Chapter 1

Getting Started

After studying this lesson, students will be able to:

- *Appreciate the use of Graphical User interface and Integrated Development Environment for creating Python programs.*
- *Work in interactive & Script mode for programming.*
- *Create and assign values to variables.*
- *Understand the concept and usage of different data types in python.*
- *Appreciate the importance and usage of different types of operator (arithmetic, Relation and logical)*
- *Create Python expression(s) and statement(s).*

Introduction

In order to tell the computer 'what you want to do', we write a program in a language which computer can understand. Though there are many different programming languages such as BASIC, Pascal, C, C++, Java, Haskell, Ruby, Python, etc. but we will study Python in this course.

Before learning the technicalities of Python, let's get familiar with it.

Python was created by Guido Van Rossum when he was working at CWI (Centrum Wiskunde & Informatica) which is a National Research Institute for Mathematics and Computer Science in Netherlands. The language was released in I991. Python got its name from a BBC comedy series from seventies- "Monty Python's Flying Circus". Python can be used to follow both Procedural approach and Object Oriented approach of programming. It is free to use.

Some of the features which make Python so popular are as follows:

- It is a general purpose programming language which can be used for both scientific and non scientific programming.
- It is a platform independent programming language.

- It is a very simple high level language with vast library of add-on modules.
- It is excellent for beginners as the language is interpreted, hence gives immediate results.
- The programs written in Python are easily readable and understandable.
- It is suitable as an extension language for customizable applications.
- It is easy to learn and use.

The language is used by companies in real revenue generating products, such as:

- In operations of Google search engine, youtube, etc.
- Bit Torrent peer to peer file sharing is written using Python
- Intel, Cisco, HP, IBM, etc use Python for hardware testing.
- Maya provides a Python scripting API
- i–Robot uses Python to develop commercial Robot.
- NASA and others use Python for their scientific programming task.

First Step with Python

We are continuously saying that Python is a programming language but don't know what a program is? Therefore, let's start Python by understanding Program.

A program is a sequence of instructions that specifies how to perform a Computation. The Computation might be mathematical or working with text.

To write and run Python program, we need to have Python interpreter installed in our computer. IDLE (GUI integrated) is the standard, most popular Python development environment. IDLE is an acronym of Integrated Development Environment. It lets edit, run, browse and debug Python Programs from a single interface. This environment makes it easy to write programs.

> We will be using version 2.7 of Python IDLE to develop and run Python code, in this course. It can be downloaded from www.python.org

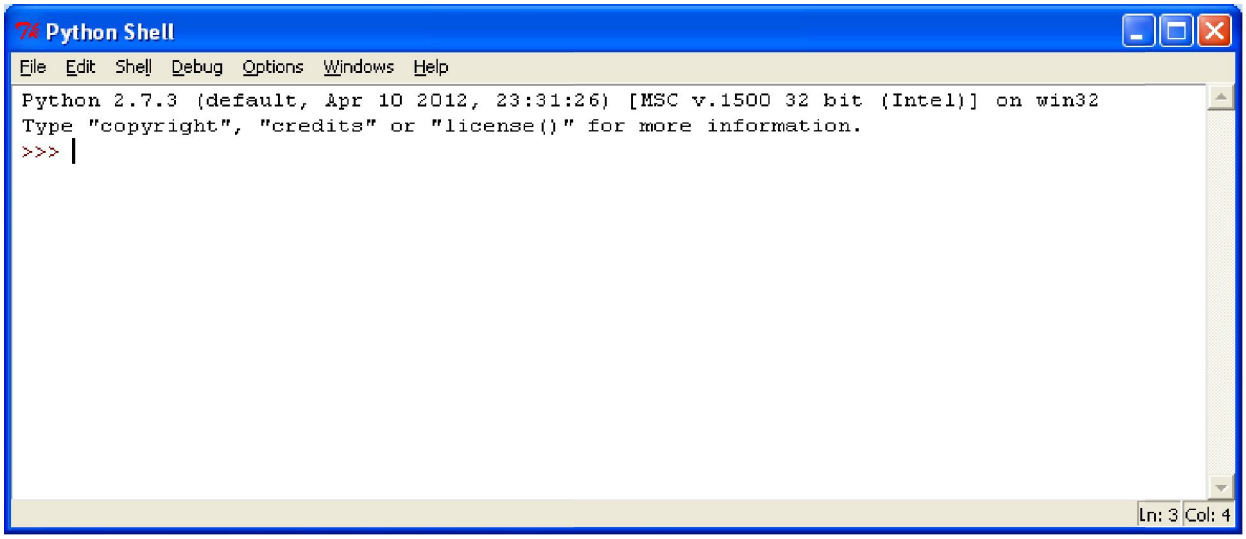
Python shell can be used in two ways, viz., interactive mode and script mode. Where Interactive Mode, as the name suggests, allows us to interact with OS; script mode let us

