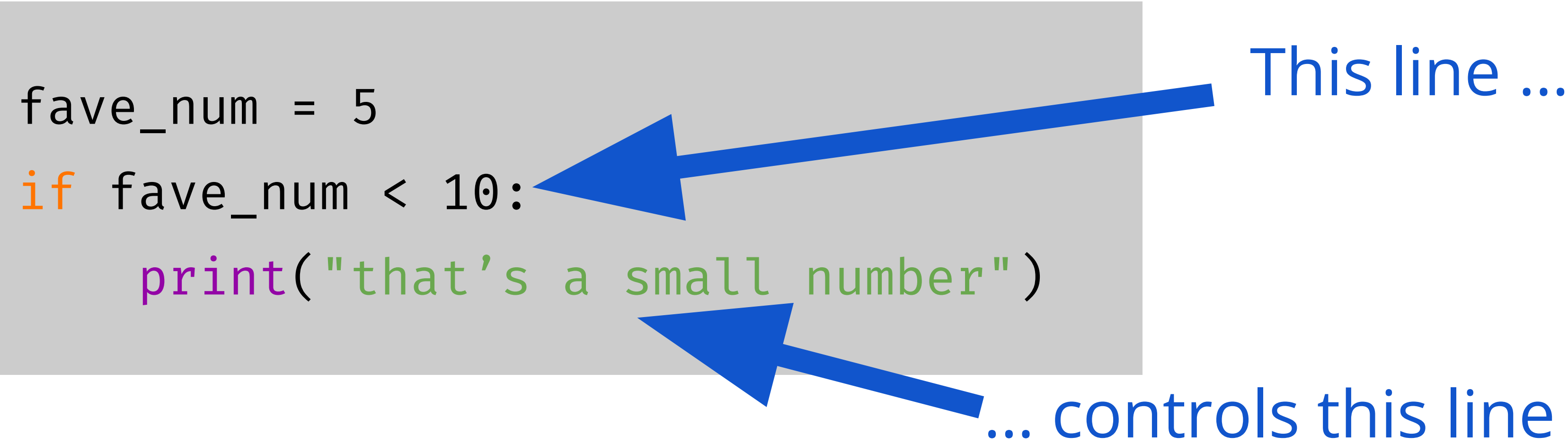
```
fave_num = 9000
if fave_num < 10:
    print("that’s a small number")
```

<u>What do you think happens?</u>

```
>>>
```

Nothing!

If statements



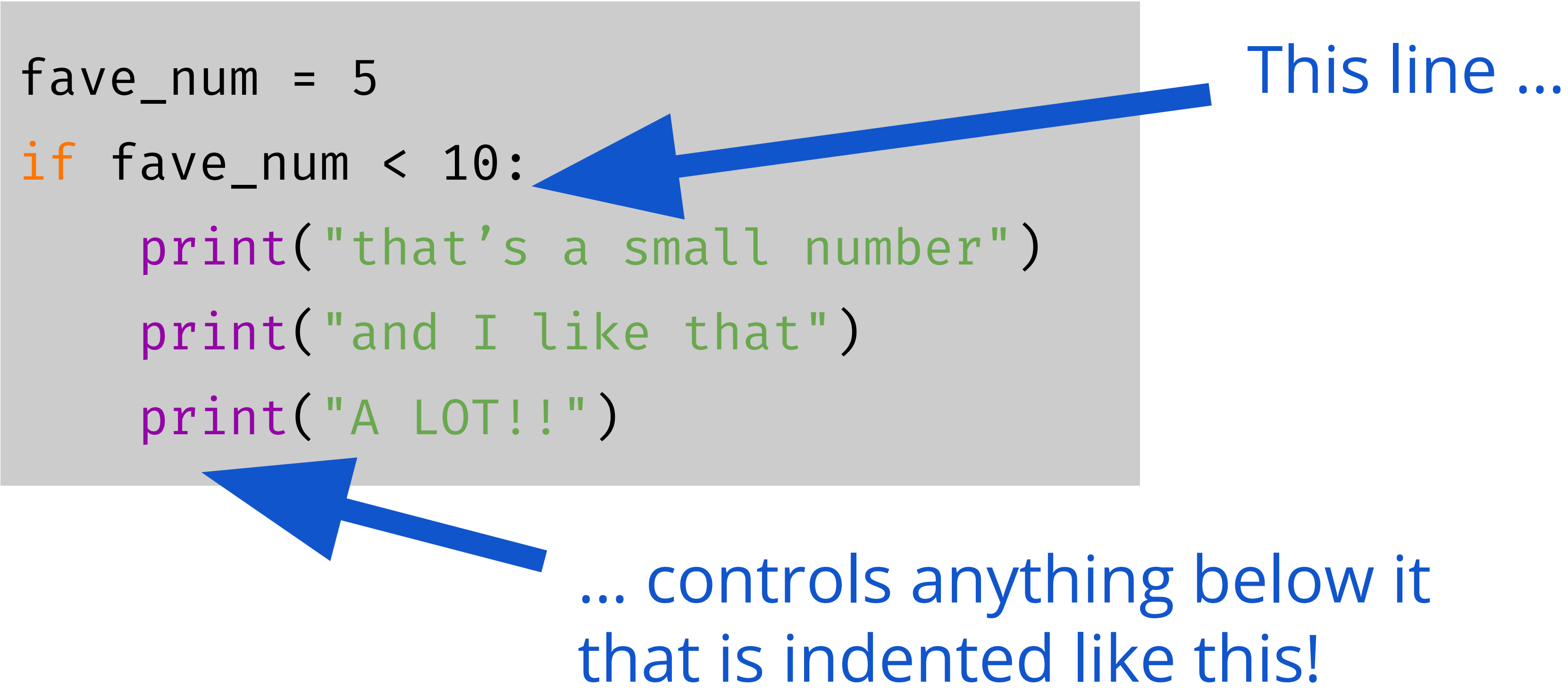
```
fave_num = 5
if fave_num < 10:
    print("that’s a small number")
```

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 awesom
```

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?

Practice Time!

1. Create a new file, call it weather.py

2. Copy this code into your file

```python
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!

Project Time!

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

See `if` you can do

Part 5 - Part 6

The tutors will be around to help!