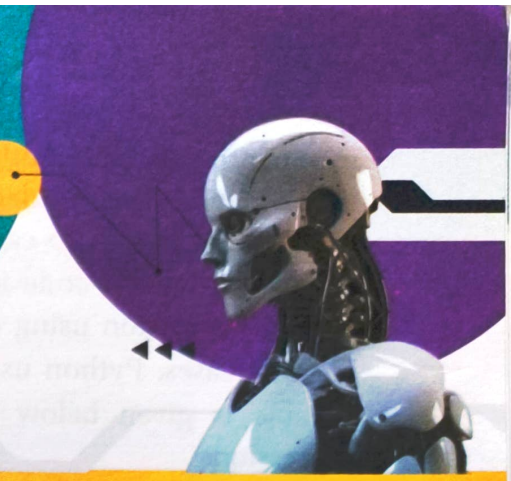
3 Introduction to Data and Programming with Python



3.3 Introduction to Operators

Learning Scope

Introduction, Operators, Usage of different operators: arithmetic (viz. addition, subtraction, multiplication, division, modulus, floor), assignment, comparison, logical, identity, membership) on data types, First step to Python programming, Input statement, Comment statement in Python.

INTRODUCTION

It is always fascinating to design Python programs which require the knowledge of some basic structure of any programming language. You already learnt some key points to Python programming that include indentation, data types in Python viz. numeric (int, float), Boolean, Sequence type (tuple, list, strings), Sets and Dictionary. In this segment, you will learn more about the components which are essentially needed for Python programming that includes operators.

Let us learn about operators in details.

OPERATORS

Operators are the symbols or tokens which perform various mathematical or logical operations. Python comprises of a number of operators to perform various tasks. The different types of operators are mentioned below:

1. Arithmetic Operators
2. Assignment Operators
3. Comparison Operators
4. Logical Operators
5. Identity Operators
6. Membership Operators

1. Arithmetic Operators

The operators which are used to perform arithmetical calculations in a program are known as arithmetic operators. Some basic calculations like addition, subtraction, multiplication, division,

exponentiation and modulus (to get the remainder when a number is divided by the other number) are often needed during programming. You can use the arithmetic operators such as +, –, *, /, ** and % to carry out some fundamental calculations.

When you write a code in Python, it is necessary to represent a mathematical expression as a Python expression using different arithmetical operators.

In some cases, Python uses arithmetical operators in different ways.

A table is given below to illustrate as how Python expressions differ from Mathematical Expressions:

Arithmetical Expressions	Python Expressions
40 × 5	40 * 5
5a + 6b	5*a + 6*b
a × b × c	a*b*c
ab + cd	a*b + c*d
$a^2 + b^2$	a**2 + b**2
$(a + b)^2$	(a+b)**2
2(l + b)	2*(l+b)
4abc	4*a*b*c
(3a – 4b)/abc	(3*a – 4*b)/a*b*c
$\frac{prt}{100}$	p*r*t/100

Just to Remember

Expression: A set of variables, constants and operators used together to obtain a meaningful result is known as an Expression. In other words, an expression in Python is a combination of operators and operands. The expression in Python produces some value or result after being interpreted by the Python interpreter.

For example, (x + y), (m – 15), (a*a + 2*a*b + b*b) are some expressions.

Python uses various types of arithmetic operators. Let us discuss these operators in detail.

(a) Addition Operator (+)

The addition operator (+) adds two or more values of the operands to yield the result. The result of this operator can be stored in a variable for further use, depending upon the user's need.

For example:

a=12; b=10.5

(i) >>>a +15 : results in 27

(ii) >>>8 + b : results in 18.5

(iii) >>>c = a + b : results in 22.5 and stores it in c

The resultant data type will depend upon the data type that are being used during the operations. Refer the given table:

Data types in operations	Resultant data types
int + int	int
int + float	float
float + int	float
float + float	float

(b) Subtraction Operator (–)

The subtraction operator (–) subtracts a value from the other value (as specified by the user) to yield the result.

For example:

m=15.4; n=10

(i) >>>m – 5 : results in 10.4

(ii) >>>25.5 – n : results in 15.5

(iii) >>>d = m – n : results in 5.4 and stores it in d

(c) Multiplication Operator (*)

The multiplication operator (*) finds the product of the two or more values to yield the result.

For example:

a=11; b=5.5

(i) >>>a * 5 : results in 55

(ii) >>>10 * b : results in 55.0

(iii) >>>p = a * b : results in 60.5 and stores it in p

Note: The resultant data type of Subtraction and Multiplication Operators follow the same rule as discussed in Addition Operator.

(d) Division Operator (/)

The division operator (/) provides the result after dividing the first operand by the second. It always returns the value in the float type (by default).

For example:

a=15; b=3

(i) >>>45/5 : results in 9.0

(ii) >>>10/3 : results in 3.333333333

(iii) >>>p = a/b : results in 5.0 and stores it in p

The resultant data type will always be float, irrespective of the data type that are being used during the operations. Refer the given table:

Data types in operations	Resultant data types
int / int	float
int / float	float
float / int	float
float / float	float

(e) Floor Division Operator (//)

The division operator (/) provides the complete result of the division operation which includes the digits after decimal point. On the contrary, the floor division operator (//) performs the division but results in only the integer part of the quotient.

Let us understand its application with some examples.

For example,

```
>>> 32/6
    5.33333333333333
>>> 32//6
    5
```

(f) Modulus Operator (%)

The modulus operator (%) is used to obtain the remainder when the first number is divided by the second. The result will either be an int or a float type data (depending on the data types of the inputs).

For example,

(i)
```
>>>37%5
2
```

(ii)
```
>>>62.5%5
2.5
```

(g) Exponentiation Operator (**)

In Python, the exponentiation operator (**) is a special operator which results in power raised to a base (i.e., $\mathbf{m^n}$) where **n** is raised to the base **m**. The return data type will either be an int or a float data type (depending on the data type of inputs).

