

EDULITO

A Complete Guide to Python Programming Part 1

1. Python -The Basics
2. The use of variables, constants, operators, inputs, outputs and assignments
3. The use of data types
4. The use of basic string manipulation

```
Python - The Basics
# This program demonstrates the use of variables, constants, operators, inputs, outputs and assignments
# It also shows how to use the range function to create a list of numbers
# The program will calculate the sum of all numbers from 1 to 100 and print the result

# Define a constant for the upper limit
UPPER_LIMIT = 100

# Initialize a variable to hold the sum
total_sum = 0

# Loop through numbers from 1 to 100 and calculate the sum
for i in range(1, UPPER_LIMIT + 1):
    total_sum = total_sum + i

# Print the final sum
print("The sum of numbers from 1 to 100 is:", total_sum)
```

Photocopiable Resources

Terms and Conditions of Use

Your school has permission to copy this resource as many times as you require and to use it as you wish within your school/organisation.

You do not have permission to distribute it as a paper or electronic document to other schools or organisations.

1. Python Basics

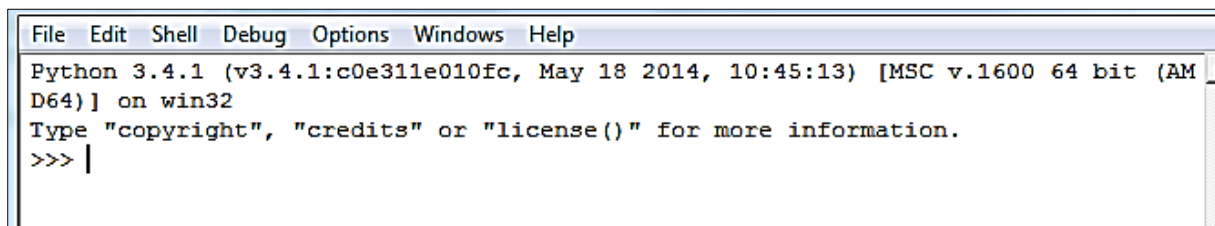
What is Python?

Python is a commonly used high level programming language that has been used to code apps such as Instagram.

To learn how to code using Python you will need to install a Python IDE on to your computer e.g. Python IDLE.

When you first open Python you will see the **Interactive** window or **shell**. This interactive window is ideal for writing short, simple programs.

It looks like this:

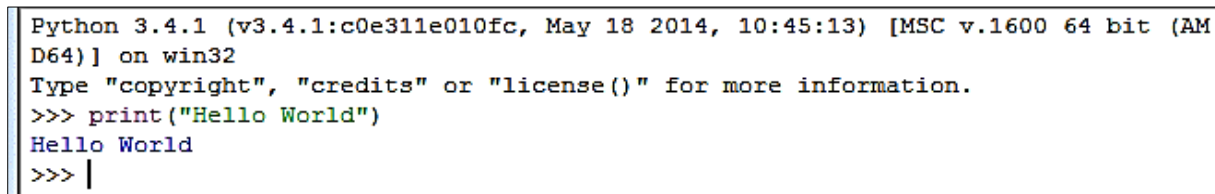
A screenshot of the Python IDLE interactive shell window. The window has a menu bar with 'File', 'Edit', 'Shell', 'Debug', 'Options', 'Windows', and 'Help'. The main text area shows the Python version and build information: 'Python 3.4.1 (v3.4.1:c0e311e010fc, May 18 2014, 10:45:13) [MSC v.1600 64 bit (AMD64)] on win32'. Below this, it says 'Type "copyright", "credits" or "license()" for more information.' and the prompt '>>>' followed by a cursor.

Writing your first Python program

If you type in:

print("Hello World")

The print statement is used to let the program know that you want to display whatever is inside the brackets. Python will read your program and display the output:

A screenshot of the Python IDLE interactive shell window showing the execution of a print statement. The window has the same menu bar as the previous screenshot. The main text area shows the Python version and build information, followed by the prompt '>>> print("Hello World")'. Below this, the output 'Hello World' is displayed, and the prompt '>>>' followed by a cursor is shown on the next line.

Activity 1.1

Now write a program that will display your name.

Getting Python to do calculations

Python can also be used to carry out calculations. Write each of the examples into Python and enter the output into the table below. The symbols used are called **operators**.