create and edit python source file. Now, we will first start with interactive mode. Here, we type a Python statement and the interpreter displays the result(s) immediately.

Interactive Mode

For working in the interactive mode, we will start Python on our computer. You can take the help of your Teacher.

When we start up the IDLE following window will appear:



What we see is a welcome message of Python interpreter with revision details and the Python prompt, i.e., '>>>'. This is a primary prompt indicating that the interpreter is expecting a python command. There is secondary prompt also which is '…' indicating that interpreter is waiting for additional input to complete the current statement.

Interpreter uses prompt to indicate that it is ready for instruction. Therefore, we can say, if there is prompt on screen, it means IDLE is working in interactive mode.

We type Python expression / statement / command after the prompt and Python immediately responds with the output of it. Let's start with typing print **"How are you"** after the prompt.

>>>print "How are you?"

How are you?

What we get is Python's response. We may try the following and check the response:

i) print 5+7

ii) 5+7

iii) 6*250/9

iv) print 5-7

It is also possible to get a sequence of instructions executed through interpreter.

Example 1	Example 2

>>> x=2
>>> y=6
>>> z = x+y
>>> print z
8
	>>> a=3
	>>> a+1, a-1
	(4,2) #result is tuple of 2 values

#result is tuple of 2 values, is a comment statement. We will talk about it in the later part of chapter.

Now we are good to write a small code on our own in Python. While writing in Python, remember Python is case sensitive. That means x & X are different in Python.

Note: If we want to repeat prior command in interactive window, you can use '↑' key to scroll backward through commands history and '↓' key to scroll forward. Use Enter key to select it. Using these keys, your prior commands will be recalled and displayed, and we may edit or rerun them also.

^D (Ctrl+D) or quit () is used to leave the interpreter.

^F6 will restart the shell.

Help of IDLE can be explored to know about the various menu options available for Programmer.

Apart from writing simple commands, let's explore the interpreter more.

Type **Credits** after the prompt and what we get is information about the organization involved in Python development. Similarly, **Copyright** and **Licenses** command can be used to know more about Python. Help command provides **help** on Python. It can be used as….. help() with nothing in parenthesis will allow us to enter an interactive help mode. And with a name (predefined) in bracket will give us details of the referred word.

To leave the help mode and return back to interactive mode, quit command can be used.

Script Mode

In script mode, we type Python program in a file and then use the interpreter to execute the content from the file. Working in interactive mode is convenient for beginners and for testing small pieces of code, as we can test them immediately. But for coding more than few lines, we should always save our code so that we may modify and reuse the code.

> **Note:** Result produced by Interpreter in both the modes, viz., Interactive and script mode is exactly same.

Python, in interactive mode, is good enough to learn, experiment or explore, but its only drawback is that we cannot save the statements for further use and we have to retype all the statements to re-run them.

To create and run a Python script, we will use following steps in IDLE, if the script mode is not made available by default with IDLE environment.