For example,

```
a=11
```

(i)
```
>>> 5**3
125
```

(ii)
```
>>> a**2
121
```

Just to Know

Binary Operators: The operators which deal with two operands, is referred to as the Binary Operators. In this process, we use all the arithmetic operators such as +, –, *, /, **, // and % to carry out some fundamental calculations along with logical and relational operators.

Binary Arithmetic Operators at a glance!

Operators	Symbols	Format	Description
Addition	+	a+b	Returns the sum
Subtraction	–	a–b	Returns the difference
Multiplication	*	a*b	Returns the product
Division	/	a/b	Returns the result of complete division
Floor Division	//	a//b	Returns the quotient
Modulus/Remainder	%	a%b	Returns the remainder
Exponentiation Operator	**	a**b	Returns the value of $\mathbf{a^b}$

[In the above table, 'a' and 'b' are considered as the two variables.]

2. Assignment Operator

The assignment operator (=) is used to assign a value to a variable. The variable and the value to be assigned to it, must be placed to the left and right side of the (=) sign respectively.

Syntax: <variable>=<value(numeric or string)>

For example:

(i) a = 10 : an integer value 10 is assigned to the variable 'a'
(ii) b= –15 : an integer value –15 is assigned to the variable 'b'
(iii) ch= 'P' : a letter 'P' is assigned to the variable 'ch'
(iv) wd= "Python" : a string "Python" is assigned to the variable 'wd'

3. Comparison Operators

The *comparison operators* compare the values of two operands and determine the relationship between them. It always returns a Boolean value (*i.e.*, True or False). The comparison operators are mainly used in the control flow statements to direct the control for executing a part of the program by ignoring the other parts. In order to compare numbers, strings, lists, tuples, etc. Python provides six types of comparison operators.

The different types of comparison operators are as follows:

Symbol	Meaning	Format
<	Less than	a<b
>	Greater than	a>b
< =	Less than or equal to	a<=b
> =	Greater than or equal to	a>=b
= =	Equal to	a==b
! =	Not equal to	a!=b

These operators compare numbers as well as sequences (strings, lists and tuples).

4. Logical Operators

The Python language provides three logical operators namely *and*, *or* and *not*. These operators yield *True* or *False* depending upon the outcome of different conditions. The different types of logical operators along with their format are shown in the table:

Logical Operators	Symbol	Format
AND	and	(a>b) and (a>c)
OR	or	(a==b) or (a==c)
NOT	not	not (a==b)

5. Identity Operators

Identity operators are the operators which are used to compare the memory location of two objects.

In Python, when the values are assigned to variables, they not only store those values but also get assigned their locations in the memory. So, python provides *id( )* function to confirm whether the objects belong to the same memory location or to different memory locations.

Syntax: id(<variable>)

For example:

```
# A program to illustrate memory location
m=11; n=12; p=11
print("Memory location of m:",id(m))
print("Memory location of n:",id(n))
print("Memory location of p:",id(p))
```

Here, you can notice that the variables representing the same value have the same memory location, which are different otherwise.

Now, we will see the implementation of id( ) function with identity operators.

Python provides two special operators viz. *is* and *is not*, which are referred to as the *Identity Operators*. These operators are used to check whether two values are located at the same memory location or not, and results in True or False accordingly.

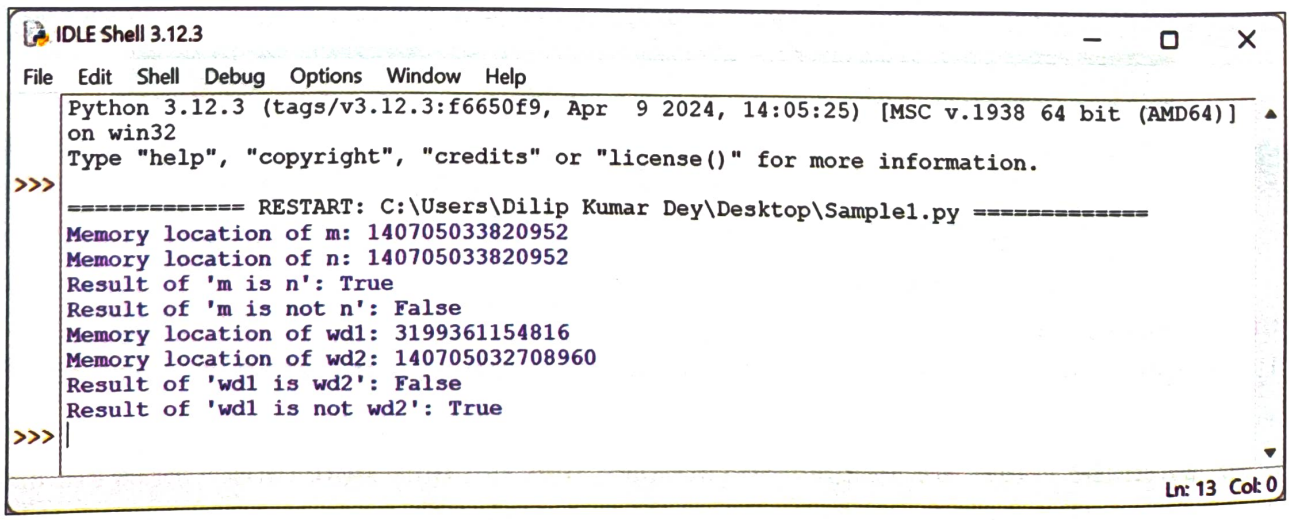
These operators work as illustrated below:

is: Results in *True*, if both the variables represent the same object (*i.e.*, the same memory location)

is not: Results in *True*, if both the variables represent the different objects (*i.e.*, different memory locations)

```
# A program to illustrate identity operator
m=12; n=12
wd1="Hello"; wd2="Python"
print("Memory location of m:",id(m))
print("Memory location of n:",id(n))
print("Result of 'm is n':", m is n)
print("Result of 'm is not n':", m is not n)
print("Memory location of wd1:",id(wd1))
print("Memory location of wd2:",id(wd2))
print("Result of 'wd1 is wd2':", wd1 is wd2)
print("Result of 'wd1 is not wd2':",wd1 is not wd2)
```

Output:



Here, you find that the memory locations of 'm' and 'n' are the same, whereas the memory locations of 'wd1' and 'wd2' are different. Hence, they result in True or False accordingly.