Name	Symbol in Python	Example	Output (Display)
Addition	+	2+2	4
Subtraction	-	5-2	
Multiplication	*	3*3	
Division	/	20/4	
Remainder	%	20%6	
Exponent	**	2**3	
Division (no remainder)	//	8//5	

The output is shown below:

```
Python 3.4.1 (v3.4.1:c0e311e010fc, May 18 2014, 10:45:13) [MSC v.1600 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> 2+2
4
>>> 5-2
3
>>> 3*3
9
>>> 20/4
5.0
>>> 20%6
2
>>> 2**3
8
```

When using Python you need to know the order in which the calculations take place. For this you must use **BODMAS**

Brackets - Start with calculating anything inside the brackets (from left to right).

Order - Do anything involving a power or a square root next (from left to right).

Division/Multiplication -multiplication and division rank equally, so you go from left to right in the sum.

Addition/Subtraction-subtraction and addition rank equally, and so you go from left to right in the sum.

Activity 1.2		
Now carry out the following calculations using your brain and using Python:	Brain Output	Python Output
50/10+3		
(7-2)*8		
7**2+(1+7-8)		
8/2*8/8+(100-90)		

The importance of using the correct syntax

Like any language, including English, It is important that you are careful as you enter code into Python. You must use the correct syntax. If you make a mistake you will get a syntax error and the program will not work. As you develop your programming skills you must be careful to follow the rules/syntax used in Python.

For example whenever you write text (in Python this is known as a string) you must always use speech marks or you will get a syntax error.

```
>>> print(Hello World)
SyntaxError: invalid syntax
>>> |
```

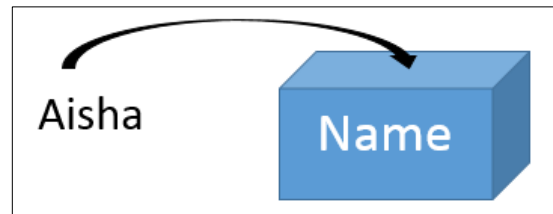
2. The use of variables, constants, operators, inputs, outputs and assignments

What is a variable?

Variables are used in computer programs to store a piece of information.

For example, if you create a computer program that asks for a person's name. A variable can be used to store the person's name.

A variable is like a box that can only store one piece of information at a time.



Using the Scripting Window

Our programs are going to get a bit more complex and so we are now going to use the scripting window. Open the Python Interactive (shell) window and then select **New** from the **File** menu.

This will open the scripting window. When you use the scripting window you must save the program before you can run it.

Write this program using the scripting window:

File	Edit	Format	Run	Options	Windows	Help
------	------	--------	-----	---------	---------	------

```
name=input("What is your name?: ")
print("Hi",name,", I'm pleased to meet you.")
```

This program uses a **variable** called name.

input is used to tell the computer that you are expecting the person using the program to enter something, and in this case you are expecting them to enter a name.

Once the person has entered a name, the **variable** is **assigned** this value.

print is used to **output** information, which in this case is a mixture of text strings and the value assigned to the variable.

Now make sure you have saved the program and then run it (select **Run module** from the **Run** menu)

When you **output** the program it will ask you to enter a name. Once you have added a name and clicked enter, the sentence will be displayed.

```
>>>
What is your name?: Aisha
Hi Aisha , I'm pleased to meet you.
>>>
```

Activity 2.1

(a) Write a program that asks a person their first name, their surname and the name of their favourite animal.

The program then displays a sentence that welcomes them and says who they are and what their favourite animal is.

(b) How many variables did you use in the program?

Constants

Sometimes the data item being stored will not change i.e. it will **not** be a variable. These items of data are called **constants**.

E.g. the program below uses Pi to calculate the circumference of a circle. Pi is a constant and will always be 3.14.

The formula is $\text{circumference} = 2 \times \pi \times \text{radius}$ is used to calculate the circumference.

```
#Calculate the circumference of a circle using the constant Pi
pi=3.14
radius=2
print("The circumference is: ",2*pi*radius)
```

The screenshot shows a Python 3.4.1 Shell window. The title bar says "Python 3.4.1 Shell". The menu bar includes "File", "Edit", "Shell", "Debug", "Options", "Windows", and "Help". The main text area shows the following output:

```
Python 3.4.1 (v3.4.1:c0e311e010fc, May 18 2014, 10:45:13) [MSC v
D64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> ===== RESTART =====
>>>
The circumference is: 12.56
```

Activity 2.2

Write a program to calculate the area of a circle using the constant pi and the variables radius and area.

