Unit II

CHAPTER 5

PYTHON – VARIABLES AND OPERATORS

U86IT

Learning Objectives

After studying this lesson, students will be able to:

- Appreciate the use of Graphical User Interface (GUI) and Integrated Development Environment (IDE) 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 operators (Arithmetic, Relational and Logical)
- Creating Python expression (s) and statement (s).

5.1 Introduction

Python is a general purpose programming language created by Guido Van Rossum from 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 supports both Procedural and Object Oriented programming approaches.

5.2 Key features of Python

✓ 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.

✓ The programs written in Python are easily readable and understandable.

The version 3.x of Python **IDLE** (**I**ntegrated **D**evelopment **L**earning **E**nvironment) is used to develop and run Python code. It can be downloaded from the web resource **www.python.org.**

5.3 Programming in Python

In Python, programs can be written in two ways namely **Interactive mode** and **Script mode.** The Interactive mode allows us to write codes in Python command prompt (>>>) whereas in script mode programs can be written and stored as separate file with the extension **.py** and executed. Script mode is used to create and edit python source file.

5.3.1 Interactive mode Programming

In interactive mode Python code can be directly typed and the interpreter displays the result(s) immediately. The interactive mode can also be used as a **simple calculator.**

(i) Invoking Python IDLE

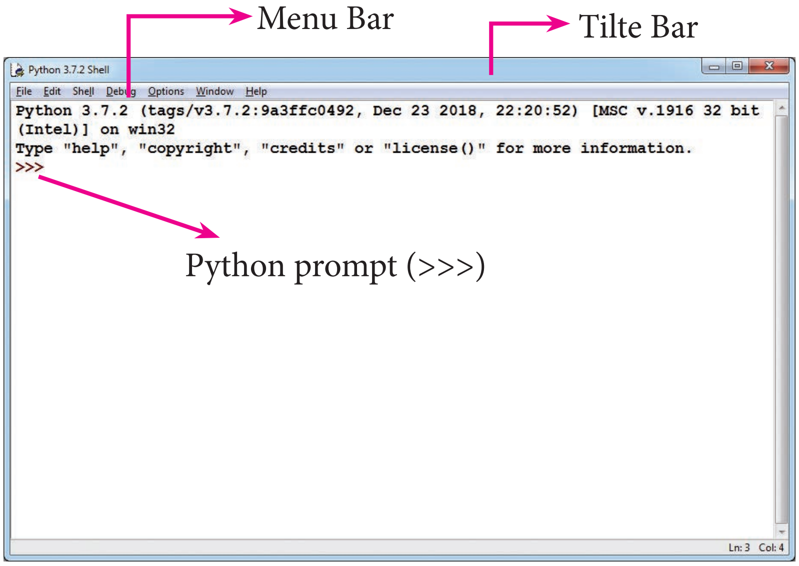
The following command can be used to invoke Python IDLE from Window OS.

Start → All Programs → Python 3.x → IDLE (Python 3.x)

(Or)

Click python Icon on the Desktop if available.

Now **Python IDLE** window appears as shown in the **Figure 5.1**



```
Python 3.7.2 (tags/v3.7.2:9a3ffc0492, Dec 23 2018, 22:20:52) [MSC v.1916 32 bit
(Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
```

Python IDLE Window

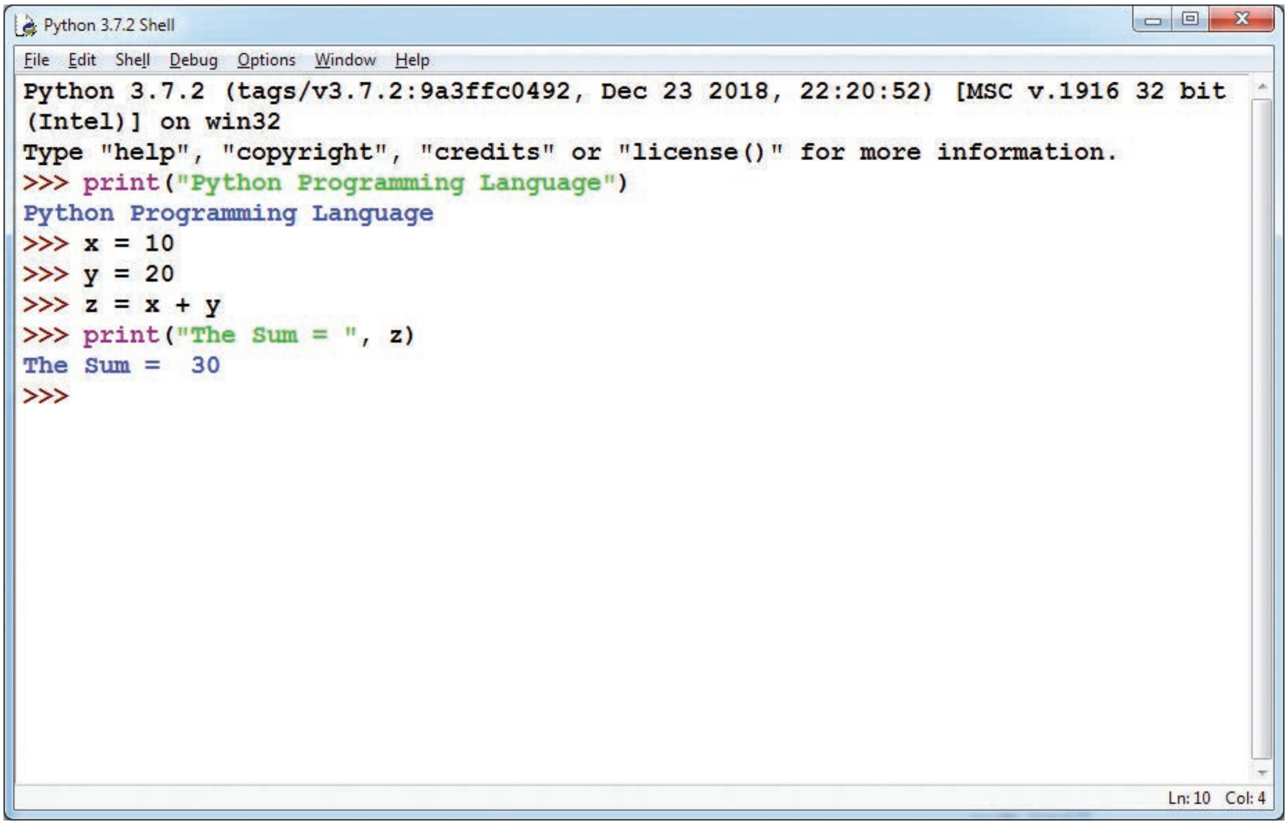
The prompt (>>>) indicates that Interpreter is ready to accept instructions. Therefore, the prompt on screen means **IDLE** is working in interactive mode. Now let us try as a simple calculator by using a simple mathematical expressions.

