SNS COLLEGE OF ENGINEERING
Kurumbapalayam (Po), Coimbatore – 641 107
An Autonomous Institution
Accredited by NBA – AICTE and Accredited by NAAC – UGC with 'A' Grade
Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai

DEPARTMENT OF CSE (IoT & CYBER SECURITY INCLUDING BLOCKCHAIN TECHNOLOGY)

19IT103 – COMPUTATIONAL THINKING AND PYTHON PROGRAMMING

- ❖ **A readable, dynamic, pleasant, flexible, fast and powerful language**

UNIT II DATA TYPES, EXPRESSIONS, STATEMENTS

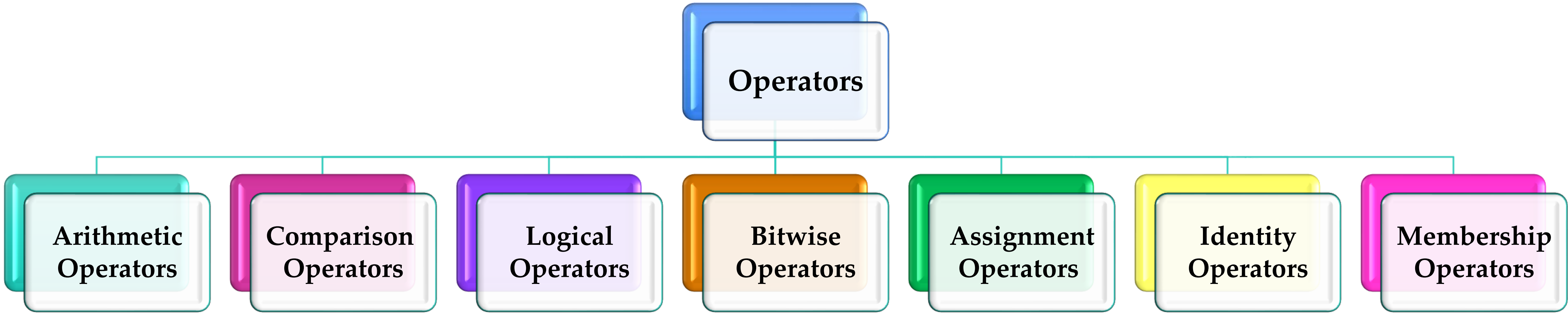
Python interpreter and interactive mode, debugging; values and types: int, float, boolean, string , and list; variables, expressions, statements, tuple assignment, **precedence of operators**, **comments**; Illustrative programs: exchange the values of two variables, circulate the values of n variables, distance between two points.

Recap

- Arithmetic Operators
- Comparison Operators
- Logical Operators
- Bitwise Operators
- Assignment Operators

Operators

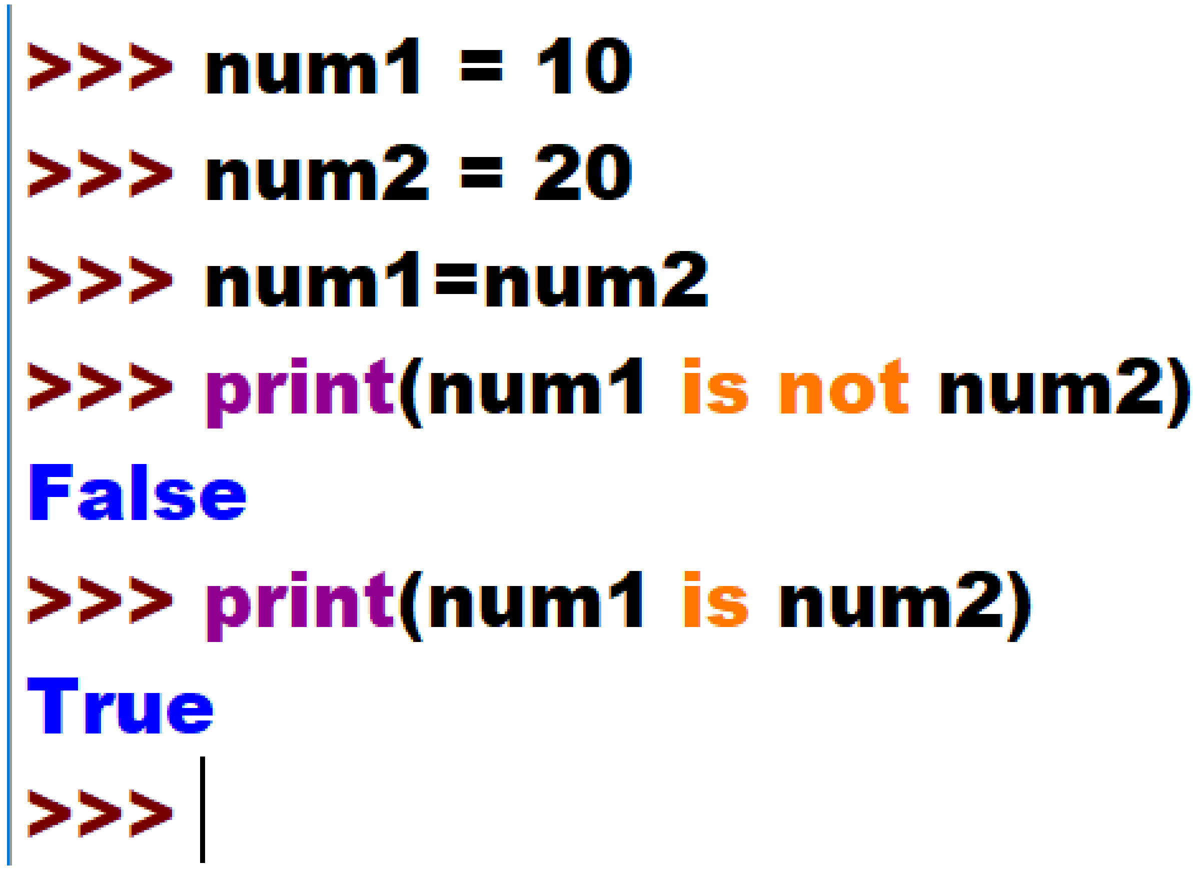
- Python Operators in general are used to **perform operations** on **values and variables**.