The formula is **area=pi*radius**2** or **(a=πr²)**

The program displays a message: The area is

Choosing an appropriate variable name

Python syntax requires that the variable name takes a particular format.

- No spaces e.g. cannot be **first name**
- Must not start with a number e.g. cannot be **5first_name**

Tips for using appropriate variable names

- Use an underscore to link words e.g. **first_name**
- Use a name that makes sense e.g. don't use data1 for first name use **first_name**
- Name all variables in a consist way e.g. **first_name**, **telephone_number** etc
- Always start with a **letter**
- Remember that variable names are case sensitive **first_name** is not the same as **first_Name**
- Don't name variables using words that are reserved keywords in Python. You can type `help("keywords")` in the interactive window (shell) to get a list of reserved words.

```
>>> help("keywords")

Here is a list of the Python keywords.  Enter any keyword to get more help.

False          def            if             raise
None           del            import         return
True           elif          in             try
and            else          is             while
as             except        lambda         with
assert         finally      nonlocal       yield
break          for           not
class          from          or
continue       global       pass
```

Activity 2.3

Using an example for each explain the meaning of the following terms:
Variables, constants, operators, inputs, outputs and assignments

Assigning multiple values

You can assign more than one variable in a line of code. The code from the previous example has been rewritten so that there are only two lines of code.

```
pi, radius=3.14, 2
print("The circumference is: ", 2*pi*radius)
```

Activity 2.4

Which variable names will cause a syntax error?
Surname; postcode; 5x; x5; GreenflyNumbers; uSERnAME; global

3. The use of data types

Python will use **data types** to decide how to handle particular type of data. Each variable will be one of four data types when using Python.

Data Type	Name in Python	When is it used
String	str	This is used for variables that store text. To identify a string in python you put the text inside speech marks (can be double speech marks or single speech marks). E.g. "Hello World"
Character	str	This is used for variables that store one single character. E.g. "a"
Integer	int	This is used for variables that store a positive or negative whole number. As it is only storing a whole number it uses up less memory than if it was to store a decimal number. E.g. 32
Real	float	This is used for variables that store a decimal number. When you use Python you must use the word float for this data type. E.g. 32.91819
Boolean	bool	This is used when there are only two possible values. This could be True or False . E.g. this is used when a program needs to find out whether something is right or wrong.

Take a look at these two programs. They are very similar but produce a different output. As you can see from these programs it is very important that you choose the correct data type or the program may not produce the outcome you would like.

Example 1

```
File Edit Form
x="2"
y="2"
print(x+y)

Python 3.4.1 Shell
File Edit Shell
Python 3.4.1
D64) ] on win
Type "copyri
>>> =====
>>>
22
```

Example 1
uses a
string data
type.

Example 2

```
File Edit Fo
x=2
y=2
print(x+y)

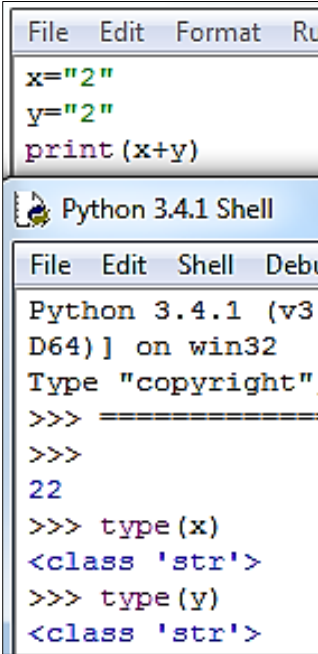
Python 3.4.1 Shell
File Edit Sh
Python 3.4
D64) ] on w
Type "copy
>>> =====
>>>
4
```

Example 2
uses an
integer
data type.

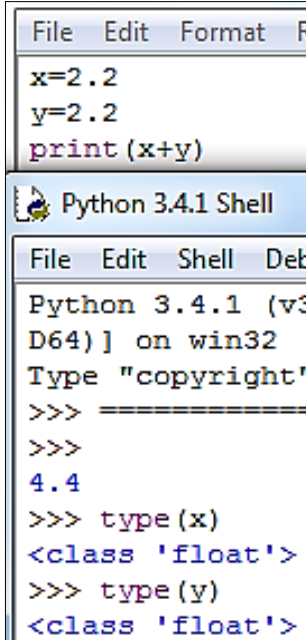
How to check the data type of a variable

You can find out the data type of a variable by using `type(x)` where the variable is called x. These three examples show how the data type is displayed.

