



**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**



# **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.



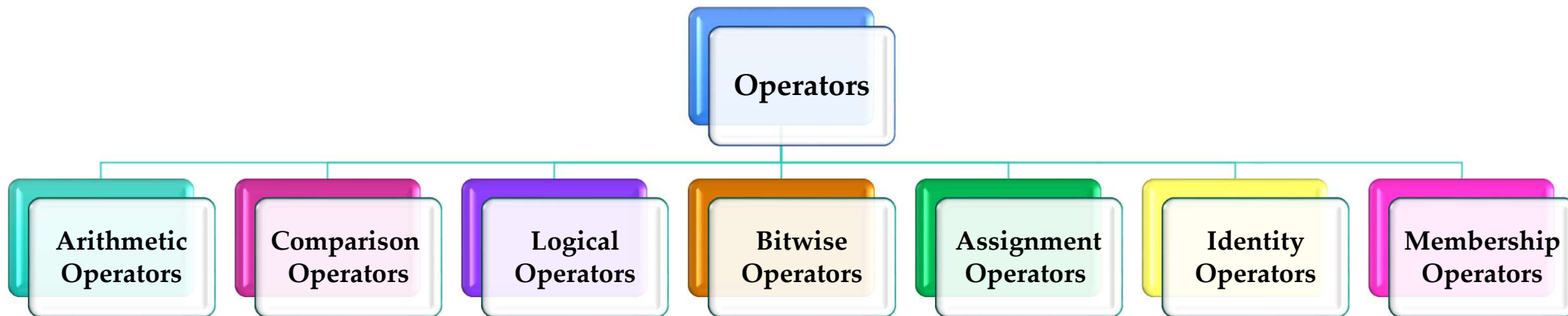
# Recap

- Expressions
- Statement
- Tuple Assignment




# Operators

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





# Arithmetic Operators

- Arithmetic operators are used to performing **mathematical operations** like **addition, subtraction, multiplication, and division**.
  - There are 7 arithmetic operators in Python :
    - **Addition (+)**
    - **Subtraction (-)**
    - **Multiplication (\*)**
    - **Division (/)**
    - **Modulus (%)**
    - **Exponentiation (\*\*)**
    - **Floor division(//)**
- 


# Arithmetic Operators

Operator	Description	Syntax
+	<b>Addition:</b> adds two operands	$x + y$
-	<b>Subtraction:</b> subtracts two operands	$x - y$
*	<b>Multiplication:</b> multiplies two operands	$x * y$
/	<b>Division (float):</b> divides the first operand by the second	$x / y$
//	<b>Division (floor):</b> divides the first operand by the second	$x // y$
%	<b>Modulus:</b> returns the remainder when the first operand is divided by the second	$x \% y$
**	<b>Power:</b> Returns first raised to power second	$x ** y$

Arithmetic Operator - Example



# Comparison/Relational Operators

- Comparison of Relational operators **compares the values**. It either **returns True or False** according to the condition.
  - There are 6 comparison operators in Python :
    - **Greater than (>)**
    - **Less than (<)**
    - **Equal to (==)**
    - **Not equal to (!=)**
    - **Greater than or equal to (>=)**
    - **Less than or equal to (<=)**
- 

# Comparison/Relational Operators


Operator	Description	Syntax
>	<b>Greater than:</b> True if the left operand is greater than the right	$x > y$
<	<b>Less than:</b> True if the left operand is less than the right	$x < y$
==	<b>Equal to:</b> True if both operands are equal	$x == y$
!=	<b>Not equal to –</b> True if operands are not equal	$x != y$
>=	<b>Greater than or equal to:</b> True if left operand is greater than or equal to the right	$x >= y$
<=	<b>Less than or equal to:</b> True if left operand is less than or equal to the right	$x <= y$