1. File>Open OR File>New Window (for creating a new script file)
2. Write the Python code as function i.e. script
3. Save it (^S)
4. Execute it in interactive mode- by using RUN option (^F5)

   Otherwise (if script mode is available) start from Step 2

> **Note:** For every updation of script file, we need to repeat step 3 & step 4

If we write Example 1 in script mode, it will be written in the following way:

Step 1: File> New Window

Step 2:

```
def test():
x=2
y=6
z = x+y
print z
```

Step 3:

Use File > Save or File > Save As - option for saving the file

(*By convention all Python program files have names which end with .py*)

Step 4:

For execution, press ^F5, and we will go to Python prompt (in other window)

```
>>> test()
8
```

Alternatively we can execute the script directly by choosing the RUN option.

Note: While working in script mode, we add 'print' statement in our program to see the results which otherwise were displayed on screen in interactive mode without typing such statements.

Variables and Types

When we create a program, we often like to store values so that it can be used later. We use objects to capture data, which then can be manipulated by computer to provide information. By now we know that object / variable is a name which refers to a value.

Every object has:

A. An Identity, - can be known using id (object)

B. A type – can be checked using type (object) and

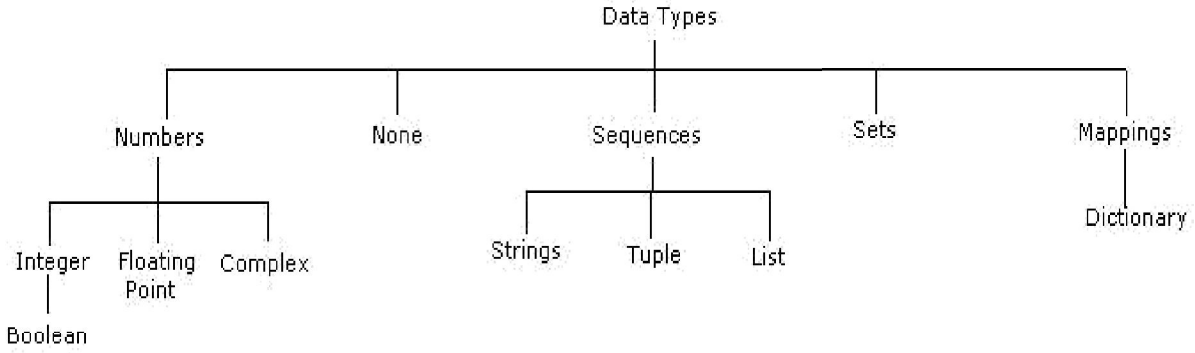
C. A value

Let us study all these in detail

A. Identity of the object: It is the object's address in memory and does not change once it has been created.

***(We would be referring to objects as variable for** now)*

B. Type (i.e data type): It is a set of values, and the allowable operations on those values. It can be one of the following:



1. Number

Number data type stores Numerical Values. This data type is immutable i.e. value of its object cannot be changed (we will talk about this aspect later). These are of three different types:

a) Integer & Long

b) Float / floating point

c) Complex

Range of an integer in Python can be from -2147483648 to 2147483647, and long integer has unlimited range subject to available memory.

1.1 Integers are the whole numbers consisting of + or – sign with decimal digits like 100000, -99, 0, 17. While writing a large integer value, don't use commas to separate digits. Also integers should not have leading zeros.

When we are working with integers, we need not to worry about the size of integer as a very big integer value is automatically handled by Python. When we want a value to be treated as very long integer value append **L** to the value. Such values are treated as long integers by python.

```
>>> a = 10
>>> b = 5192L          #example of supplying a very long value to a variable
>>> c= 4298114
>>> type(c)            # type ( ) is used to check data type of value
<type 'int'>
>>> c = c * 5669
>>> type(c)
<type 'long'>
```

We can know the largest integer in our version of Python by following the given set of commands:

```
>>> import sys
>>> print sys.maxint
```

Integers contain Boolean Type which is a unique data type, consisting of two constants, **True** & **False**. A Boolean True value is Non-Zero, Non-Null and Non-empty.

Example

```
>>> flag = True
>>> type(flag)
<type 'bool'>
```

1.2 **Floating Point:** Numbers with fractions or decimal point are called floating point numbers.

A floating point number will consist of sign (+,-) sequence of decimals digits and a dot such as 0.0, -21.9, 0.98333328, 15.2963. These numbers can also be used to represent a number in engineering / scientific notation.