Example 1:

```
>>> 5 + 10
15
>>> 5 + 50 *10
505
>>> 5 ** 2
25
```

Example 2:

```
>>>print (“Python Programming Language”)
Python Programming Language
>>>x=10
>>>y=20
>>>z=x + y
>>>print (“The Sum”, z)
The Sum = 30
```



```
Python 3.7.2 (tags/v3.7.2:9a3ffc0492, Dec 23 2018, 22:20:52) [MSC v.1916 32 bit
(Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> print("Python Programming Language")
Python Programming Language
>>> x = 10
>>> y = 20
>>> z = x + y
>>> print("The Sum = ", z)
The Sum =  30
>>>
```

Python Interactive Window

5.3.2 Script mode Programming

Basically, a script is a text file containing the Python statements. Python Scripts are reusable code. Once the script is created, it can be executed again and again without retyping. The Scripts are editable.

(i) Creating Scripts in Python

1. Choose **File** → **New File** or press **Ctrl** + **N** in Python shell window.

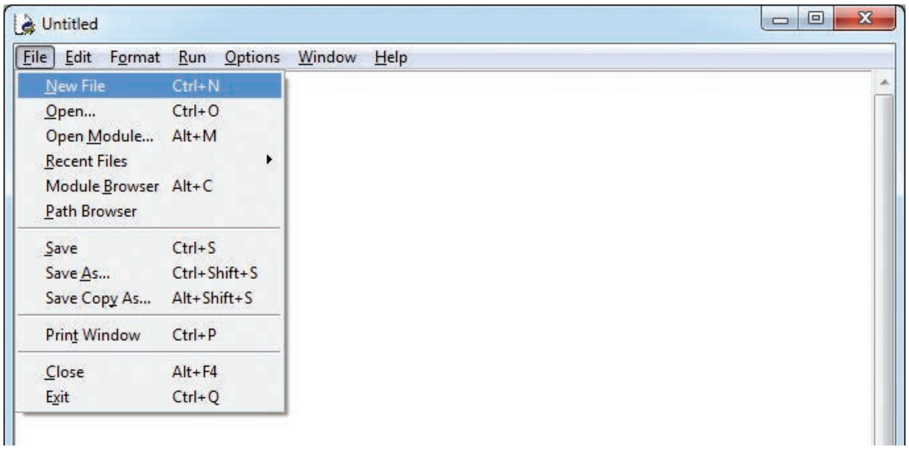


Figure 5.3 – To create new File

2. An **untitled** blank script text editor will be displayed on screen as shown in **Figure 5.3(a)**

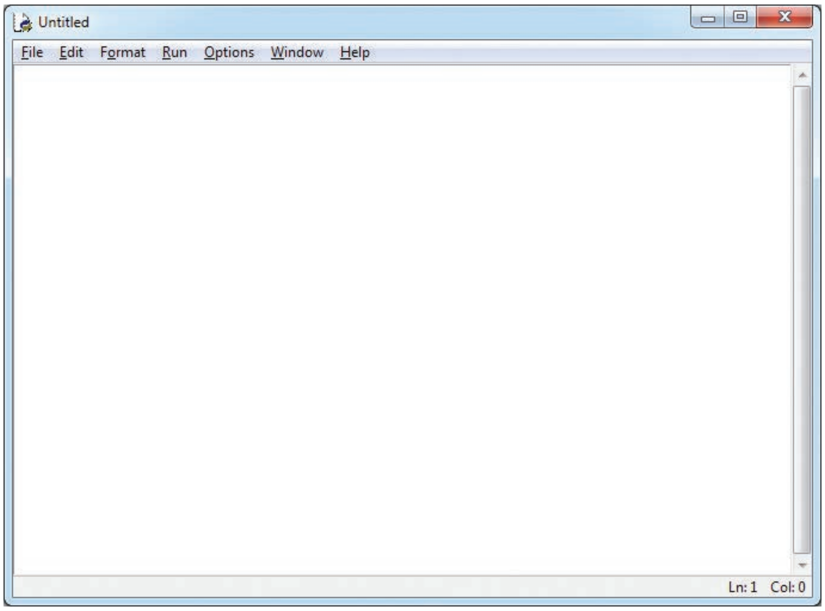


Figure 5.3(a) Untitled, blank Python script editor

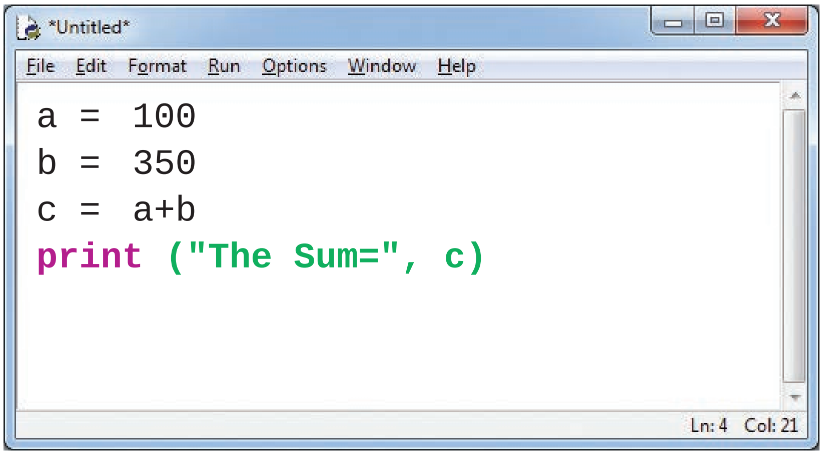
3. Type the following code in Script editor

```
a =100

b = 350

c = a+b

print ("The Sum=", c)
```



```
a = 100
b = 350
c = a+b
print ("The Sum=", c)
```

Figure 5.4 – Python Sample code

(ii) Saving Python Script

(1) Choose **File** → **Save** or Press **Ctrl + S**

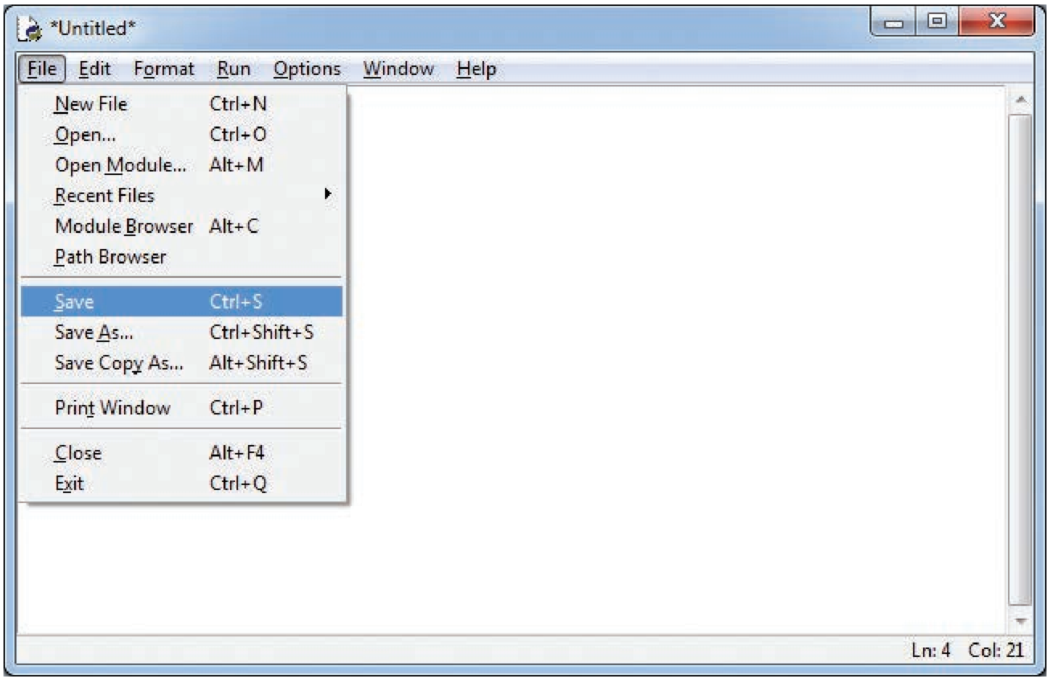
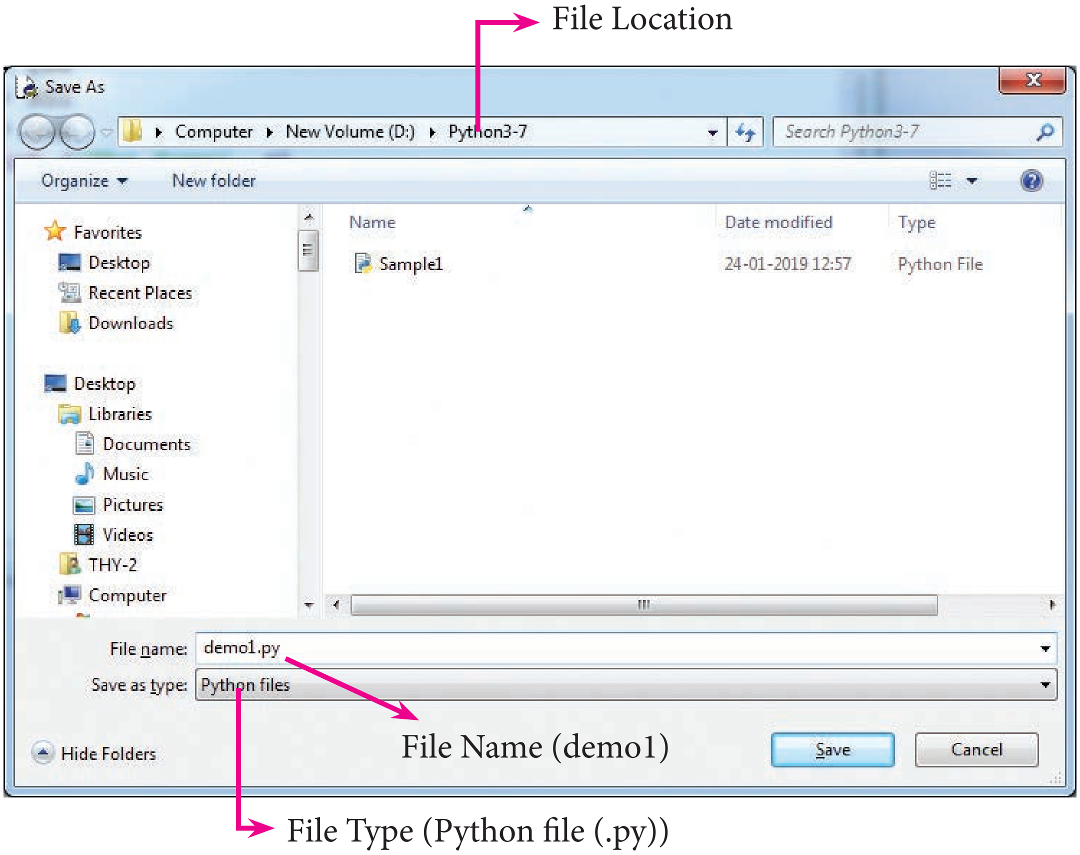


