Lesson 1 – Console, Variables and Creating a Program

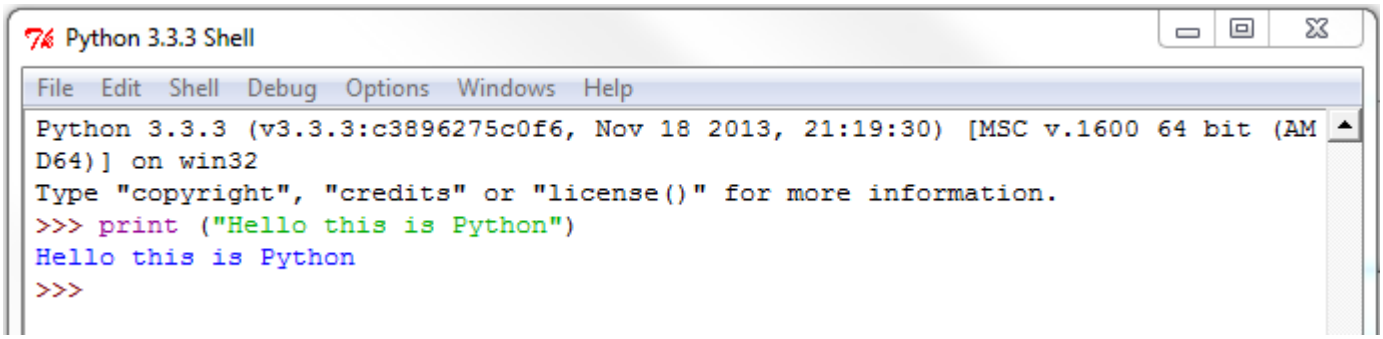
Lesson Outcomes

In this lesson you will learn:

- how to use the Python Console;
- the use of variables and basic arithmetic operations;
- about different data types;
- how to get input from the user;
- how to create your first program.

The Console & Basic operations

In Python the console is the command line interface where are all programs are executed. It is referred to as the SHELL and is used to either run programs or commands. So for example, the most common command is "print" which you can type in the shell as follows:



NOTE: Python is case sensitive e.g. type **print** as opposed to **Print**

The command **print** is followed by an opening bracket and then the text you want to display enclosed in quotes – you may use single or double quotes. You then finish the statement with a closed bracket. As well as print we may use a whole array of different expressions and commands.

Task 1.1 – Basic Commands

Try typing the following commands and basic expressions (press return after each command):

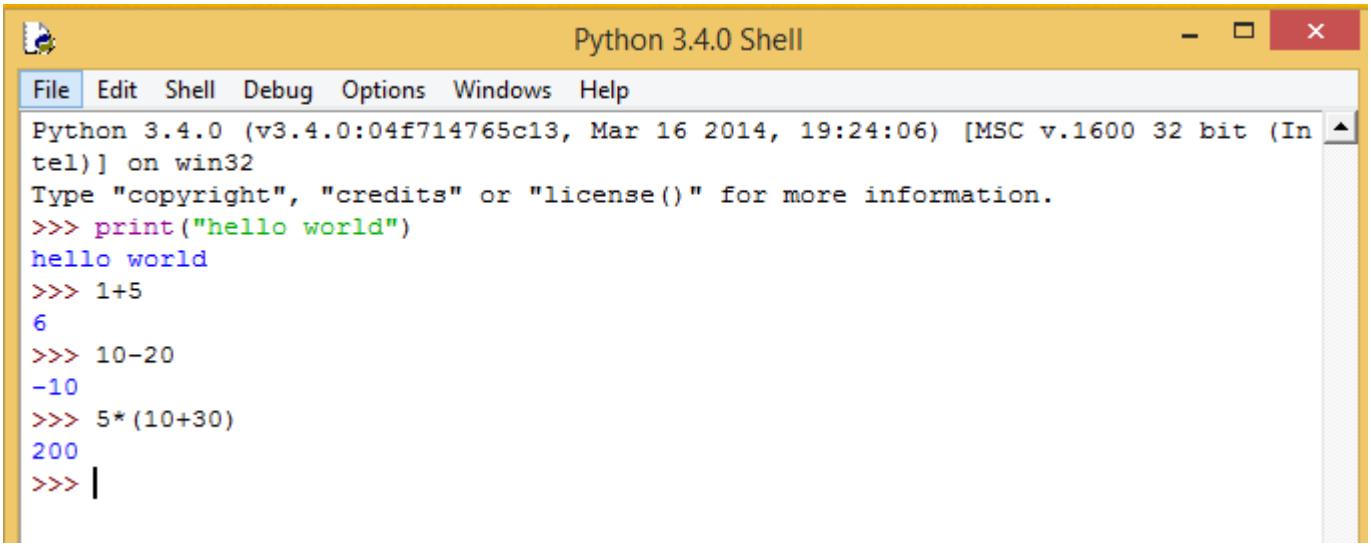
```
>>> print ('Hello world')

>>> 1 + 5

>>> 10 -20

>>> 5 * (10  + 30)
```