$-2.0X\ 10^5$ will be represented as -2.0e5

$2.0X10^{-5}$ will be 2.0E-5

Example

y= 12.36

A value when stored as floating point in Python will have 53 bits of precision.

1.3 **Complex:** Complex number in python is made up of two floating point values, one each for real and imaginary part. For accessing different parts of variable (object) x; we will use x.real and x.image. Imaginary part of the number is represented by 'j' instead of 'i', so 1+0j denotes zero imaginary part.

Example

```
>>> x = 1+0j
>>> print x.real,x.imag
1.0 0.0
```

Example

```
>>> y = 9-5j
>>> print y.real, y.imag
9.0 -5.0
```

2. None

This is special data type with single value. It is used to signify the absence of value / false in a situation. It is represented by **None**.

3. Sequence

A sequence is an ordered collection of items, indexed by positive integers. It is combination of mutable and non mutable data types. Three types of sequence data type available in Python are Strings, Lists & Tuples.

3.1 **String:** is an ordered sequence of letters/characters. They are enclosed in single quotes (' ') or double (" "). The quotes are not part of string. They only tell the computer where the string constant begins and ends. They can have any character or sign, including space in them. These are immutable data types. We will learn about immutable data types while dealing with third aspect of object i.e. value of object.

Example

```
>>> a = 'Ram'
```

A string with length 1 represents a character in Python.

Conversion from one type to another

If we are not sure, what is the data type of a value, Python interpreter can tell us:

```
>>> type ('Good Morning')
<type 'str'>
>>> type ('3.2')
<type 'str'>
```

It is possible to change one type of value/ variable to another type. It is known as type conversion or type casting. The conversion can be done explicitly (programmer specifies the conversions) or implicitly (Interpreter automatically converts the data type).

For explicit type casting, we use functions (constructors):

```
int ()
float ()
str ()
bool ()
```

Example

```
>>> a= 12.34
>>> b= int(a)
>>> print b
12
```

Example

```
>>>a=25
>>>y=float(a)
>>>print y
25.0
```

3.2 **Lists:** List is also a sequence of values of any type. Values in the list are called elements / items. These are mutable and indexed / ordered. List is enclosed in square brackets.

Example

l = ['spam', 20.5,5]

3.3 **Tuples:** Tuples are a sequence of values of any type, and are indexed by integers. They are immutable. Tuples are enclosed in (). We have already seen a tuple, in Example 2 (4, 2).

4. Sets

Set is an unordered collection of values, of any type, with no duplicate entry. Sets are immutable.

Example

```
s = set ([1,2,34])
```

5. Mapping

This data type is unordered and mutable. Dictionaries fall under Mappings.

5.1 **Dictionaries:** Can store any number of python objects. What they store is a key – value pairs, which are accessed using key. Dictionary is enclosed in curly brackets.

Example

```
d = {1:'a',2:'b',3:'c'}
```

C. Value of Object (variable) – to bind value to a variable, we use assignment operator (=). This is also known as building of a variable.

Example

```
>>> pi = 31415
```

Here, value on RHS of '=' is assigned to newly created 'pi' variable.

Mutable and Immutable Variables

A mutable variable is one whose value may change in place, whereas in an immutable variable change of value will not happen in place. Modifying an immutable variable will rebuild the same variable.

Example

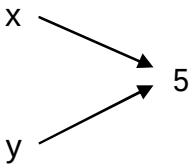
```
>>>x=5
```

Will create a value 5 referenced by x

x → 5

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

This statement will make y refer to 5 of x




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

As x being integer (immutable type) has been rebuild.

In the statement, expression on RHS will result into value 10 and when this is assigned to LHS (x), x will rebuild to 10. So now

x ⟶ 10 and

y ⟶ 5

After learning about what a variable can incorporate, let's move on with naming them. Programmers choose the names of the variable that are meaningful. A variable name:

1. Can be of any size
2. Have allowed characters, which are a-z, A-Z, 0-9 and underscore (_)
3. should begin with an alphabet or underscore
4. should not be a keyword

It is a good practice to follow these identifier naming conventions:

1. Variable name should be meaningful and short
2. Generally, they are written in lower case letters

Keywords

They are the words used by Python interpreter to recognize the structure of program. As these words have specific meaning for interpreter, they cannot be used for any other purpose.