6. Membership Operators

There are two operators viz. *in* and *not in* which are said to be the *Membership Operators*. These operators are used to test whether a variable or value is an element of a sequence (*i.e.*, string, list, tuple or dictionary) or not. It results in *True* if yes, otherwise *False*.

It is illustrated as:

in: Results in *True*, if the value is found in the sequence

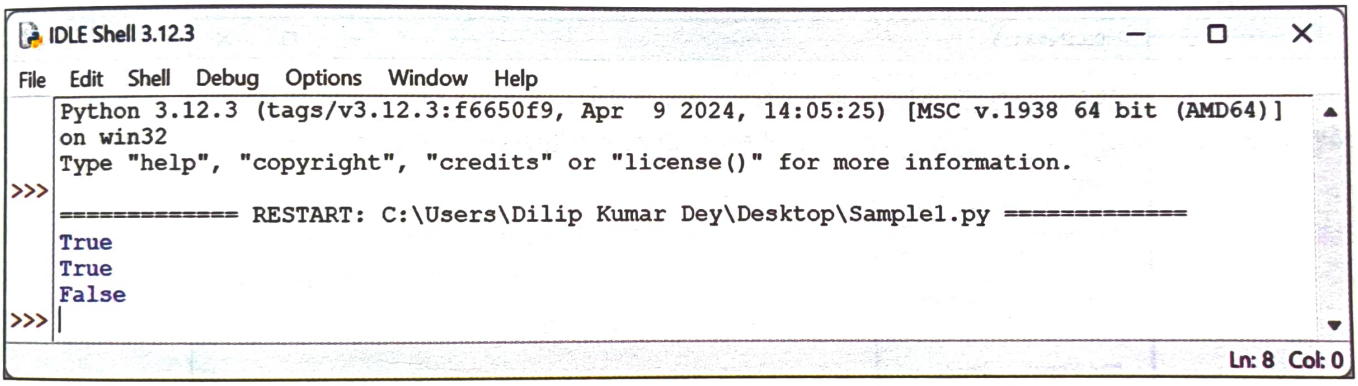
not in: Results in *True*, if the value is not found in the sequence

Let us understand with the examples shown below:

Example 1:

```
# Membership operators in Python using in Operator
animals = ["Cow", "Dog", "Lion"]
res1 = "Cow" in animals
res2 = "Lion" in animals
res3 = "Tiger" in animals
print(res1)
print(res2)
print(res3)
```

Output:

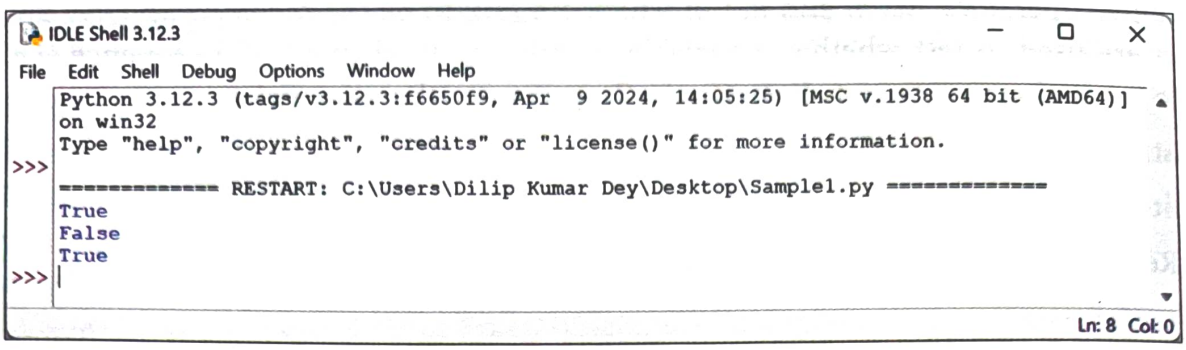


Refer the above example. The membership operators checks if "Cow" is present in the list of animals, resulting in True. In the subsequent examples, it checks for "Lion" and "Tiger". It results in True for the "Lion" and False for "Tiger" indicating its absence in the list.

Example 2:

```
# Membership operators using not in Operator
Subjects = ["Robotics", "Artificial Intelligence", "Data Science "]
res1 = "Java" not in Subjects
res2 = "Robotics" not in Subjects
res3 = "Machine Learning" not in Subjects
print(res1)
print(res2)
print(res3)
```

Output:



In the above example, the *not in* operator checks, if "Java" is not present in the list Subjects, resulting in True. In the subsequent examples, it checks for "Robotics" and "Machine Learning" with False and True outputs respectively, based on their presence or absence in the list.