Example 1
Example 3



Example 2



Activity 3.1

Create the three programs above, run them and find out the **data type** for each variable.

Casting

Sometimes when you write a program you need to be able to change a variable that has one data type to another data type. This is called **casting**.

In the example below both x and y start off with the data type string, but then they are **cast** into a different data type integer.

```
x="2"
y="2"
x=int(x)
y=int(y)
print(x+y)
```

Data type: **str**

Casting used to change data type to **int**

How to check the data type of a variable

In the example below, the data type was changed using casting from a **string** data type to an **integer** data type.

```
x="2"
y="2"
x=int(x)
y=int(y)
print(x+y)
```

We can make sure that the data type is correct by running the program and entering:

type(variable name)

In this case you would enter **type(x)** as the variable name is **x** and **type(y)** as the other variable name is **y**.

```
Python 3.4.1 (v3.4.1:
D64)] on win32
Type "copyright",
>>> =====
>>>
4
>>> type(x)
<class 'int'>
>>> type(y)
<class 'int'>
```

Activity 3.2

Write a program to calculate the circumference of a circle, using the constant **pi** and the variable **radius**.

Start by:

- assigning the value 3.14 to pi using the data type string (**str**)
- assigning the value 4 to radius using the data type string (**str**)

Then:

- use casting to change the data type for pi from string to **float** (Float is the data type used for decimal numbers)
- use casting to change the data type for radius from string to **integer** (Integer is the data type used for whole numbers)

The formula is **circumference=2*pi*radius** or **(c=2πr)**

The program displays a message: The circumference is

4. The use of basic string manipulation

Should I use double quotation marks or single quotation marks?

Using double or single quotation marks generally produces the same output.

```
>>> print('Welcome to Python')
Welcome to Python
>>> print("Welcome to Python")
Welcome to Python
>>> |
```

How do I display text over a number of lines?

You can use triple quote marks to display text over a number of lines.

```
>>> print("""
Welcome to the
world of programming
using Python""")

Welcome to the
world of programming
using Python
```

Sometimes you may want to use quote marks in a string. You can use single quotation marks for this.

```
>>>
>>> print('Is this really "Python"?')
Is this really "Python"?
>>> |
```

Using Escape Characters

Sometimes you need to perform an action inside the string. To do this you use escape characters.

When using single quotation marks you can use `\\` to show a backslash in your output.

```
>>> print('C:\\Drive')
C:\\Drive
```

When using single quote marks you can use `\'` to show an apostrophe in your output.

```
>>> print('It was Dave\'s program')
It was Dave's program
```

When you want to display text on a new line in your program.

```
>>> print("This is the first line\nThis is the second line")
This is the first line
This is the second line
```

When you want to indent or tab your text.

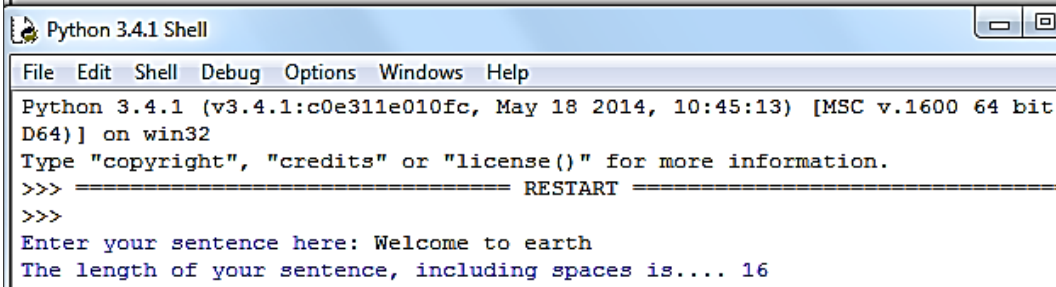
```
>>> print("Item 1\tItem 2\tItem 3")
Item 1  Item 2  Item 3
```

Handling strings in a program

When you write a program you will often need to manipulate strings.