A partial list of keywords in Python 2.7 is

and	del	from	not
while	as	elif	global
or	with	assert	else
if	pass	yield	break
except	import	print	class
exec	in	raise	continue
finally	is	return	def
for	lambda	try	

Remember:

- ✡ Variables are created when they are first assigned a value.
- ✡ Variables must be assigned a value before using them in expression,
- ✡ Variables refer to an object and are never declared ahead of time.

Operators and Operands

Operators are special symbols which represents computation. They are applied on operand(s), which can be values or variables. Same operator can behave differently on different data types. Operators when applied on operands form an expression. Operators are categorized as Arithmetic, Relational, Logical and Assignment. Value and variables when used with operator are known as operands.

Following is the partial list of operators:

Mathematical/Arithmetic Operators

Symbol	Description	Example 1	Example 2
+	Addition	>>>55+45 100	>>> 'Good' + 'Morning' GoodMorning
-	Subtraction	>>>55-45 10	>>>30-80 -50
*	Multiplication	>>>55*45 2475	>>> 'Good'* 3 GoodGoodGood
/	Division	>>>17/5 3 >>>17/5.0 3.4 >>> 17.0/5 3.4	>>>28/3 9

%	Remainder / Modulo	>>>17%5 2	>>> 23%2 1
**	Exponentiation	>>>2**3 8 >>>16**.5 4.0	>>>2**8 256
/ /	Integer Division	>>>7.0/ /2 3.0	>>>3/ / 2 1

Note: Division is Implementation Dependent

Relational Operators

Symbol	Description	Example 1	Example 2

<	Less than	>>>7<10
		True
		>>> 7<5
		False
		>>> 7<10<15
		True
		>>>7<10 and 10<15
		True
			>>>'Hello'< 'Goodbye'
			False
			>>>'Goodbye'< 'Hello'
			True

>	Greater than	>>>7>5 True >>>10<10 False	>>>'Hello'> 'Goodbye' True >>>'Goodbye'> 'Hello' False
<=	less than equal to	>>> 2<=5	>>>'Hello'<= 'Goodbye'

		True >>> 7<=4 False	False >>>'Goodbye' <= 'Hello' True
>=	greater than equal to	>>>10>=10 True >>>10>=12 False	>>>'Hello'>= 'Goodbye' True >>>'Goodbye' >= 'Hello' False
! =, <>	not equal to	>>>10!=11 True >>>10!=10 False	>>>'Hello'!= 'HELLO' True >>> 'Hello' != 'Hello' False
==	equal to	>>>10==10 True >>>10==11 False	>>>'Hello' == 'Hello' True >>>'Hello' == 'Good Bye' False

Note: Two values that are of different data type will never be equal to each other.

Logical Operators

Symbol	Description
or	If any one of the operand is true, then the condition becomes true.
and	If both the operands are true, then the condition becomes true.
not	Reverses the state of operand / condition.

Assignment Operators

Assignment Operator combines the effect of arithmetic and assignment operator

Symbol	Description	Example	Explanation
=	Assigned values from right side operands to left variable	>>>x=12* >>>y='greetings'	

*(*we will use it as initial value of x for following examples)*

+=	added and assign back the result to left operand	>>>x+=2	The operand / expression / constant written on RHS of operator is will change the value of x to 14
-=	subtracted and assign back the result to left operand	x-=2	x will become 10
*=	multiplied and assign back the result to left operand	x*=2	x will become 24
/=	divided and assign back the result to left operand	x/=2	x will become 6
%=	taken modulus using two operands and assign the result to left operand	x%=2	x will become 0
**=	performed exponential (power) calculation on operators and assign value to the left operand	x**=2	x will become 144
//=	performed floor division on operators and assign value to the left operand	x //= 2	x will become 6

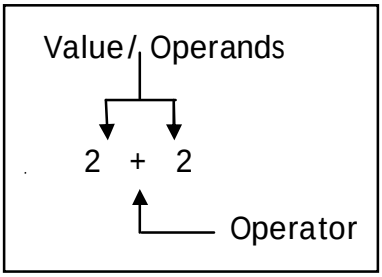
Note:

1. Same operator may perform a different function depending on the data type of the value to which it is applied.
2. Division operator '/' behaves differently on integer and float values.