Relational Operator - Example





# Logical Operators

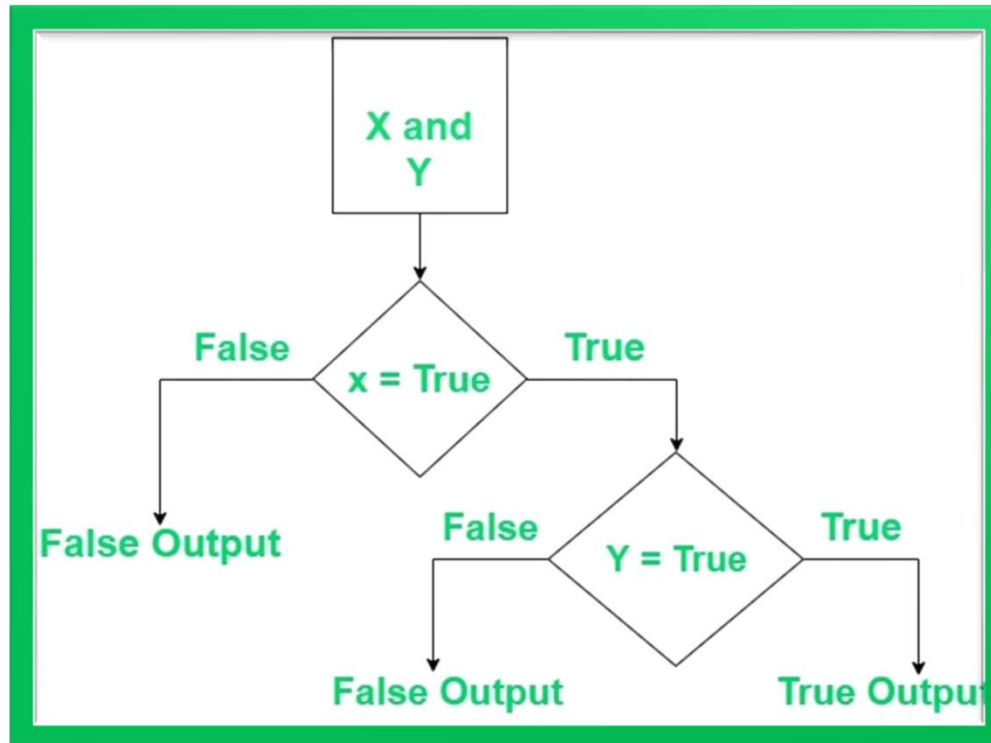
- Logical operators perform **Logical AND, Logical OR, and Logical NOT operations.**
  - It is used to **combine conditional statements.**
  - There are 3 basic logical operators in Python :
    - **and**
    - **or**
    - **not**
- 

# Logical Operators

Operator	Description	Syntax
<b>and</b>	<b>Logical AND: True if both the operands are true</b>	<b>x and y</b>
<b>or</b>	<b>Logical OR: True if either of the operands is true</b>	<b>x or y</b>
<b>not</b>	<b>Logical NOT: True if the operand is false</b>	<b>not x</b>

# Logical AND

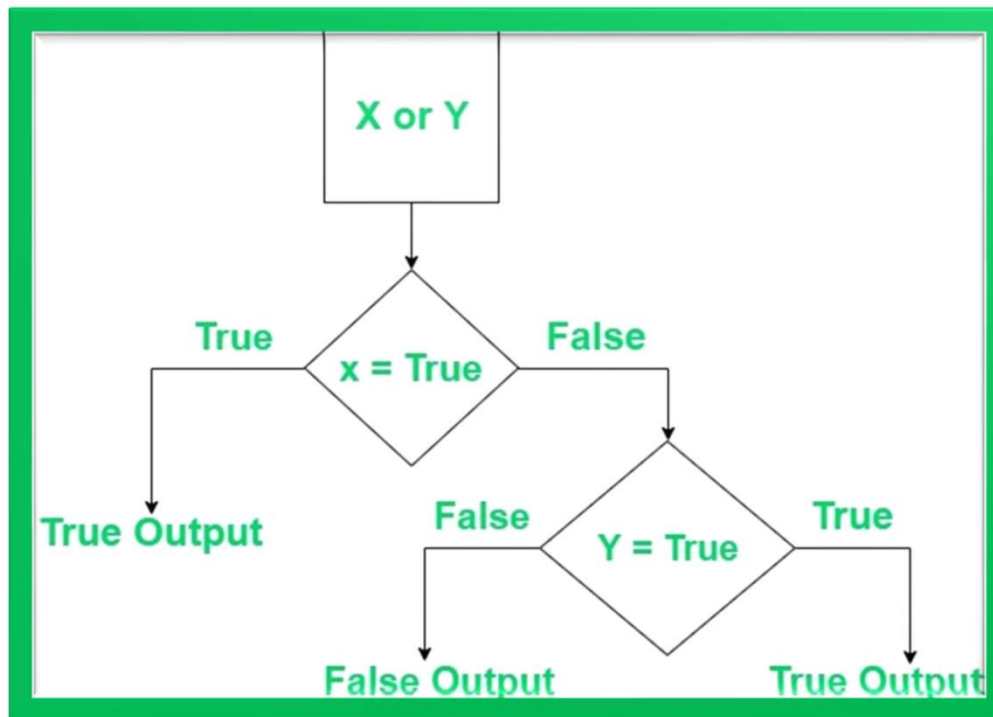
- Logical operator returns True if both the operands are True else it returns False.