Find the length of a string using - `len(variable_name)`

```
sentence=input("Enter your sentence here: ")
print("The length of your sentence, including spaces is....",len(sentence))
```



Python 3.4.1 Shell

File Edit Shell Debug Options Windows Help

Python 3.4.1 (v3.4.1:c0e311e010fc, May 18 2014, 10:45:13) [MSC v.1600 64 bit D64] on win32

Type "copyright", "credits" or "license()" for more information.

```
>>> ===== RESTART =====
>>>
Enter your sentence here: Welcome to earth
The length of your sentence, including spaces is.... 16
```

Change the string to UPPER case using `variable_name.upper()`

```

File Edit Format Run Options Windows Help
sentence=input("Enter your sentence here: ")
print(sentence.upper())

Python 3.4.1 Shell
File Edit Shell Debug Options Windows Help
Python 3.4.1 (v3.4.1:c0e311e010fc, May 18 2014, D64) on win32
Type "copyright", "credits" or "license()" for more
>>> ===== RESTART =====
>>>
Enter your sentence here: Welcome to Earth
WELCOME TO EARTH

```

Change the string to lower case using `variable_name.lower()`

```

sentence=input("Enter your sentence here: ")
print(sentence.lower())

Python 3.4.1 Shell
File Edit Shell Debug Options Windows Help
Python 3.4.1 (v3.4.1:c0e311e010fc, May 18 2014, D64) on win32
Type "copyright", "credits" or "license()" for more
>>> ===== RESTART =====
>>>
Enter your sentence here: Welcome to Earth
welcome to earth
>>>

```

Change the string to capitalise the first letter using `variable_name.capitalize()`

```

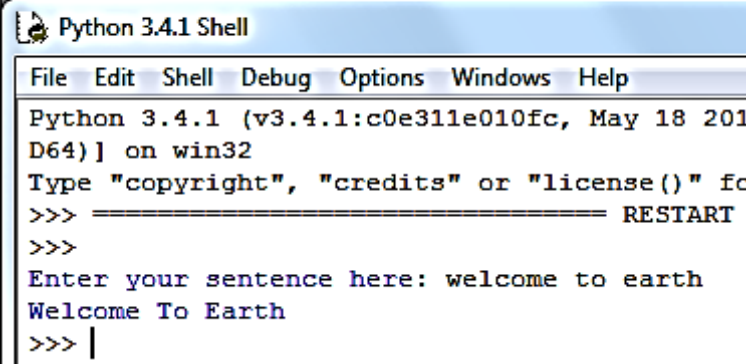
sentence=input("Enter your sentence here: ")
print(sentence.capitalize())

Python 3.4.1 Shell
File Edit Shell Debug Options Windows Help
Python 3.4.1 (v3.4.1:c0e311e010fc, May 18 2014, D64) on win32
Type "copyright", "credits" or "license()" for more
>>> ===== RESTART =====
>>>
Enter your sentence here: welcome to earth
Welcome to earth
>>> |

```

Change the string to capitalise the first letter of each word using `variable_name.title()`

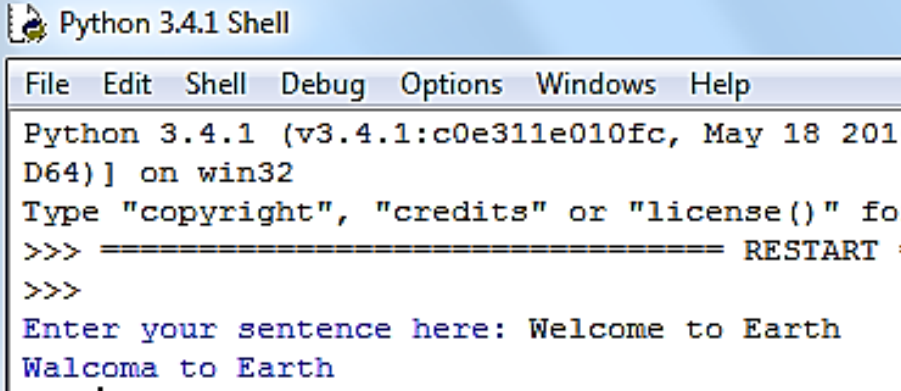
```
sentence=input("Enter your sentence here: ")
print(sentence.title())
```



The screenshot shows the Python 3.4.1 Shell interface. The user has entered the code to take an input sentence and print it with the first letter of each word capitalized. The prompt 'Enter your sentence here: ' is shown, followed by the user input 'welcome to earth'. The output is 'Welcome To Earth'.