After each entry you should get the following:



The result of each command is displayed on the line below. You can use this shell to type in any command or expression and see the results. Try experimenting with your own expressions.

Variables

Variables are a temporary storage of data. You can think of variables as similar to those used in algebra

E.g. x = 10, y=20; x * y =200

In programming variables can not only store numbers but a whole range of different data types. They are used extensively in programs to store data in memory while operations are performed upon them. Again, we can define variables within the console.

Task 1.2 – Creating Variables

Try typing the following commands (press return after each command):

```
>>> x = (10 +2)

>>> y = x * 2

>>> x

>>> y
```

You should get the following results:

```
>>> x=(10+2)
>>> y= x*2
>>> x
12
>>> y
24
>>>
```

In the above exercise you assign the results of calculations to different variables. In assigning a number to a variable the **interpreter** (Python) tells the computer that this variable is storing values of TYPE **integer** e.g. whole number. Obviously, if you were to type something other than a number, for example text, it would store it as a different type e.g. for text it would be a **STRING.**

Task 1.3

Try the following:

```
>>> x = “My name is John”
>>> a = 10
>>> b = 10.5
>>> y = 19090902
>>> z = [‘John’, ‘89’, ‘David’, 29]
```

You can check the content of these variables by simple typing the variable name and pressing return:

```
>>> x
'My name is John'
>>> a
10
>>> b
10.5
>>> y
19090902
>>> z
['john', '89', 'David', 29]
>>>
```

Each one of the above variables have been allocated a **DATA TYPE.**

Variable	Content	Data Type
x	“My name is Mr Bradshaw”	This is what we call a **STRING**
a	10	This is a whole number referred to as an **INTEGER**
b	10.5	This is a decimal number and is what we call a **FLOAT**.
y	28989374878782323	This is a very big number therefore we store it as type **LONG**
z	[‘John’, 89, ‘David’, 29]	This is a list of values and in Python is stored as type **LIST** (we will learn about these later on in the course)

Converting Variable Types

Task 1.4

When we create variables Python assigns a type to that variable. So, as discussed before, if you assign a whole number to a variable then it becomes an INTEGER data type. In Python, and indeed in many languages, you can't mix data types unless you tell the interpreter to convert from one to another. So, for example, if you try to add a STRING "Hello my name is John" to the number 9 (INTEGER) then Python will give you an error – as you would expect.

Try the following:

```
>>> x = 39033300000 (big number)

>>> y = 10

>>> X + Y

>>> x + "Hello"
```

The last statement should create an ERROR. The reason for this is an **INTEGER** data type is being added to I noa **STRING data type.** Python doesn't allow you to mix or add completely different **DATA TYPES** together without some conversion first.

To help us to mix and match different **DATA TYPES** we can use **functions.** A **function** is defined as a command which takes a **PARAMETER** (a user defined value) placed n brackets:

Function	Description
int ("10")	converts the TEXT "10" into a number
str (1)	converts the number 1 to a string i.e. text
float(10)	converts the integer 10 to a decimal
float("10.4")	converts the string "10.4" to a decimal
int (10.6)	converts the decimal 10.6 to an integer e.g. the number 10 (truncates rather than rounds)

So, for example, if we want to add an **INTEGER** to a **STRING** we have to type the following

TASK 1.5

Type the following commands:

```
>>> print ("10 + 20 = " + str(10+20))

>>> int (10.8)

>>> float ("10.46")
```

You should get:

```
>>> print ("10+20 = " + str(10+20))
10+20 = 30
>>> int(10.8)
10
>>> float("10.46")
10.46
>>>
```

In the first print command we convert the result of the expression 10+20 into a STRING to display. If you didn't have this conversion Python would display an error – you are trying to bring together to different data types. The second truncates the number 10.8 to the integer 10, and the final command converts the string "10.46" into a floating point decimal number.