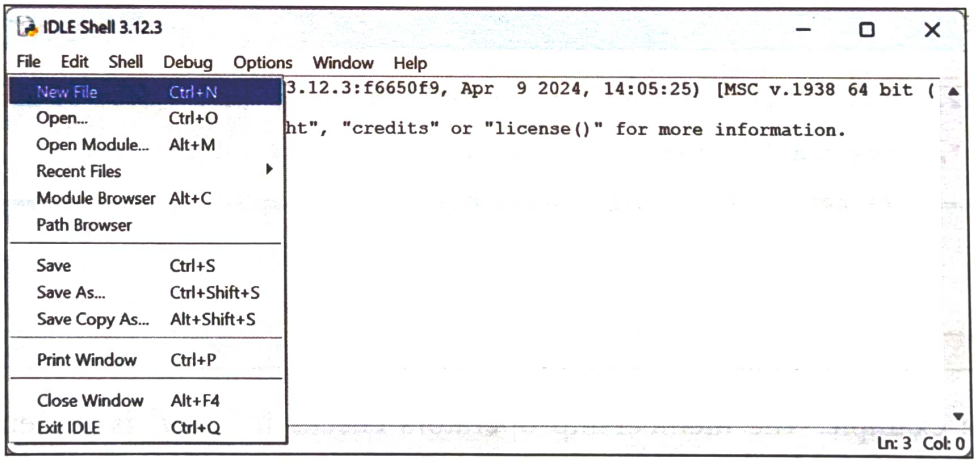
FIRST STEP TO PYTHON PROGRAMMING

In this segment, you will learn to create a program file in the Script mode. In this mode, you can save complete set of instructions in a program file that can be used for future puposes.

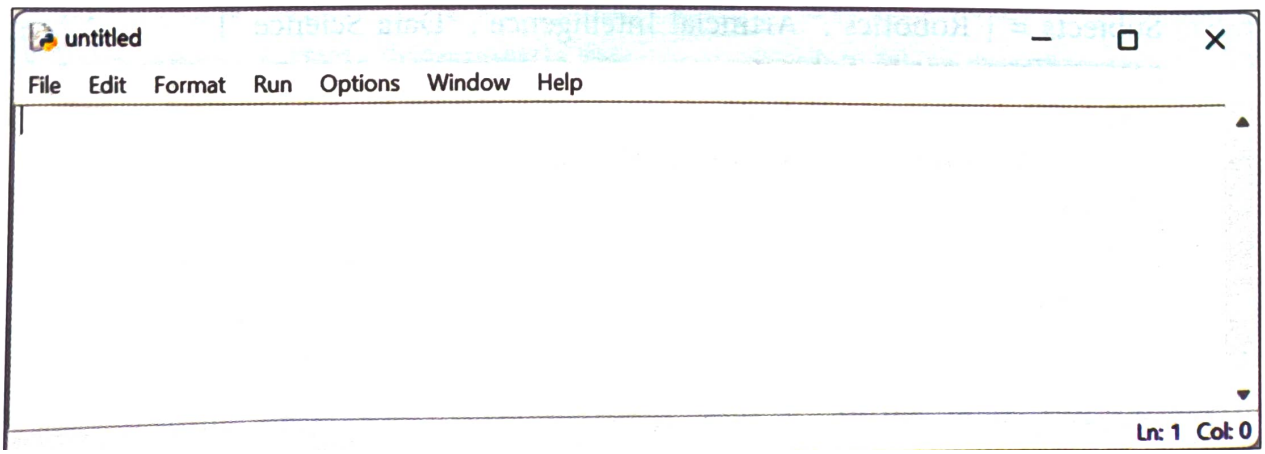
Let us learn to write a program in Script mode. To do so, you need to follow these steps:

Step 1: Click IDLE (Python 3.12 64-bit) shortcut icon which is pinned to taskbar. The IDLE Shell window appears on the screen.

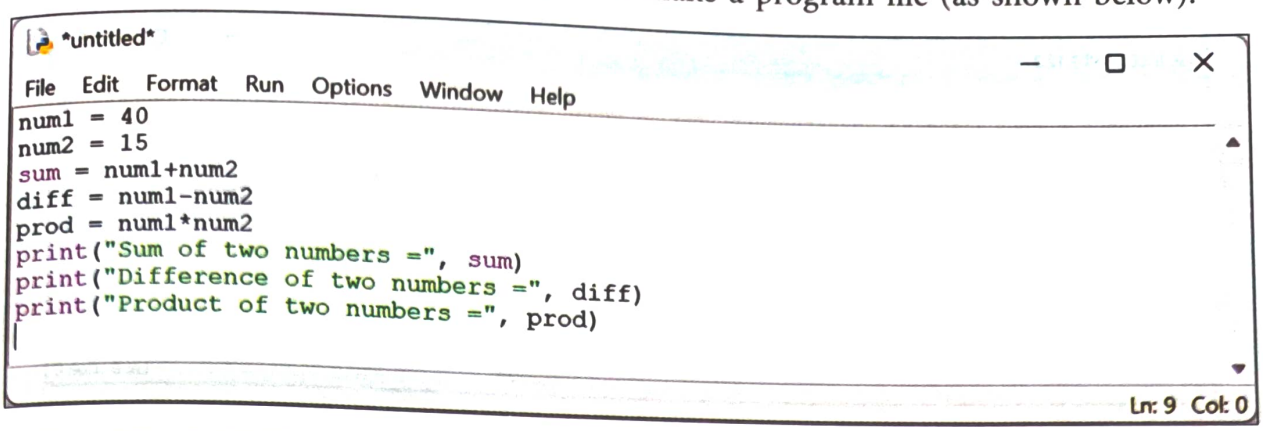
Step 2: Click 'File' and select 'New File' from the dropdown menu.