Figure 5.5 – To Save the file First time

(2) Now, **Save As** dialog box appears on the screen as shown in the **Figure 5.6**



File name: demo1.py

Save as type: Python files

File Location

File Name (demo1)

File Type (Python file (.py))

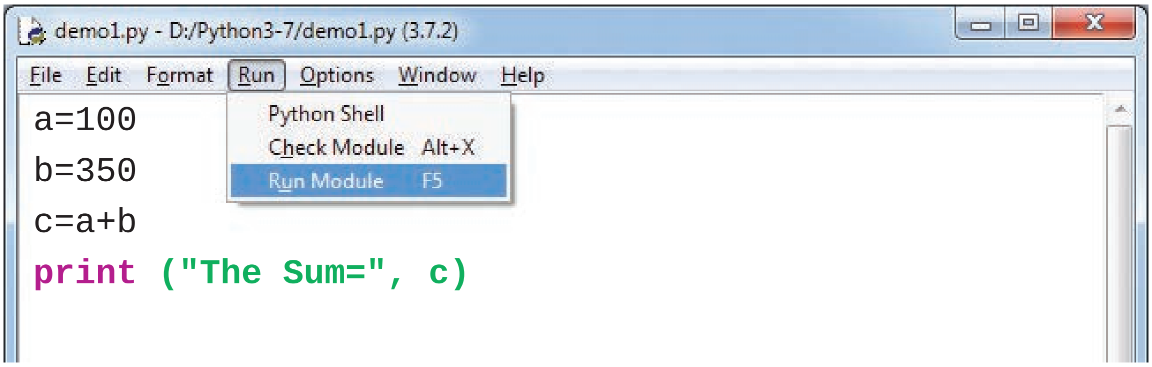
Figure 5.6 – Save As Dialog Box

(3) In the **Save As** dialog box, select the location where you want to save your Python code, and type the file name in **File Name** box. Python files are by default saved with extension **.py**. Thus, while creating Python scripts using Python Script editor, no need to specify the file extension.

(4) Finally, click **Save** button to save your Python script.