Replace a letter with another letter using `variable_name.replace(x,y)`

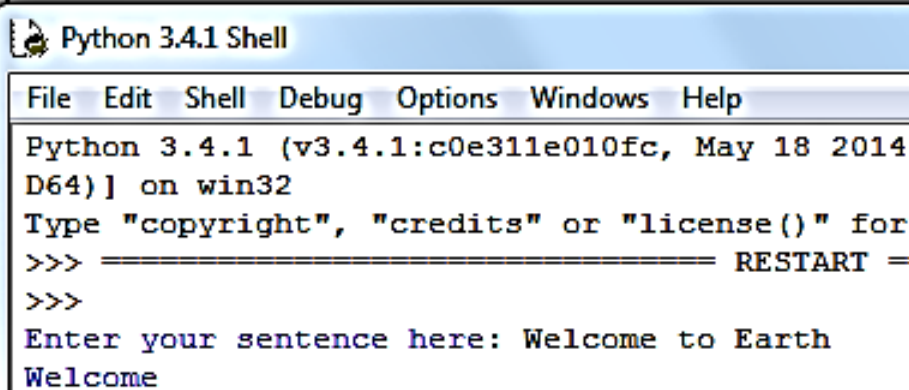
```
sentence=input("Enter your sentence here: ")
print(sentence.replace("e","a"))
```



The screenshot shows the Python 3.4.1 Shell interface. The user has entered the code to take an input sentence and print it with all 'e's replaced by 'a's. The prompt 'Enter your sentence here: ' is shown, followed by the user input 'Welcome to Earth'. The output is 'Walcoma to Earth'.

Return a sub-string of the original string using `variable_name[x:y]`

```
sentence=input("Enter your sentence here: ")
print(sentence[0:7])
```



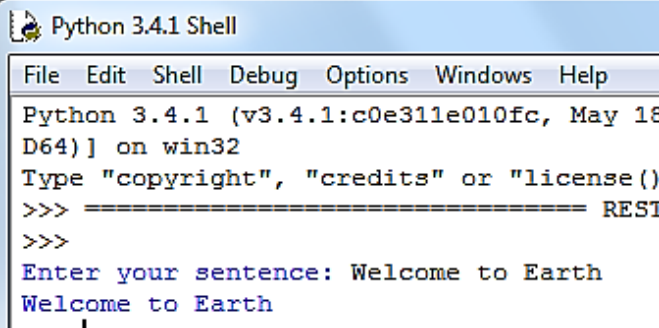
The screenshot shows the Python 3.4.1 Shell interface. The user has entered the code to take an input sentence and print the first 7 characters. The prompt 'Enter your sentence here: ' is shown, followed by the user input 'Welcome to Earth'. The output is 'Welcome'.

String Alignment

You sometimes need to format your string. You might want the text on the left, on the right or in the centre.

Making text appear on the left using <

```
sentence=input("Enter your sentence: ")
print('{:<30}'.format(sentence))
```



Python 3.4.1 Shell

File Edit Shell Debug Options Windows Help

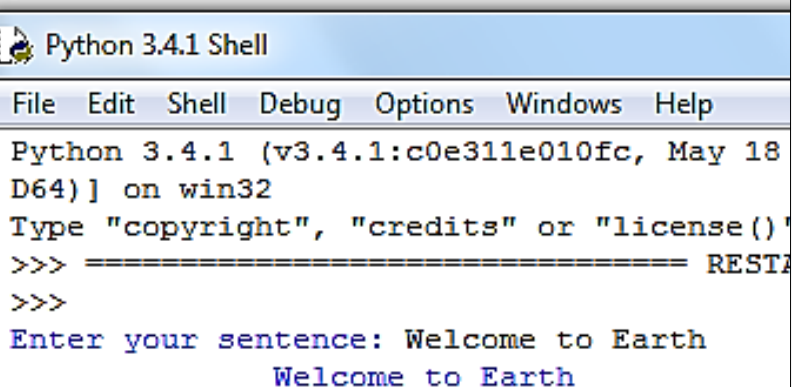
Python 3.4.1 (v3.4.1:c0e311e010fc, May 18 2014) on win32