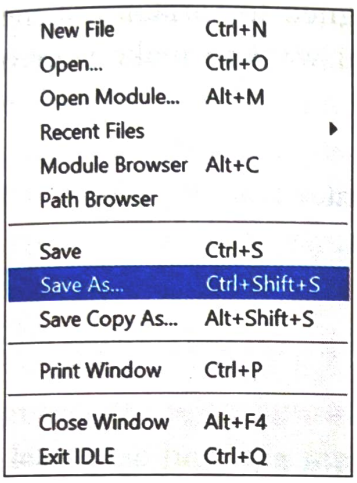


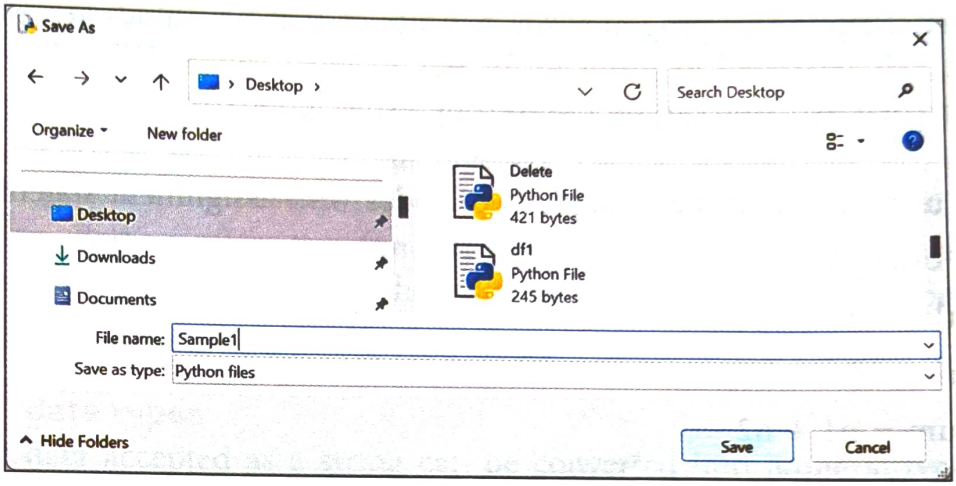
An untitled window will appear. This window provides a platform for the user to write programs on it.



Step 3: Type the code in the 'untitled' window to make a program file (as shown below):







Step 4: Click 'File' and select 'Save As' from the dropdown menu.

Step 5: Select the drive (here, desktop), where you want to save the program file on your computer.

Step 6: Enter the file name (say, Sample1) in the 'File name' text box.

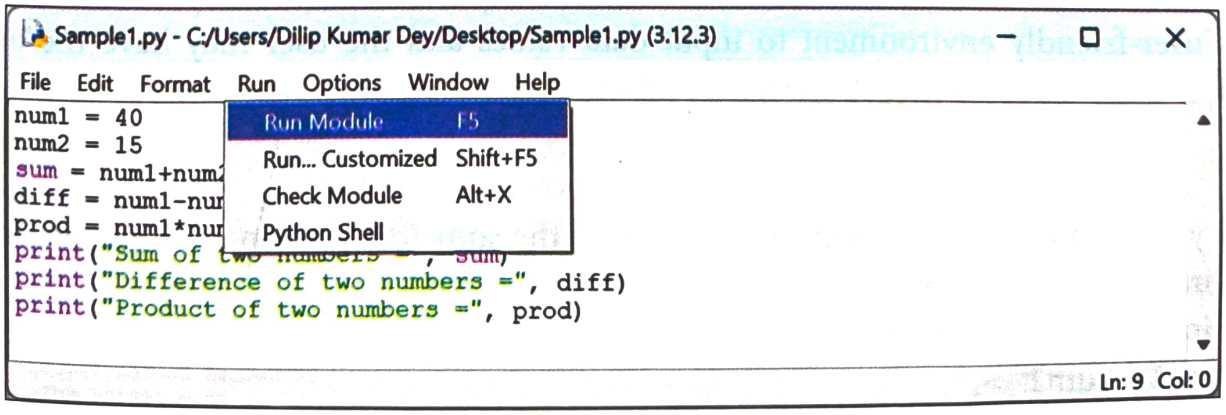
Step 7: Finally, click on 'Save' button.

Thus, the file will be saved in the system with the extension *.py* (by default).

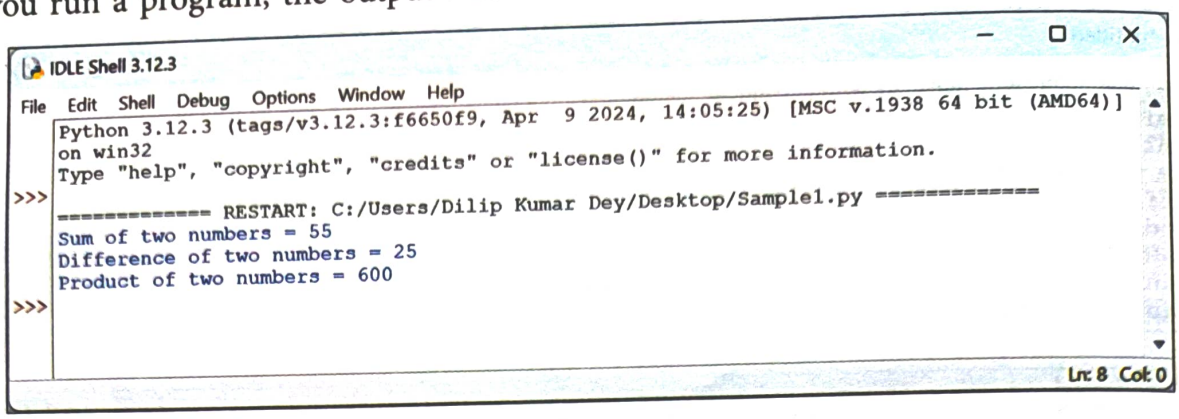
EXECUTION OF THE PROGRAM FILE

To execute the program file:

- Open file from saved location.
- Click 'Run' followed by Run Module (F5).



When you run a program, the output will be available on the interactive window (as shown).



Refer the previous program. You will notice that the values are assigned to variables. It means that it is restricted to result in for the assigned values. What if, you want to make your code user friendly? This can be resolved by using input statement.