Identity Operators

- is and is not are the identity operators both are used to check if two values are located on the same part of the memory.
- Two variables that are equal do not imply that they are identical.
  - **is** **True if the operands are identical**
  - **is not** **True if the operands are not identical**

Identity Operators



```
>>> num1 = 10
>>> num2 = 20
>>> num1=num2
>>> print(num1 is not num2)
False
>>> print(num1 is num2)
True
>>>
```

Membership Operators

- **in and not in** are the membership operators; used to test whether a value or variable is in a sequence.
  - **in** **True if value is found in the sequence**
  - **not in** **True if value is not found in the sequence**

Example

Precedence and Associativity of Operators

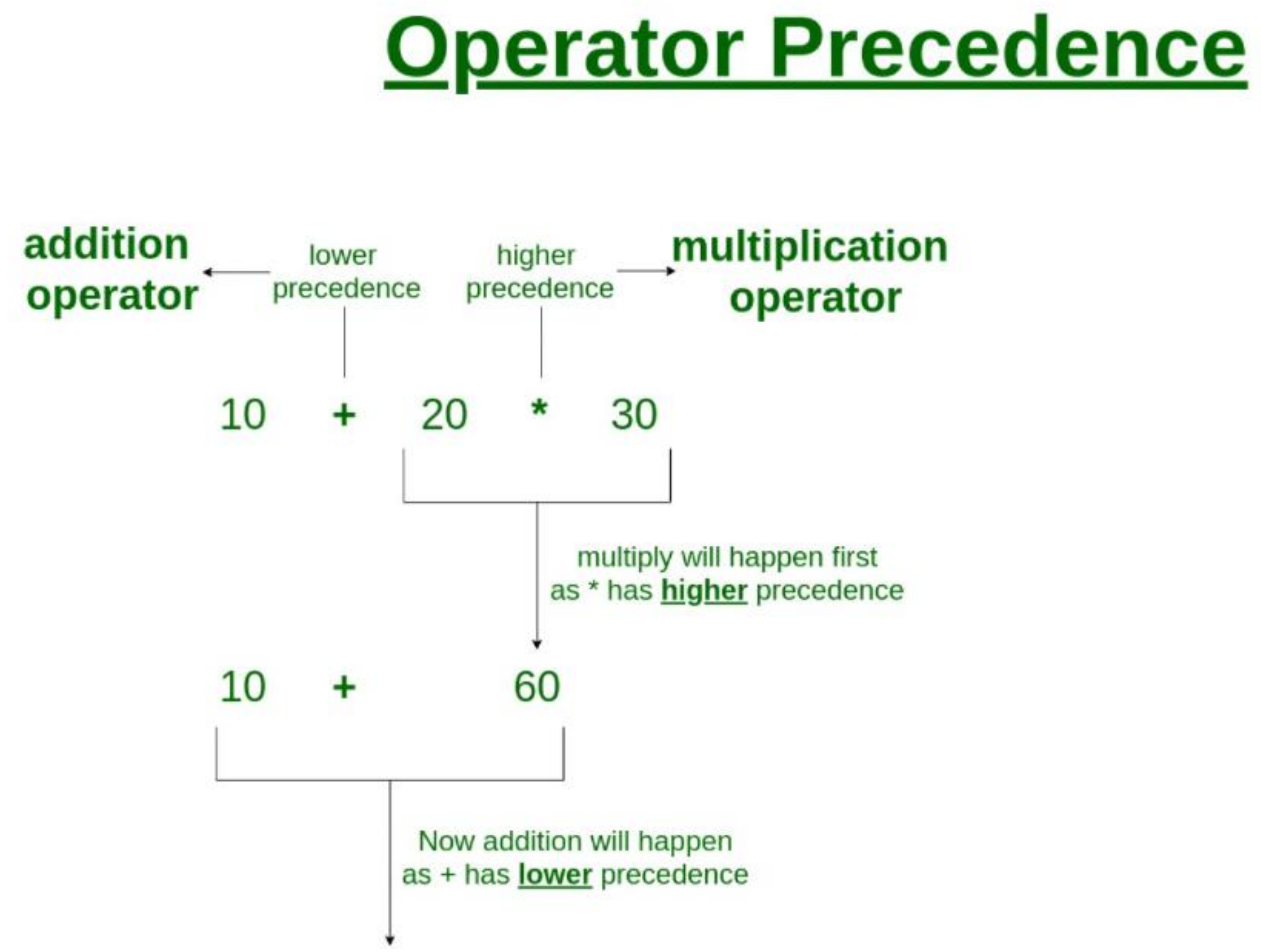
- When dealing with operators in Python we have to know about the concept of Python operator precedence and associativity as these determine the priorities of the operator.
- **Operator Precedence:** This is used in an expression with more than one operator with different precedence to determine which operation to perform first.

Precedence and Associativity of Operators

- Example: **10 + 20 * 30**
- **Option a: 900**
- **Option b: 70**
- **Which is correct?**