Task 1.6 - Corrections

CORRECT the following statements (they have deliberate errors in them):

```
>>> 10 + "20"

>>> "One plus twenty two equals " + (1 + 22)

>>> 20.0  + 40

>>> "£s and pence: " + (20 + 5.56) + "left"

>>>  int(20 + 5.56)

>>> int("25.56")
```

Writing your first program

From the IDE , Select **File → New File**

Task 1.6 – Creating program

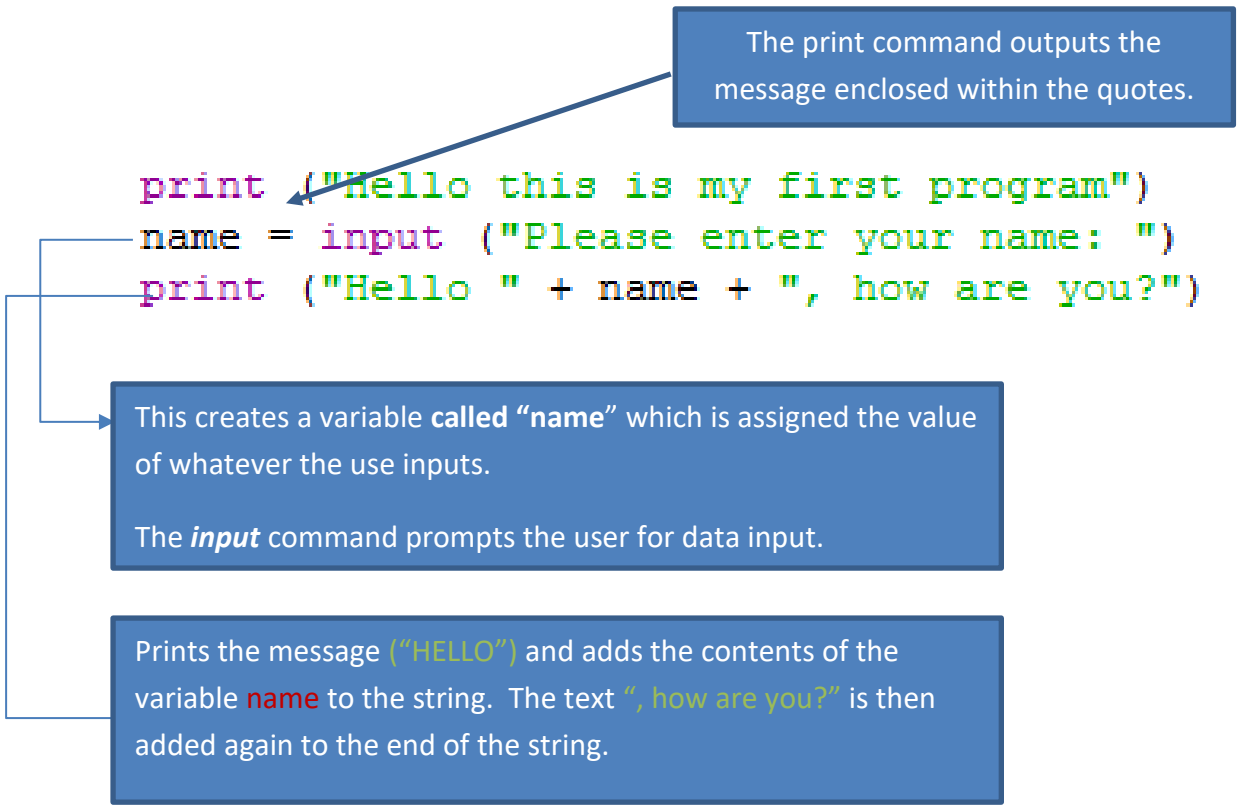
Enter the following code in the new window:

```
File  Edit  Format  Run  Options  Windows  Help
print ("Hello this is my first program")
name = input("Please enter your name: ")
print ("Hello" + name + ", how are you?")
```

Save and **Run** the program.