(iii) Executing Python Script

(1) Choose **Run** → **Run Module** or Press **F5**

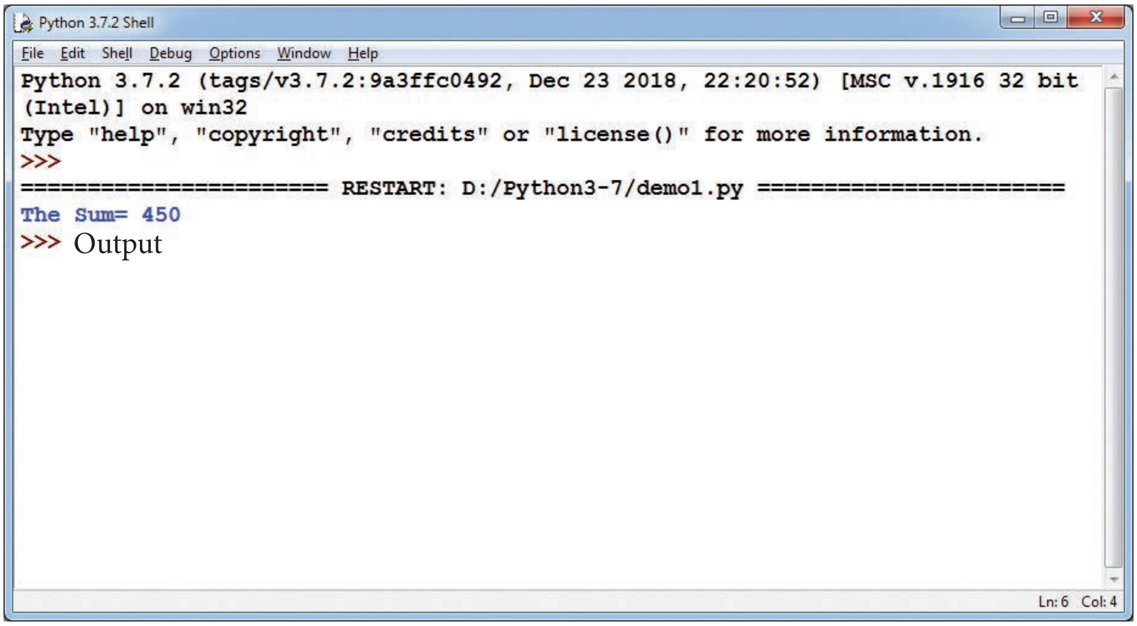

demo1.py - D:/Python3-7/demo1.py (3.7.2)
File Edit Format Run Options Window Help
Python Shell
Check Module Alt+X
Run Module F5

```
a=100
b=350
c=a+b
print ("The Sum=", c)
```

Figure 5.7 – To Execute Python Script

(2) If your code has any error, it will be shown in red color in the IDLE window, and Python describes the type of error occurred. To correct the errors, go back to Script editor, make corrections, save the file using **Ctrl + S** or **File** → **Save** and execute it again.

(3) For all error free code, the output will appear in the IDLE window of Python as shown in **Figure 5.8**



```
Python 3.7.2 (tags/v3.7.2:9a3ffc0492, Dec 23 2018, 22:20:52) [MSC v.1916 32 bit
(Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
======================= RESTART: D:/Python3-7/demo1.py =======================
The Sum= 450
>>> Output
```

Figure 5.8 –Python Script Output Window

5.4 Input and Output Functions

A program needs to interact with the user to accomplish the desired task; this can be achieved using **Input-Output functions**. The **input()** function helps to enter data at run time by the user and the output function **print()** is used to display the result of the program on the screen after execution.

5.4.1 The print() function

In Python, the **print()** function is used to display result on the screen. The syntax for **print()** is as follows:

Example

```
print (“string to be displayed as output ” )
print (variable )
print (“String to be displayed as output ”, variable)
print (“String1 ”, variable, “String 2”, variable, “String 3” ……)
```

Example

```
>>> print (“Welcome to Python Programming”)
        Welcome to Python Programming
>>> x = 5
>>> y = 6
>>> z = x + y
>>> print (z)
        11
>>> print (“The sum = ”, z)
        The sum = 11
>>> print (“The sum of ”, x, “ and ”, y, “ is ”, z)
        The sum of 5 and 6 is 11
```

The **print ( )** evaluates the expression before printing it on the monitor. The print () displays an entire statement which is specified within print ( ). **Comma (,)** is used as a separator in **print** ( ) to print more than one item.

5.4.2 input() function

In Python, **input( )** function is used to accept data as input at run time. The syntax for **input()** function is,

```
Variable = input (“prompt string”)
```

Where, **prompt string** in the syntax is a statement or message to the user, to know what input can be given.

If a prompt string is used, it is displayed on the monitor; the user can provide expected data from the input device. The **input( )** takes whatever is typed from the keyboard and stores the entered data in the given variable. If prompt string is not given in **input( )** no message is displayed on the screen, thus, the user will not know what is to be typed as input.

Example 1:input( ) with prompt string

```
>>> city=input (“Enter Your City: ”)
        Enter Your City: Madurai
>>> print (“I am from “, city)
        I am from Madurai
```

Example 2:input( ) without prompt string

```
>>> city=input()
        Rajarajan
>>> print (“I am from”, city)
        I am from Rajarajan
```

Note that in example-2, the **input( )** is not having any prompt string, thus the user will not know what is to be typed as input. If the user inputs irrelevant data as given in the above example, then the output will be unexpected. So, to make your program more interactive, provide prompt string with **input( ).**

The **input ( )** accepts all data as string or characters but not as numbers. If a numerical value is entered, the input values should be explicitly converted into numeric data type. The **int( )** function is used to convert string data as integer data explicitly. We will learn about more such functions in later chapters.

Example 3:

```
x = int (input(“Enter Number 1: ”))
y = int (input(“Enter Number 2: ”))
print (“The sum = ”, x+y)
```

Output:

```
        Enter Number 1: 34
        Enter Number 2: 56
        The sum = 90
```

Example 4: Alternate method for the above program

```
x,y=int (input("Enter Number 1 :")),int(input("Enter Number 2:"))
print ("X = ",x," Y = ",y)
```

Output:

```
        Enter Number 1 :30
        Enter Number 2:50
        X =  30  Y =  50
```

5.5 Comments in Python

In Python, comments begin with hash symbol **(#)**. The lines that begins with # are considered as comments and ignored by the Python interpreter. Comments may be single line or no multi-lines. The multiline comments should be enclosed within a set of ''' '''(triple quotes) as given below.