Precedence and Associativity of Operators

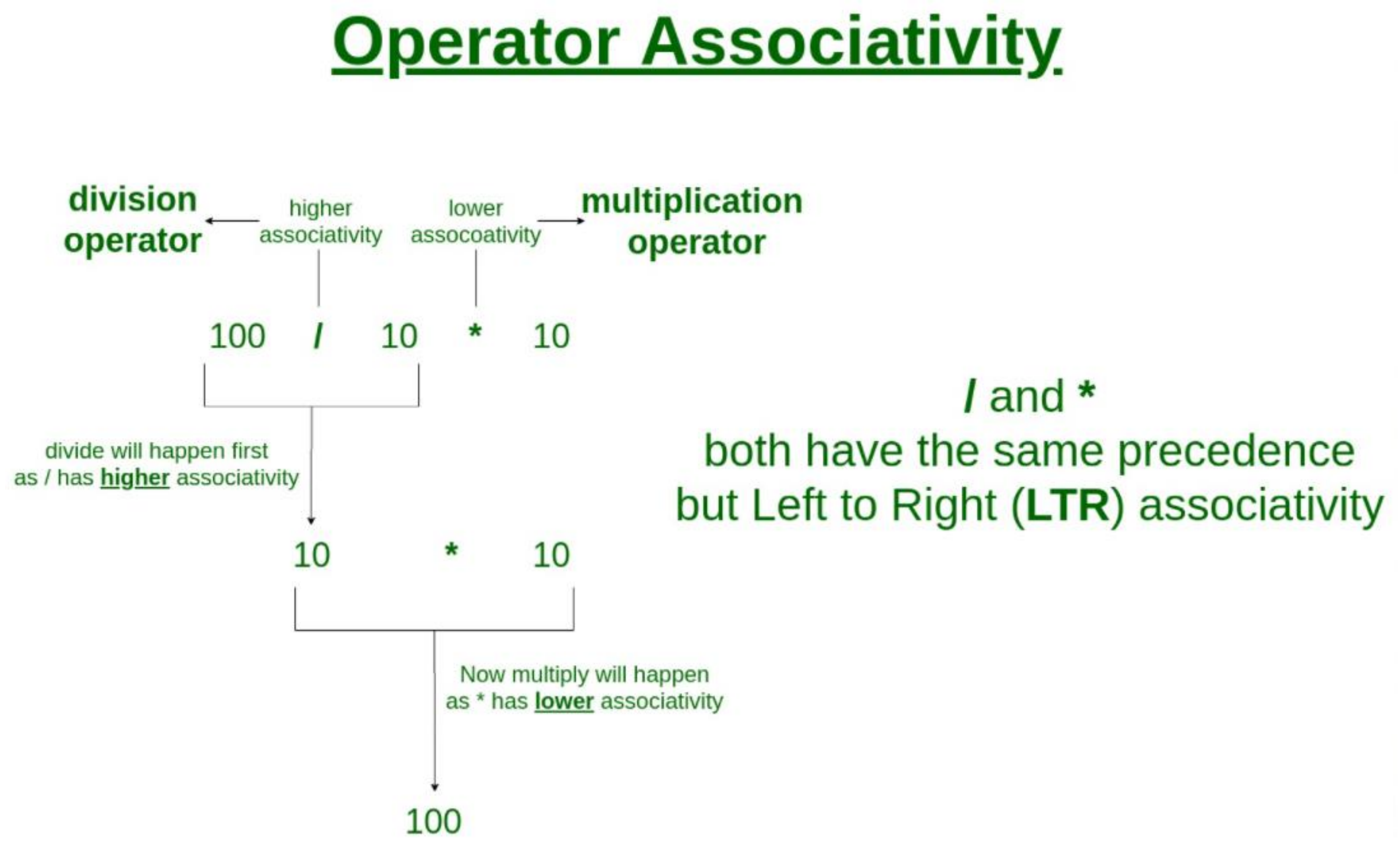
- Example: **10 + 20 * 30**



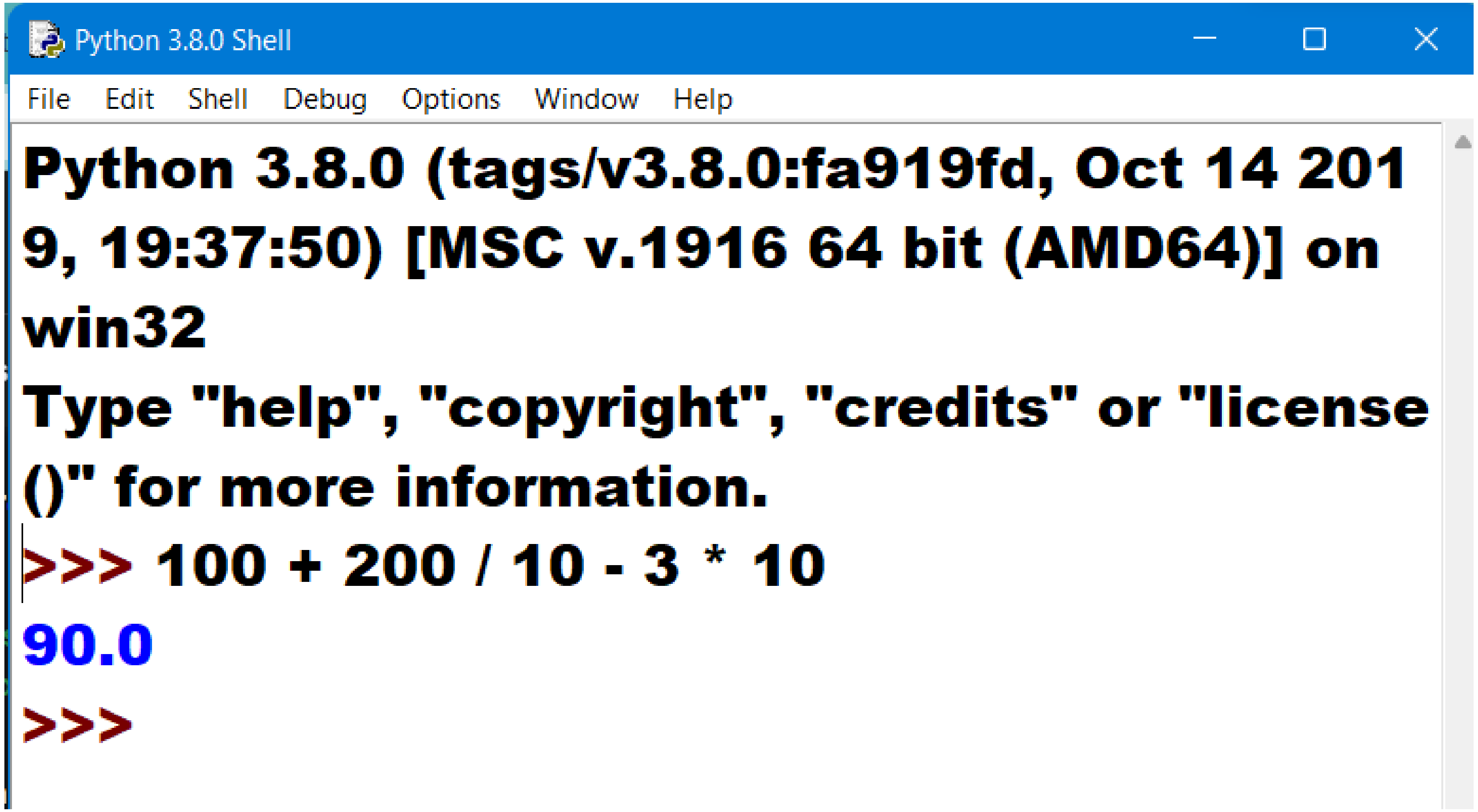
Precedence and Associativity of Operators