Let us understand the limitations of codes with assignment statement.

For example,

```
n1 = 15
n2 = 10
sum = n1 + n2
print(sum)
```

In this code, two values (i.e., 15 and 10) have been assigned to two variables ('n1 ' and 'n2') to display the sum.

However, if the sum of any two numbers is to be calculated then we cannot do so by using the assignment statement. This is only possible with the use of input statement. With the use of input statement, you can enter any value to numeric variables or string variables at the time of execution. In other words, you can make your codes user friendly.

Input Statement in Python: input( )

The use of input statement ensures the interaction or communication with the user. The input often comes from the keyboard to accept the data from the user. Python language provides the input( ) function for reading data from the standard input device (*i.e.*, keyboard). This system provides user-friendly environment to input data values and the user may have the option to change the data values during the next execution of the program.

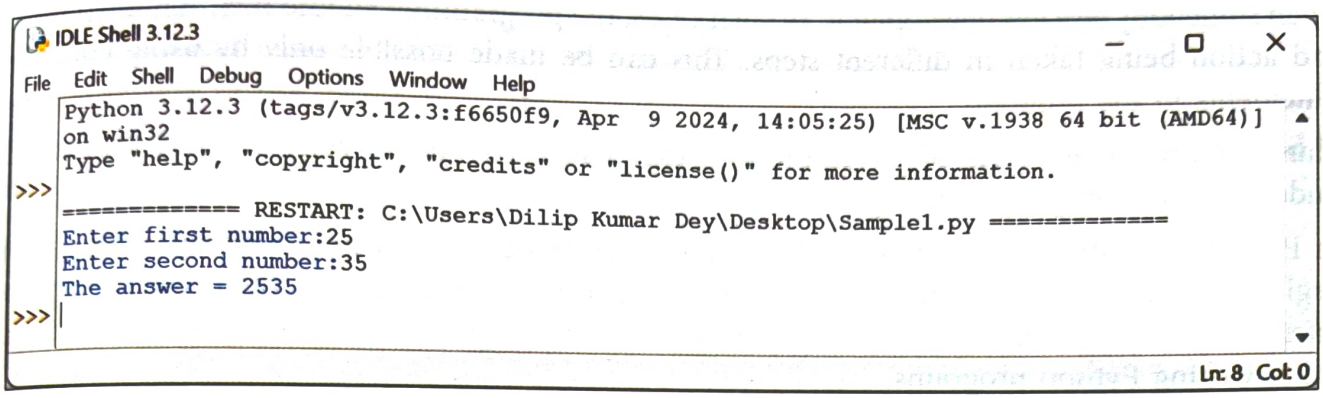
Syntax:

variable = input(<"Message"><Prompt>)

Suppose, you want to input two numbers to display the sum. The Python code can be written as:

```
num1 = input("Enter first number:")
num2 = input("Enter second number:")
sum = num1+ num2
print("The answer =",sum)
```

When you run the above code, you will find that Python has simply concatenated the values (as shown):



Probably, you will not be able to answer for the output obtained. In Python, the *input( )* will always accept the value entered from the keyboard as a string. When an integer value is given as an input, Python accepts as a string and they are put together. Hence, it does not able to perform an arithmetical sum to produce the desired result.

To overcome the above situation, Python offers two functions (i.e.; int( ) and float( )). These functions are capable of converting a string input into the value of their respective data types.

Note: By default, the input( ) function returns a String type value.

Use of input( ) with numeric data types

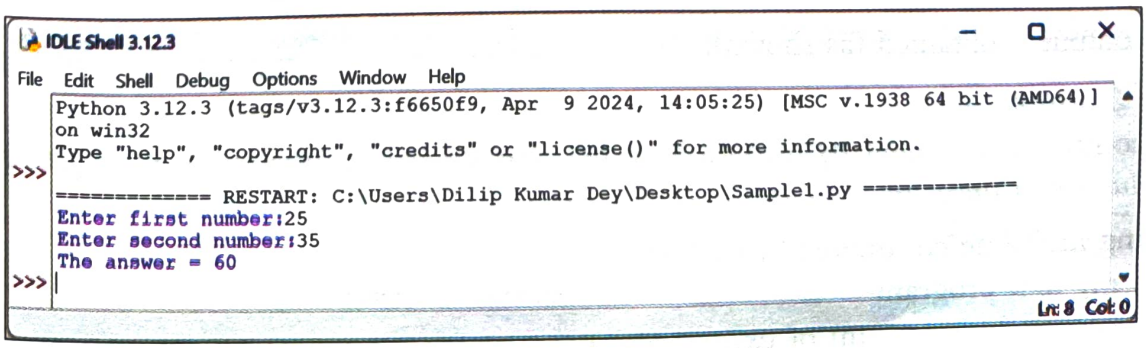
Now, let us see how the input data accepted as a string can be converted into numeric types (int or float) so that any arithmetical operation could be carried on it.

- **to accept an integer data type**
  Syntax: variable = int(input(<"Message">)) Prompt <input value>
- **to accept a float data type**
  Syntax: variable = float(input(<"Message">)) Prompt <input value>

Now, let us modify the above code as shown as:

```
num1=int(input("Enter first number:"))
num2=int(input("Enter second number:"))
sum=num1+ num2
print("The answer =",sum)
```

When you run the above code, the result will appear as shown below:



Comment Statements in Python

Sometimes, it becomes difficult for a user to understand the logic used in a program, especially when somebody else has developed it. In such cases, the programmer should mention the purpose and action being taken in different steps. This can be made possible only by using comment statements in the program.

Thus, comments are referred to as the non-executable statements which enable the user to understand the programming logic.