Type "copyright", "credits" or "license()" for more

```
>>> ===== RESTART >>>
>>>
Enter your sentence: Welcome to Earth
Welcome to Earth
```

Making text appear on the right using >

```
sentence=input("Enter your sentence: ")
print('{:>30}'.format(sentence))
```



Python 3.4.1 Shell

File Edit Shell Debug Options Windows Help

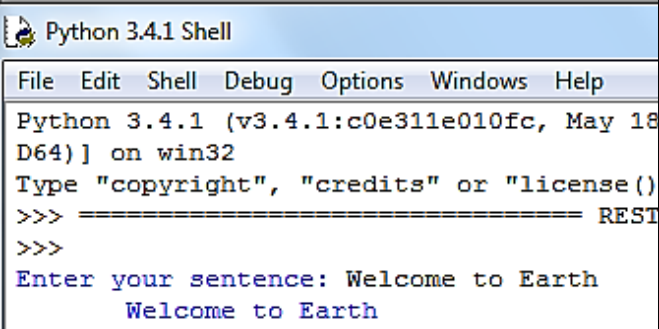
Python 3.4.1 (v3.4.1:c0e311e010fc, May 18 2014) on win32

Type "copyright", "credits" or "license()" for more

```
>>> ===== RESTART >>>
>>>
Enter your sentence: Welcome to Earth
Welcome to Earth
```

Making text appear in the centre using ^

```
sentence=input("Enter your sentence: ")
print('{:^30}'.format(sentence))
```



Python 3.4.1 Shell

File Edit Shell Debug Options Windows Help

Python 3.4.1 (v3.4.1:c0e311e010fc, May 18 2014) on win32

Type "copyright", "credits" or "license()" for more

```
>>> ===== RESTART >>>
>>>
Enter your sentence: Welcome to Earth
Welcome to Earth
```


Adding asterisks around a centred title

```
sentence=input("Enter your sentence: ")
print('{:~^30}'.format(sentence))
```

Python 3.4.1 Shell

File Edit Shell Debug Options Windows Help

Python 3.4.1 (v3.4.1:c0e311e010fc, May 1 D64) on win32

Type "copyright", "credits" or "license(>>> ===== RES

>>>

Enter your sentence: Welcome to Earth

*****Welcome to Earth*****

Using formatting to create a table of results

You can simply print a set of data without using formatting, but it does not look very professional and is harder to read (left) or you can use formatting (right)

```
c1="Dacio Sandero"
c2="Suzuki Celerio"
c3="Skoda Citigo"
c4="Peugeot 108"
p1=7395
p2=7999
p3=8995
p4=9349
print("Car prices")
print(c1,p1)
print(c2,p2)
print(c3,p3)
print(c4,p4)
```

Python 3.4.1 Shell

File Edit Shell Debug Opt

Python 3.4.1 (v3.4.1:c0e311e010fc, May 18 2014, 10 D64) on win32

Type "copyright", "cre

>>> =====

>>>

Car prices

Dacio Sandero 7395

Suzuki Celerio 7999

Skoda Citigo 8995

Peugeot 108 9349

```
c1="Dacio Sandero"
c2="Suzuki Celerio"
c3="Skoda Citigo"
c4="Peugeot 108"
p1=7395
p2=7999
p3=8995
p4=9349
print("Car prices")
print("{0:<15}{1:<4}".format("Car Type", "Price"))
print("{0:<15}{1:<4}".format(c1,p1))
print("{0:<15}{1:<4}".format(c2,p2))
print("{0:<15}{1:<4}".format(c3,p3))
print("{0:<15}{1:<4}".format(c4,p4))
```

Python 3.4.1 Shell

File Edit Shell Debug Options Windows Help

Python 3.4.1 (v3.4.1:c0e311e010fc, May 18 2014, 10 D64) on win32

Type "copyright", "credits" or "license()" for mor

>>> ===== RESTART =====

>>>

Car prices

Car Type	Price
Dacio Sandero	7395
Suzuki Celerio	7999
Skoda Citigo	8995
Peugeot 108	9349

Activity 4.1

the cow jumped over the moon

1. Using Python change the string above to display using:

- A. All words in upper case
- B. Capitalise the first word in the sentence
- C. Capitalise all words in the sentence

2. Create a table using Python to display the following results:

Pets - How long do they live?

Dogs 13 years Cats 15 years Gold Fish 8 years Hamster 3 years rabbit 7 years