[Logical AND- Example](#)

# Logical OR

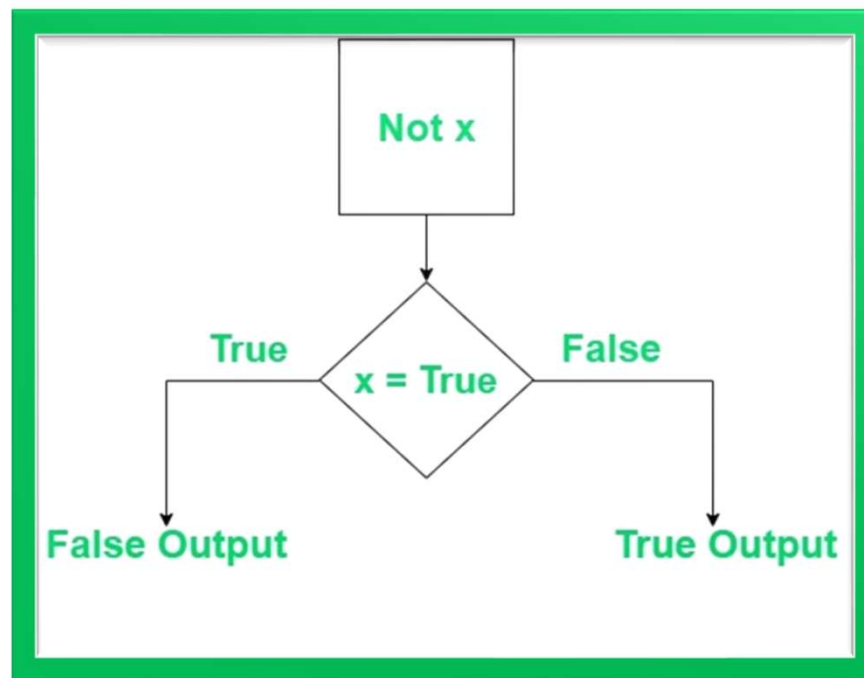
- Logical or operator returns True if either of the operands is True.



[Logical OR- Example](#)

# Logical NOT


- Logical or operator returns True if either of the operands is True.



Logical NOT - Example



# Bitwise Operators

- Bitwise operators act on bits and perform the **bit-by-bit operations**.
  - These are used to operate on binary numbers.
  - The integers are first **converted into binary** and then operations are performed on **bit by bit**, hence the name bitwise operators.
  - Then the result is returned in **decimal format**.
- 

# Bitwise Operators

Operator	Description	Syntax
<b>&amp;</b>	<b>Bitwise AND</b>	<b>x &amp; y</b>
<b> </b>	<b>Bitwise OR</b>	<b>x   y</b>
<b>~</b>	<b>Bitwise NOT</b>	<b>~x</b>
<b>^</b>	<b>Bitwise XOR</b>	<b>x ^ y</b>
<b>&gt;&gt;</b>	<b>Bitwise right shift</b>	<b>x&gt;&gt;</b>
<b>&lt;&lt;</b>	<b>Bitwise left shift</b>	<b>x&lt;&lt;</b>

Bitwise Operator - Example

# Assignment Operators

- Assignment operators are used to assigning values to the variables.

Operator	Description	Syntax
=	Assign value of right side of expression to left side operand	$x = y + z$
+=	Add and Assign: Add right side operand with left side operand and then assign to left operand	$a += b$
-=	Subtract AND: Subtract right operand from left operand and then assign to left operand: True if both operands are equal	$a -= b$
*=	Multiply AND: Multiply right operand with left operand and then assign to left operand	$a *= b$
<u>/=</u>	Divide AND: Divide left operand with right operand and then assign to left operand	$a /= b$
%=	Modulus AND: Takes modulus using left and right operands and assign result to left operand	$a \% = b$



# Assignment Operators

Operator	Description	Syntax
<code>//=</code>	<b>Divide(floor) AND: Divide left operand with right operand and then assign the value(floor) to left operand</b>	<code>a //= b</code>
<code>**=</code>	<b>Exponent AND: Calculate exponent(raise power) value using operands and assign value to left operand</b>	<code>a **= b</code>
<code>&amp;=</code>	<b>Performs Bitwise AND on operands and assign value to left operand</b>	<code>a &amp;= b</code>
<code> =</code>	<b>Performs Bitwise OR on operands and assign value to left operand</b>	<code>a  = b</code>
<code>^=</code>	<b>Performs Bitwise xOR on operands and assign value to left operand</b>	<code>a ^= b</code>
<code>&gt;&gt;=</code>	<b>Performs Bitwise right shift on operands and assign value to left operand</b>	<code>a &gt;&gt;= b</code>
<code>&lt;&lt;=</code>	<b>Performs Bitwise left shift on operands and assign value to left operand</b>	<code>a &lt;&lt;= b</code>

## Assignment Operator - Example