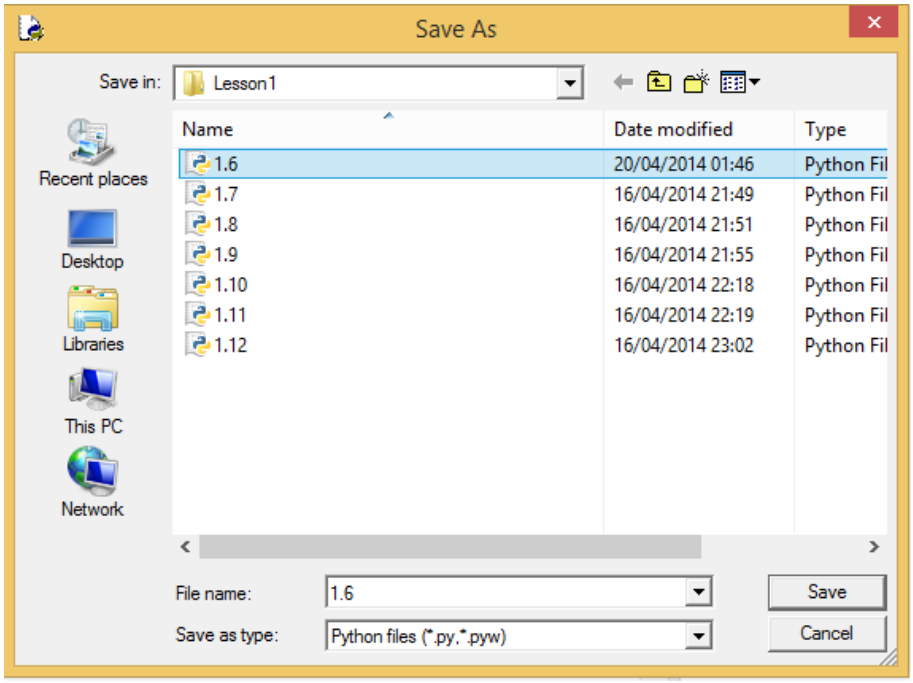
This program should display the HELLO message and then prompt you for your name. Once you have entered your name the program should display your entry in a "HELLO" message.

Let's look at this program again, and see what it does:



Saving

When you have written your program save in an appropriate folder, making sure it has the extension .py :



Examples of programs

The INT function converts text to a number

Inputting numbers.

```
number1 = int (input("Please enter number 1"))
number2 = int (input("Please enter number 2"))
```

The above program asks the user for an input and then converts the TEXT input into an INTEGER number for the variable.

Programming Tasks

1.7 Write a program to add two numbers together and display the result.

1.8 Write a program which asks for their surname and first name and displays your complete name in one line of text.

1.9 Write a program which prompts the user for 3 numbers and then displays the sum of those numbers.

HINT: Remember to convert your input into an INTEGER.

1.10 Enter the length, width and depth of a rectangular swimming pool. Calculate the volume of water required to fill the pool and display the volume.

Good practice – naming variables

You have already started to use lots of variables in your programs, with each one given a different name e.g. number1 or x or y. In programming it is considered good practice to make sure that variables are named sensibly according to purpose and the nature of data stored. For example, in Task 1.7 you may of used a variable called "x" for the first number and "y" for the second. Although this will work it doesn't tell you much about what the variable is and what it stored. This becomes very important when creating fairly large programs with lots of variables. Trying to debug a program where you don't know what the variable is what it does can be extremely frustrating! To resolve this problem we simply try to give variables sensible names. So, in Task 1.7, you could name your variables ***number1*** and ***number2,*** and then for the total store the result in a variable called ***Total.***

Arithmetic operations

So far you have written basic programs to take input from a user and perform basic arithmetic operations. Python allows many different types of arithmetic operations:

Arithmetic Operator	Operation	Example
+	Addition	X + Y
-	Subtraction	Result – 1
*	Multiplication	P * Interest Rate
/	Division	X / 2
//	Integer Division	X // 2
%	Remainder	Y % 2
**	Performs power	X**Y will give X to the power of Y

Extension Tasks

1.11 x / y calculates how many times y divides into x. If x and y are integers then only a whole number is displayed. The "%" gives the remainder. Using these operators to write a program that will read in two integers, `number1` and `number2`, and display the whole number part and the remainder of dividing `number1` by `number2`.

1.12 Write a program to enter an amount of money as a whole number, for example £78, and display the minimum number of £20, £10, £5 notes and £2 and £1 coins that make up this amount.

For example, the value £78 would give 3 twenty pound notes, 1 ten pound note, 1 five pound note, 1 two pound coin and 1 one pound coin.