# It is Single line Comment

''' *It is multiline comment*

which contains more than one line '''

5.6 Indentation

Python uses whitespace such as **spaces** and **tabs** to define program blocks whereas other languages like C, C++, java use curly braces { } to indicate blocks of codes for class, functions or body of the loops and block of selection command. The number of whitespaces (spaces and tabs) in the indentation is not fixed, but all statements within the block must be indented with same amount spaces.

5.7 Tokens

Python breaks each logical line into a sequence of elementary lexical components known as **Tokens**. The normal token types are

1) Identifiers,
2) Keywords,
3) Operators,
4) Delimiters and
5) Literals.

Whitespace separation is necessary between tokens, identifiers or keywords.

5.7.1. Identifiers

An Identifier is a name used to identify a variable, function, class, module or object.

- An identifier must start with an alphabet (A..Z or a..z) or underscore (_).
- Identifiers may contain digits (0 .. 9)
- Python identifiers are case sensitive i.e. uppercase and lowercase letters are distinct.
- Identifiers must not be a **python** keyword.
- Python does not allow punctuation character such as %,$, @ etc., within identifiers.

Example of valid identifiers

Sum, total_marks, regno, num1

Example of invalid identifiers

12Name, name$, total-mark, continue

5.7.2. Keywords

Keywords are special 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.

Table 5.1 Python's Keywords

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

5.7.3 Operators

In computer programming languages operators are special symbols which represent computations, conditional matching etc. The value of an operator used is called **operands**. Operators are categorized as Arithmetic, Relational, Logical, Assignment etc. Value and variables when used with operator are known as **operands**.

(i) Arithmetic operators

An arithmetic operator is a mathematical operator that takes two operands and performs a calculation on them. They are used for simple arithmetic. Most computer languages contain a set of such operators that can be used within equations to perform different types of sequential calculations.

Python supports the following Arithmetic operators.

Operator - Operation	Examples	Result
Assume a=100 and b=10. Evaluate the following expressions		
+ (Addition)	>>> a + b	110
- (Subtraction)	>>>a – b	90
* (Multiplication)	>>> a*b	1000
/ (Divisioin)	>>> a / b	10.0
% (Modulus)	>>> a % 30	10
** (Exponent)	>>> a ** 2	10000
// (Floor Division)	>>> a//30 (Integer Division)	3

Program 5.1 To test Arithmetic Operators:

```
#Demo Program to test Arithmetic Operators
        a=100
        b=10
        print ("The Sum        = ",a+b)
        print ("The Difference = ",a-b)
        print ("The Product    = ",a*b)
        print ("The Quotient   = ",a/b)
        print ("The Remainder  = ",a%30)
        print ("The Exponent   = ",a**2)
        print ("The Floor Division =",a//30)
#Program End
```

Output:

```
        The Sum              =  110
        The Difference       =  90
        The Product          =  1000
        The Quotient         =  10.0
        The Remainder        =  10
        The Exponent         =  10000
        The Floor Division   = 3
```

(ii) Relational or Comparative operators

A Relational operator is also called as **Comparative** operator which checks the relationship between two operands. If the relation is true, it returns **True**; otherwise it returns **False**.

Python supports following relational operators

Operator - Operation	Examples	Result
Assume the value of a=100 and b=35. Evaluate the following expressions.		
== (is Equal)	>>> a==b	False
> (Greater than)	>>> a > b	True
< (Less than)	>>> a < b	False
>= (Greater than or Equal to)	>>> a >= b	True
<= (Less than or Equal to)	>>> a <= b	False
!= (Not equal to)	>>> a != b	True

Coding 5.2 To test Relational Operators:

```
#Demo Program to test Relational Operators
        a=int (input("Enter a Value for A:"))
        b=int (input("Enter a Value for B:"))
        print ("A = ",a," and B = ",b)
        print ("The a==b = ",a==b)
        print ("The a > b = ",a>b)
        print ("The a < b = ",a<b)
        print ("The a >= b = ",a>=b)
        print ("The a <= b = ",a<=b)
        print ("The a != b = ",a!=b)
#Program End
```

Output:

```
        Enter a Value for A:35
        Enter a Value for B:56
        A =  35  and B          = 56
        The a==b                = False
        The a > b               = False
        The a < b               = True
        The a >= b              = False
        The a <= b              = False
        The a != b              = True
```

(iii) Logical operators

In python, Logical operators are used to perform logical operations on the given relational expressions. There are three logical operators they are **and, or** and **not**.