Expression and Statements

An expression is a combination of value(s) (i.e. constant), variable and operators. It generates a single value, which by itself is an expression.

Example



The expression is solved by Computer and gets it value. In the above example, it will be 4, and we say the expression is evaluated.

Note: Expression values in turn can act as, Operands for Operators

We have seen many such expressions (with list of operator as example). 10+5 and 9+4+2 are two expressions which will result into value 15. Taking another example, 5.0/4+ (6-3.0) is an expression in which values of different data types are used. These type of expressions are also known as mixed type expressions.

When mixed type expressions are evaluated, Python promotes the result of lower data type to higher data type, i.e. to float in the above example. This is known as implicit type casting. So the result of above expression will be 4.25. Expression can also contain another expression. As we have already seen in 9+4+2. When we have an expression consisting of sub expression(s), how does Python decide the order of operations?

It is done based on precedence of operator. Higher precedence operator is worked on before lower precedence operator. Operator associativity determines the order of evaluation when they are of same precedence, and are not grouped by parenthesis. An operator may be Left-associative or Right –associative. In left associative, the operator falling on left side will be evaluated first, while in right assosiative operator falling on right will be evaluated first.

Note: In python '=' and '**' are Right Associative.

Precedence of operator - Listed from high precedence to low precedence.

Operator	Description
**	Exponentiation (raise to the power)
+ , -	unary plus and minus
* , / , %, / /	Multiply, divide, modulo and floor division
+ , -	Addition and subtraction
<, <=, >, >=	Comparison operators
==, !=	Equality operators
% =, / =, / / = , - =, + =, * =	Assignment operators
not and or	Logical operators

Using the above table, we know that 9+4 itself is an expression which evaluates to 13 and then 13+2 is evaluated to 15 by computer. Similarly, 5.0/4 + (6-3.0) will be evaluated as 5.0/4+3.0 and then to 1.25 + 3.0, and then 4.25.

If we just type 10+, we will get an error message. This happens because 10+ is not a complete expression. A complete expression will always have appropriate number of value (Operands) with each operator. '+' needs two operands and we have given just one.

Note: Remember precedence of operators is applied to find out which sub expression should be evaluated first.

Expression can be combined together to form large expressions but no matter how big the expression is, it always evaluate into a single value.

A Python statement is a unit of code that the Python interpreter can execute.

Example of statement are:

```
>>> x=5
>>> area=x**2          #assignment statement
>>>print x             #print statement
5
>>>print area
25
>>> print x, area
5  25
```

Note: To print multiple items in same line, separate them with comma.

Statements normally go to the end of a line.

```
X= "good morning"          #comment
```

What we have seen as an example till now were simple statements, i.e. they do not contain a nested block. In Python, there are compound / group statements also. They are sometimes called nested block. Statement belonging to a block are indented (usually by 4 spaces). Leading whitespace at the beginning of logical line is used to determine the indentation level of line. That means statement(s) which go together must have same indentation level.

Example of Compound Statement

Example

```
if i<0:
    print "i is negative"
else:
    print "i is non-negative"
```

Example

```
if i>0:
    print "i is positive"
else:
    print "i is equal to 0"
```

While writing Python statements, keep the following points in mind:

1. Write one python statement per line (Physical Line). Although it is possible to write two statements in a line separated by semicolon.
2. Comment starts with '#' outside a quoted string and ends at the end of a line. Comments are not part of statement. They may occur on the line by themselves or at the end of the statement. They are not executed by interpreter.
3. For a long statement, spanning multiple physical lines, we can use '/' at the end of physical line to logically join it with next physical line. Use of the '/' for joining lines is not required with expression consists of ( ), [ ], { }
4. When entering statement(s) in interactive mode, an extra blank line is treated as the end of the indented block.
5. Indentation is used to represent the embedded statement(s) in a compound / Grouped statement. All statement(s) of a compound statement must be indented by a consistent no. of spaces (usually 4)
6. White space in the beginning of line is part of indentation, elsewhere it is not significant.

