Python Operators

Operators

- Operators are pre-defined symbols that allow to perform specific actions
- An expression contains operators and operands
- The operators are assigned precedence values
- This precedence value indicates the order in which the operators are evaluated in an expression

Arithmetic Operators - I

Arithmetic Operators are used to perform operations on numbers

We use the BODMAS rule to solve precedence between the arithmetic operators.

Brackets are used to force precedence of operators

Arithmetic Operators - II

- Following table lists the different arithmetic operators in Python:

Operator	**Name**	**Description**
+	Addition	Adds the operands
-	Subtraction	Subtracts the second operand from the first operand
*	Multiplication	Multiplies operands with each other
/	Division	Divides the first operand by the second operand
//	Integer Division	Floor Division – Gives only Integer Quotient
%	Modulus	Returns the remainder when the first operand is divided by the second operand
**	Exponent	Calculates exponential power value

Relational Operators - I

- Following table lists the different relational operators in Python

Operator	Name	Description
==	Equal to	Returns true if both the operands are equal
!=	Not equal to	Returns true if the first operand is not equal to the second operand
<	Less than	Returns true if the first operand is less than the second operand

Relational Operators - II

Operator	Name	Description
<=	Less than or equal to	Returns true if the first operand is less than or equal to the second operand
>	Greater than	Returns true if the first operand is greater than the second operand
>=	Greater than or equal to	Returns true if the first operand is greater than or equal to the second operand

Logical Operators

Logical operators enable us to combine two or more test expression in a condition

Evaluate expressions and return a Boolean value

Logical Operators - I

- Python provides us with the following logical operators:

Operator	Description	Example
and Logical AND	If both the operands are true then condition becomes true.	(a and b) is true.
or Logical OR	If any of the two operands are non-zero then condition becomes true.	(a or b) is true.
not Logical NOT	Used to reverse the logical state of its operand.	Not(a and b) is false.

Bitwise Operators - I

Operate on the bits of an operand

Work on small-scale binary representation of data

Bitwise Operators - II

- Following table lists the different bitwise operators in

Operator	**Name**	**General Form**	**Description**
&	AND	Operand1 & Operand2	Sets to 1 if both the bits of both the operands are 1 and 0 otherwise
|	OR	Operand1 | Operand2	Sets to 1 if either of the bits of the operands are 1 and 0 otherwise

Bitwise Operators - III

Operator	Name	General Form	Description
^	EXCLUSIVE-OR	Operand1 ^ Operand2	Compares two bits and sets the bit to 1 if the bits are different and 0 otherwise
~	COMPLEMENT	~ Operand	Compares and sets the bits that are not set and 0 otherwise

Bitwise Operators - IV

Operator	Name	General Form	Description
<<	SHIFT LEFT	Operand1 << Operand2	Shifts the bits of Operand1, Operand2 times to the left
>>	SHIFT RIGHT	Operand1 >> Operand2	Shifts the bits of Operand1, Operand2 times to the right

Assignment Operator

Enables us to set the operand on the left side to the value of the expression on the right side

‘=’ sign is the assignment operator

Different from the equal to ‘= ‘sign used as the relational operator

PYTHON MEMBERSHIP OPERATORS

Operator	Description	Example
in	Evaluates to true if it finds a variable in the specified sequence and false otherwise.	x in y, here in results in a 1 if x is a member of sequence y.
not in	Evaluates to true if it does not finds a variable in the specified sequence and false otherwise.	x not in y, here not in results in a 1 if x is not a member of sequence y.

Precedence of Operators

Sr.No.	Operator & Description
1	** Exponentiation (raise to the power)
2	~ + - Complement, unary plus and minus (method names for the last two are +@ and -@)
3	* / % // Multiply, divide, modulo and floor division
4	+ - Addition and subtraction
5	>> << Right and left bitwise shift
6	& Bitwise 'AND'
7	^ | Bitwise exclusive `OR' and regular `OR'
8	<= < > >= Comparison operators
9	<> == != Equality operators
10	= %= /= //= -= += *= **= Assignment operators
11	is is not Identity operators
12	in not in Membership operators
13	not or and Logical operators