- **Operator Associativity:** If an expression contains two or more operators with the same precedence then Operator Associativity is used to determine.
- It can either be Left to Right or from Right to Left.
- **Example: '*' and '/' have the same precedence and their associativity is Left to Right**

Precedence and Associativity of Operators


Operator Associativity
division operator
higher associativity
lower assocoativity
multiplication operator
100 / 10 * 10
divide will happen first as / has higher associativity
10 * 10
Now multiply will happen as * has lower associativity
100
/ and * both have the same precedence but Left to Right (LTR) associativity

Precedence and Associativity of Operators



```
Python 3.8.0 (tags/v3.8.0:fa919fd, Oct 14 201
9, 19:37:50) [MSC v.1916 64 bit (AMD64)] on
win32
Type "help", "copyright", "credits" or "license
()" for more information.
>>> 100 + 200 / 10 - 3 * 10
90.0
>>>
```

Precedence and Associativity of Operators

Operator	Description	Associativity
( )	Parentheses	left-to-right
**	Exponent	right-to-left
* / %	Multiplication/division/modulus	left-to-right
+ –	Addition/subtraction	left-to-right
<< >>	Bitwise shift left, Bitwise shift right	left-to-right
< <= > >=	Relational less than/less than or equal to Relational greater than/greater than or equal to	left-to-right

Precedence and Associativity of Operators

Operator	Description	Associativity
== !=	Relational is equal to/is not equal to	left-to-right
is, is not in, not in	Identity Membership operators	left-to-right
&	Bitwise AND	left-to-right
^	Bitwise exclusive OR	left-to-right
|	Bitwise inclusive OR	left-to-right
not	Logical NOT	right-to-left

Precedence and Associativity of Operators

Operator	Description	Associativity
and	Logical AND	left-to-right
or	Logical OR	left-to-right
= += -= *= /= %= &= ^= |= <<= >>=	Assignment Addition/subtraction assignment Multiplication/division assignment Modulus/bitwise AND assignment Bitwise exclusive/inclusive OR assignment Bitwise shift left/right assignment	right-to-left

Comments

- Comments in Python are the lines in the code that are ignored by the compiler during the execution of the program.
- Comments enhance the readability of the code and help the programmers to understand the code very carefully.
- There are three types of comments in Python –
  - **Single line Comments**
  - **Multiline Comments**
  - **Docstring Comments**

Single-Line Comments

- Python single line comment starts with the hashtag symbol **(#)** with no white spaces and lasts till the end of the line.
- If the comment exceeds one line then put a hashtag on the next line and continue the comment.
- Python's single-line comments are proved useful for supplying short explanations for variables, function declarations, and expressions.

```
# Print "GeeksforGeeks !" to console
```

Multi-Line Comments

- Python does not provide the option for multiline comments.
- However, there are different ways through which we can write multiline comments.
- **Using Multiple Hashtags (#)**

```
# Python program to demonstrate
# multiline comments
```

Multi-Line Comments

- Using String Literals

```
""" Python program to demonstrate
multiline comments"""
```

Python Docstring

- Python docstring is the string literals with triple quotes that are appeared right after the function.
- It is used to associate documentation that has been written with Python modules, functions, classes, and methods.
- It is added right below the functions, modules, or classes to describe what they do.
- In Python, the docstring is then made available via the __doc__ attribute.

Python Docstring

Example