Note:

- ✡ Wrong indentation can give rise to syntax error(s).
- ✡ Most Python editor will automatically indent the statements.
- ✡ A physical line is what you see as a line when you write a program and a logical line is what Python sees as a single statement.

Input and Output

A Program needs to interact with end user to accomplish the desired task, this is done using Input-Output facility. Input means the data entered by the user (end user) of the program. While writing algorithm(s), getting input from user was represented by Take / Input. In python, we have raw-input() and input ( ) function available for Input.

raw_input()

Syntax of raw_input() is:

raw_input ([prompt])

Optional

If prompt is present, it is displayed on the monitor after which user can provide the data from keyboard. The function takes exactly what is typed from keyboard, convert it to string and then return it to the variable on LHS of '='.

Example (in interactive mode)

```
>>>x=raw_input ('Enter your name: ')
```

Enter your name: ABC

x is a variable which will get the string (ABC), typed by user during the execution of program. Typing of data for the raw_input function is terminated by 'enter' key.

We can use raw_input() to enter numeric data also. In that case we typecast, i.e., change the datatype using function, the string data accepted from user to appropriate Numeric type.

Example

y=int(raw_input("enter your roll no"))

enter your roll no. 5

will convert the accepted string i.e. 5 to integer before assigning it to 'y'.

input()

Syntax for input() is:

Input ([prompt])

Optional

If prompt is present, it is displayed on monitor, after which the user can provide data from keyboard. Input takes whatever is typed from the keyboard and evaluates it. As the input provided is evaluated, it expects valid python expression. If the input provided is not correct then either syntax error or exception is raised by python.

Example

x= input ('enter data:')

Enter data: 2+1/2.0

Will supply 2.5 to x

input ( ), is not so popular with python programmers as:

i) Exceptions are raised for non-well formed expressions.

ii) Sometimes well formed expression can wreak havoc.

Output is what program produces. In algorithm, it was represented by print. For output in Python we use print. We have already seen its usage in previous examples. Let's learn more about it.

Print Statement

Syntax:

print expression/constant/variable

Print evaluates the expression before printing it on the monitor. Print statement outputs an entire (complete) line and then goes to next line for subsequent output (s). To print more than one item on a single line, comma (,) may be used.

Example

```
>>> print "Hello"
Hello
>>> print 5.5
5.5
>>> print 4+6
10
```

Try this on the computer and evaluate the output generated

```
>>>print 3.14159* 7**2
>>>print "I", "am" + "class XI", "student"
>>>print "I'm",
>>>print "class XI student"
>>>print "I'm ", 16, "years old"
```

Comments

As the program gets bigger, it becomes difficult to read it, and to make out what it is doing by just looking at it. So it is good to add notes to the code, while writing it. These notes are known as comments. In Python, comment start with '#' symbol. Anything written after # in a line is ignored by interpreter, i.e. it will not have any effect on the program.

A comment can appear on a line by itself or they can also be at the end of line.

Example

```
# Calculating area of a square
>>> area = side **2
```

or

```
>>>area= side**2                #calculating area of a square
```

For adding multi-line comment in a program, we can:

i) Place '#' in front of each line, or

ii) Use triple quoted string. They will only work as comment, when they are not being used as docstring. (A docstring is the first thing in a class / function / module, and will be taken up in details when we study functions).

The comment line "#calculating area of a rectangle" can also be written as following using triple quote:

1. """ Calculating area of a rectangle """
2. """ Calculating area
   of a rectangle """

We should use as many useful comments as we can, to explain

*Any assumptions made

*important details or decisions made in the program. This will make program more readable. We already know the importance of comments (documented in the program).