In Python, a comment statement starts with a # (hash) sign. It can be positioned either at the beginning along with the statements or at the end of a line. It must be noted that the Python interpreter ignores them while executing the code. You can give comments, wherever necessary, while writing Python programs.

There are two ways to give comments in Python:

(a) Single Line Comment

(b) Multiline Comment

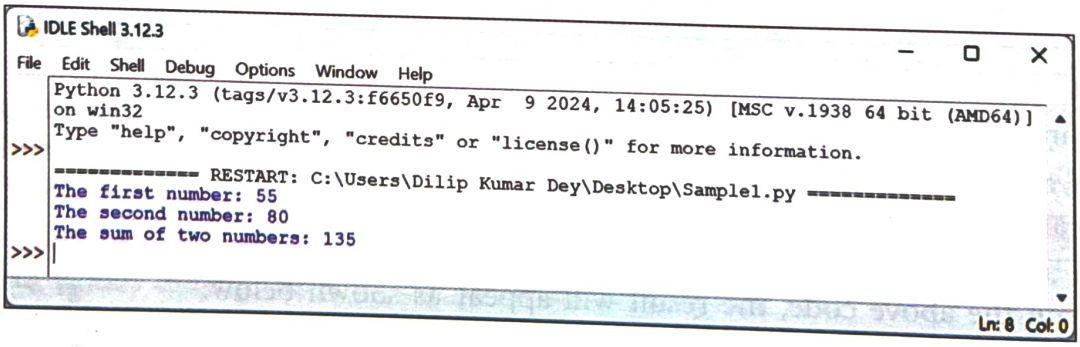
Single Line Comment

The single line comment is written simply by beginning a line with a hash (#) character. It may be written at the beginning of the program, in between the statements or along with the statements to explain the code in detail. It is illustrated as:

```
# A program to find the sum of two numbers
n1 = 55
n2 = 80
sum = n1+n2 # Finds the sum of two numbers
print("The first number:", n1)
print("The second number:", n2)
print("The sum of two numbers:", (sum)) # Displays the output
```

Single Line Comment

When you execute the above program, you will notice that the comment lines are ignored by the Python interpreter.



Thus, the output is obtained (as shown).

Multiline Comments

When a comment statement by using a hash (#) character are expressed in multiple lines, refer to as a *Multiline Comment.*

```
# Learning multiline comments in Python
# This is my first program
# A program to find the sum of two numbers
a=int(input("Enter first number:"))
```

Multilline Comment

```
b=int(input("Enter second number:"))
sum = a+b
print("The first number:", a)
print("The second number:", b)
print("The sum of two numbers:", sum)
```

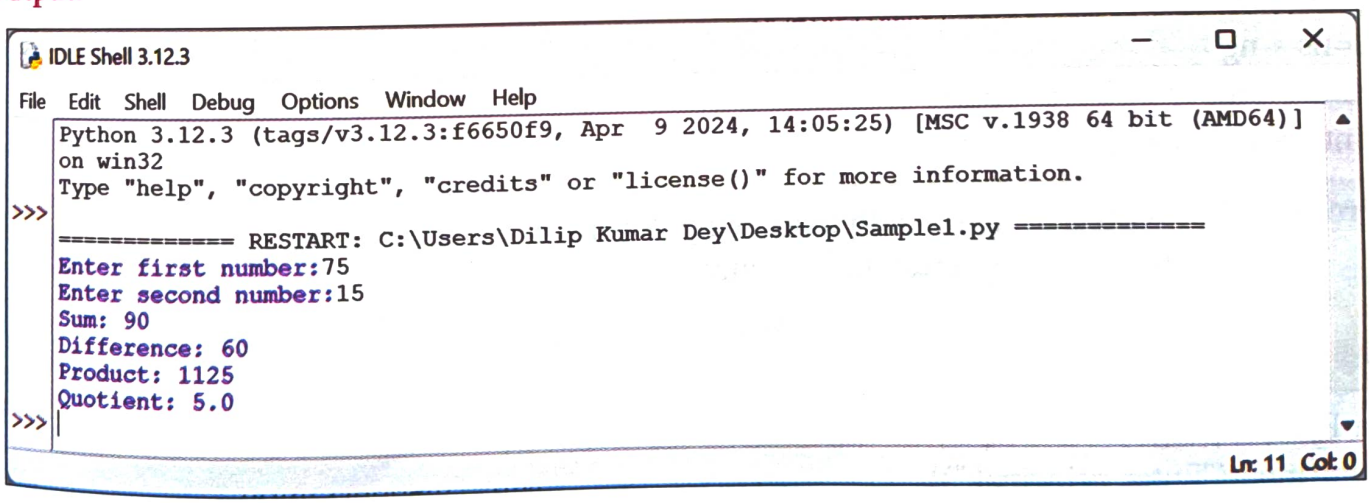
When you execute the above program, you will notice that the comments lines are ignored by the Python interpreter.

Solved Python Programs

Program 1 Write a program in Python to input two numbers. Display the sum, difference, product and quotient, performing arithmetic operations.

```
# a code to perform arithmetic operations
n1=int(input("Enter first number:"))
n2=int(input("Enter second number:"))
sum=n1+n2
diff=n1-n2
prod=n1*n2
div=n1/n2
print("Sum:", sum)
print("Difference:", diff)
print("Product:", prod)
print("Quotient:", div)
```

Output:



Program 2 Write a program in Python to input a number. Display the square and cube of the input number.
Sample Input: 8
Sample Output: 64
512

```
# a code to display square and cube of a number
num=int(input("Enter a number:"))
sq=num*num
cb=num*num*num
print("Square of the number:", sq)
print("Cube of the number:", cb)
```

Program 3 Write a program in Python to input temperature in Celsius. Convert the temperature into Fahrenheit and display the temperature

[**Hint :** $\frac{c}{5} = \frac{f-32}{9}$]

```
# a Python code to display temperature
c=int(input("Enter temperature in Celsius:"))
f=9*c/5+32
print("Temperature in Fahrenheit:", f)
```

Program 4 In a class of 60 students, 40% are girls. Write a program in Python to find the number of boys and girls in the class.

```
# a program to display number of boys and girls in a class
ns= int(input("Enter the number of students in the class:"))
ng=(ns*45)//100
nb=ns - ng
print("Number of boys =",nb)
print("Number of girls =",ng)
```

Program 5 Write a program in Python to input principal, rate and time. Calculate an display the simple interest and amount.