Operator	Example	Result
Assume a = 97 and b = 35, Evaluate the following Logical expressions		
or	>>> a>b or a==b	True
and	>>> a>b and a==b	False
not	>>> not a>b	False i.e. Not True

Program 5.3 To test Logical Operators:

Example – Code

```
#Demo Program to test Logical Operators
a=int (input("Enter a Value for A:"))
b=int (input("Enter a Value for B:"))
print ("A = ",a, " and b = ",b)
print ("The a > b or a == b  = ",a>b or a==b)
print ("The a > b and a == b = ",a>b and a==b)
print ("The not a > b        = ",not a>b)
#Program End
```

Example - Result

```
Enter a Value for A:50
Enter a Value for B:40
A =  50  and b =  40
The a > b or a == b  =  True
The a > b and a == b =  False
The not a > b        =  False
```

(iv) Assignment operators

In Python, = is a simple assignment operator to assign values to variable. Let **a** = 5 and **b** = 10 assigns the value 5 to **a** and 10 to **b** these two assignment statement can also be given as **a,b=5,10** that assigns the value 5 and 10 on the right to the variables a and b respectively. There are various compound operators in Python like +=, -=, *=, /=, %=, **= and //= are also available.

Operator	Description	Example
Assume x=10		
=	Assigns right side operands to left variable	>>> x=10 >>> b="Computer"
+=	Added and assign back the result to left operand i.e. x=30	>>> x+=20 # x=x+20
-=	Subtracted and assign back the result to left operand i.e. x=25	>>> x-=5 # x=x-5
*=	Multiplied and assign back the result to left operand i.e. x=125	>>> x*=5 # x=x*5
/=	Divided and assign back the result to left operand i.e. x=62.5	>>> x/=2 # x=x/2

%=	Taken modulus(Remainder) using two operands and assign the result to left operand i.e. x=2.5	>>> x%=3 # x=x%3
**=	Performed exponential (power) calculation on operators and assign value to the left operand i.e. x=6.25	>>> x**=2 # x=x**2
//=	Performed floor division on operators and assign value to the left operand i.e. x=2.0	>>> x//=3

Program 5.4 To test Assignment Operators:

Program Coding

```
#Demo Program to test Assignment Operators
x=int (input("Type a Value for X : "))
print ("X = ",x)
print ("The x is       =",x)
x+=20
print ("The x += 20 is  =",x)
x-=5
print ("The x -= 5 is   = ",x)
x*=5
print ("The x *= 5 is   = ",x)
x/=2
print ("The x /= 2 is   = ",x)
x%=3
print ("The x %= 3 is   = ",x)
x**=2
print ("The x **= 2 is   = ",x)
x//=3
print ("The x //= 3 is   = ",x)
#Program End
```

Output

```
Type a Value for X : 10
X =  10
The x is       = 10
The x += 20 is  = 30
The x -= 5 is   =  25
The x *= 5 is   =  125
The x /= 2 is   =  62.5
The x %= 3 is   =  2.5
The x **= 2 is   =  6.25
The x //= 3 is   =  2.0
```

(v) Conditional operator

Ternary operator is also known as conditional operator that evaluate something based on a condition being true or false. It simply allows testing a condition in a single line replacing the multiline if-else making the code compact.

The Syntax conditional operator is,

Variable Name = [on_true] if [Test expression] else [on_false]

Example :

```
min= 49 if 49<50 else 50 # min = 49

min= 50 if 49>50 else 49 # min = 49
```

Program 5.5 To test Conditional (Ternary) Operator:

```
# Program to demonstrate conditional operator
        a, b = 30, 20
# Copy value of a in min if a < b else copy b
        min = a if a < b else b
        print ("The Minimum of A and B is ",min)
# End of the Program
```

Output:

```
        The Minimum of A and B is  20
```

5.7.4 Delimiters

Python uses the symbols and symbol combinations as delimiters in expressions, lists, dictionaries and strings. Following are the delimiters.

(	)	[	]	{	}
,	:	.	‘	=	;
+=	-=	*=	/=	//=	%=
&=	|=	^=	>>=	<<=	**=

5.7.5 Literals

Literal is a raw data given to a variable or constant. In Python, there are various types of literals.

1) Numeric
2) String
3) Boolean

(i) Numeric Literals

Numeric Literals consists of digits and are immutable (unchangeable). Numeric literals can belong to 3 different numerical types Integer, Float and Complex.

Program 5.6 : To demonstrate Numeric literals

```
# Program to demonstrate Numeric Literals
        a = 0b1010                  #Binary Literals
        b = 100                     #Decimal Literal
        c = 0o310                   #Octal Literal
        d = 0x12c                   #Hexadecimal Literal
        print ("Integer Literals :",a,b,c,d)
        #Float Literal
        float_1 = 10.5
        float_2 = 1.5e2
        print ("Float Literals :",float_1,float_2)
        #Complex Literal
        x = 1 + 3.14 j
        print ("Complex Literals :")
        Print ("x = ", x , "Imaginary part of x = ", x.imag, "Real part of x = ", x.real)
#End of the Program
```

Output:

```
        Integer Literals : 10 100 200 300
        Float Literals : 10.5 150.0
        Complex Literals :
        x = (1+3.14j) Imaginary part of x = 3.14 Real part of x = 1.0
```

(ii) String Literals

In Python a string literal is a sequence of characters surrounded by quotes. Python supports single, double and triple quotes for a string. A character literal is a single character surrounded by single or double quotes. The value with triple-quote ''' ''' is used to give multi-line string literal. A Character literal is also considered as string literal in Python.

Program 5.7 To test String Literals

```
# Demo Program to test String Literals
        strings = "This is Python"
        char = "C"
        multiline_str = '''This is a multiline string with more than one line code.'''
        print (strings)
        print (char)
        print (multiline_str)
# End of the Program
```

Output:

```
        This is Python
        C
        This is a multiline string with more than one line code.
```

(iii) Boolean Literals

A Boolean literal can have any of the two values: True or False.

Program 5.8 To test Boolean Literals:

```
# Demo Program to test String Literals
        boolean_1 = True
        boolean_2 = False
        print ("Demo Program for Boolean Literals")
        print ("Boolean Value1 :",boolean_1)
        print ("Boolean Value2 :",boolean_2)
# End of the Program
```

Output:

```
        Demo Program for Boolean Literals
        Boolean Value1 : True
        Boolean Value2 : False
```

(iv) Escape Sequences

In Python strings, the backslash "\" is a special character, also called the **"escape"** character. It is used in representing certain whitespace characters: **"\t"** is a tab, **"\n"** is a newline, and **"\r"** is a carriage return. For example to print the message "It's raining", the Python command is

```
>>> print ("It\'s rainning")
It's rainning
```

Python supports the following escape sequence characters.

Escape sequence character	Description	Example	Output
\\	Backslash	>>> print("\\test")	\test
\’	Single-quote	>>> print("Doesn\'t")	Doesn't
\”	Double-quote	>>> print("\"Python\"")	"Python"
\n	New line	print("Python","\n","Lang..")	Python Lang..
\t	Tab	print("Python","\t","Lang..")	Python Lang..

5.8 Python Data types

All data values in Python are objects and each object or value has type. Python has Built-in or Fundamental data types such as Number, String, Boolean, tuples, lists, sets and dictionaries etc.

5.8.1 Number Data type

The built-in number objects in Python supports integers, floating point numbers and complex numbers.

Integer Data can be decimal, octal or hexadecimal. Octal integer use digit **0** (Zero) followed by letter '**o**' to denote octal digits and hexadecimal integer use **0X** (Zero and either uppercase or lowercase X) and L (only upper case) to denote long integer.

Example :

```
102, 4567, 567              # Decimal integers
0o102, 0o876, 0o432         # Octal integers
0X102, 0X876, 0X432         # Hexadecimal integers
34L, 523L                   # Long decimal integers
```

A floating point data is represented by a sequence of decimal digits that includes a decimal point. An Exponent data contains decimal digit part, decimal point, exponent part followed by one or more digits.

Example :

```
123.34,  456.23,  156.23          # Floating point data
12.E04, 24.e04                    # Exponent data
```

Complex number is made up of two floating point values, one each for the real and imaginary parts.

5.8.2 Boolean Data type

A Boolean data can have any of the two values: True or False.

Example :

```
Bool_var1=True
Bool_var2=False
```

5.8.3 String Data type

String data can be enclosed in single quotes or double quotes or triple quotes.

Example :

```
Char_data = ‘A’
    String_data= "Computer Science"
    Multiline_data= ”””String data can be enclosed in single quotes or
    double quotes or triple quotes.”””
```

Points to remember:

- Python is a general purpose programming language created by Guido Van Rossum.
- Python shell can be used in two ways, viz., Interactive mode and Script mode.
- Python uses whitespace (spaces and tabs) to define program blocks
- Whitespace separation is necessary between tokens, identifiers or keywords.
- A Program needs to interact with end user to accomplish the desired task, this is done using Input-Output facility.
- Python breaks each logical line into a sequence of elementary lexical components known as Tokens.
- Keywords are special words that are used by Python interpreter to recognize the structure of program.

Evaluation

Part - I

Choose the best answer **(1 Marks)**

1. Who developed Python ?

   A) Ritche B) Guido Van Rossum

   C) Bill Gates D) Sunder Pitchai

2. The Python prompt indicates that Interpreter is ready to accept instruction.

   A) >>> B) <<<

   C) # D) <<

3. Which of the following shortcut is used to create new Python Program ?

   A) Ctrl + C B) Ctrl + F

   C) Ctrl + B D) Ctrl + N

4. Which of the following character is used to give comments in Python Program ?

   A) # B) & C) @ D) $

5. This symbol is used to print more than one item on a single line.

   A) Semicolon(;) B) Dollor($)

   C) comma(,) D) Colon(:)

6. Which of the following is not a token ?

   A) Interpreter B) Identifiers

   C) Keyword D) Operators

7. Which of the following is not a Keyword in Python ?

A) break	B) while
C) continue	D) operators

8. Which operator is also called as Comparative operator?

A) Arithmetic	B) Relational
C) Logical	D) Assignment

9. Which of the following is not Logical operator?

A) and	B) or
C) not	D) Assignment

10. Which operator is also called as Conditional operator?

A) Ternary	B) Relational
C) Logical	D) Assignment

Part - II

Answer the following questions : (2 Marks)

1. What are the different modes that can be used to test Python Program ?
2. Write short notes on Tokens.
3. What are the different operators that can be used in Python ?
4. What is a literal? Explain the types of literals ?
5. Write short notes on Exponent data?

Part - III

Answer the following questions : (3 Marks)

1. Write short notes on Arithmetic operator with examples.
2. What are the assignment operators that can be used in Python?
3. Explain Ternary operator with examples.
4. Write short notes on Escape sequences with examples.
5. What are string literals? Explain.

Part - IV

Answer the following questions : (5 Marks)

1. Describe in detail the procedure Script mode programming.
2. Explain input() and print() functions with examples.
3. Discuss in detail about Tokens in Python