[**Hint :** Simple Interest $= \frac{p*r*t}{100}$]

```
# a Python program to display simple interest and amount
p=int(input("Enter principal:"))
r=int(input("Enter rate:"))
t=int(input("Enter time:"))
si = p*r*t/100
amt = p + si
print("Simple Interest = ₹",si)
print("Amount = ₹",amt)
```

Program 6 Write a program in Python to input the name and basic salary of an employee. Calculate and display the name, gross salary and net salary of the employee when:
da = 30% of basic
hra=18% of basic
pf=12.5% of basic
gross= basic+da+hra
ns=gross–pf

```
# a code to calculate and display gross and net salary of an employee
nm=input("Enter employee's name:")
bs=int(input("Enter Basic salary:"))
da=bs*30/100
hra=bs*18/100
pf=bs*12.5/100
gs=bs+da+hra
ns=gross-pf
print("Employee's name:", nm)
print("Gross Salary:₹", gs)
print("Net Salary:₹", ns)
```

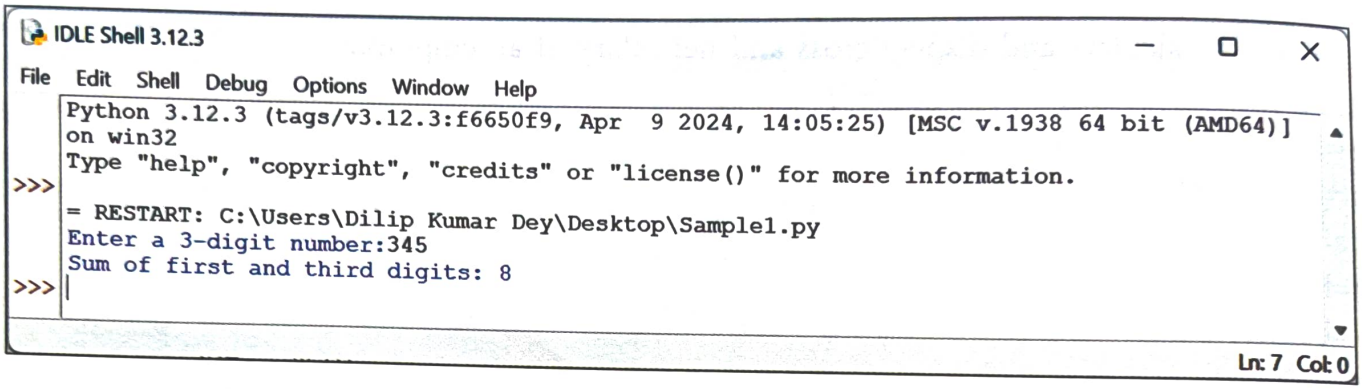
Program 7 Write a program in Python to input three numbers and swap them as under:
(i) First number becomes the second number
(ii) Second number becomes the third number
(iii) Third number becomes the first number
Sample Input: a = 95, b = 46, c = 81
Sample Output: a = 46, b= 81, c = 95

```
# a code to swap the numbers
num1=int(input("Enter first number:"))
num2=int(input("Enter second number:"))
num3=int(input("Enter third number:"))
num1, num2, num3=num2, num3, num1
print("After Swapping:")
print("First number:", num1)
print("Second number:", num2)
print("Third number:", num3)
```

Program 8 Write a program in Python to input a three-digit number and display the sum of first and last digit.
[Hint: Use % and // operators]
Sample Input: 345
Sample Output: 3 + 5 = 8

```
# a code to find the sum of first and third digit of a three-digit number
num=int(input("Enter a 3-digit number:"))
d1=num//100
rem=num%100
d2=rem%10
sum=d1+d2
print("Sum of first and third digits:", sum)
```

Output:



Chapter at a Glance

- The operators are the symbols or tokens which perform various mathematical or logical operations. Some of the operators include such as Arithmetic operators, Binary operators, Assignment operators.
- Some of the commonly used arithmetic operators are:
  - (i) Addition operator (+)
  - (ii) Subtraction operator(–)
  - (iii) Multiplication operator (*)
  - (iv) Division operator(/)
  - (v) Exponentiation operator (**)
  - (vi) Floor division operator(//)
  - (vii) Modulus operator (%)
- An arithmetic operator which deals with two operands is termed as the *Binary Operator*.
- The assignment operator (=) is used to assign a value to a variable.
- The *comparison operators* compare the values of two operands and determine the relationship between them.
- In Python, there are three logical operators viz. *and*, *or* and *not*. These operators yield *True* or *False* depending upon the outcome of the associated conditions.
- The use of input statement ensures the interaction or communication with the user.
- Python provides *input( )* function for reading data from the standard input device.
- By default, the *input( )* function accepts the data as a string.
- In Python, the Script mode is used to make a program file for the future use.
- In Python, the program file is saved in the system with the extension *.py* (by default).
- The comments are referred to as the non-executable statements.
- The single line comment is written simply by beginning a line with a hash (#) character.
- Python provides two special operators viz. is and *is not*, which are referred to as the *Identity Operators*.
- There are two operators viz. *in* and *not in* which are said to be the *Membership Operators*.
- When a comment statement by using a hash (#) character is expressed in multiple lines, refer to as a *Multiline Comment*.