PROGRAMMING WITH PYTHON

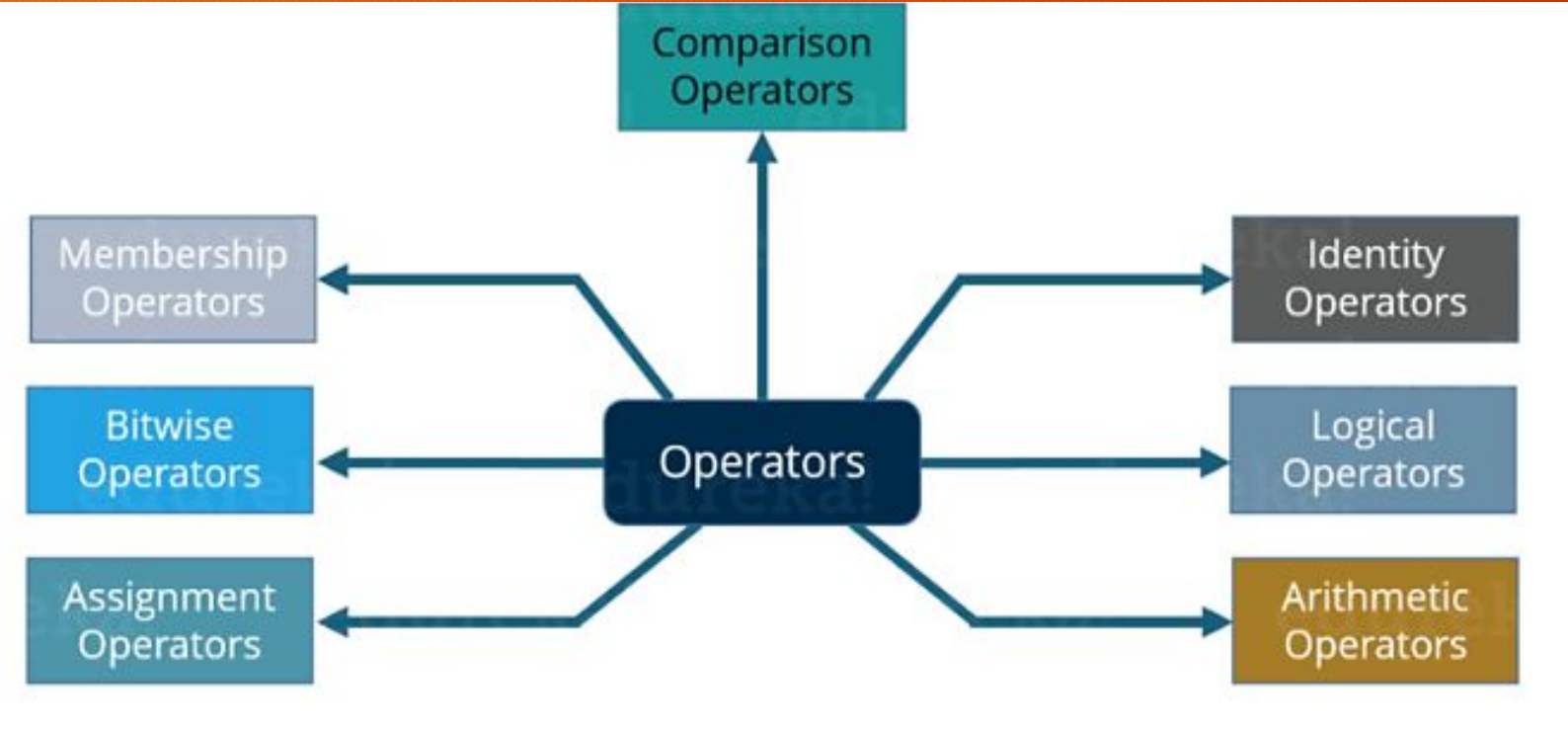
Karuna sheel

NIELIT kurukshetra

Operators in Python

Operators are the constructs which can manipulate the values of the operands. Consider the expression 2 + 3 = 5, here 2 and 3 are operands and + is called operator.

Python supports the following types of Operators:



Arithmetic Operators:

These Operators are used to perform mathematical operations like addition, subtraction etc. Assume that A = 10 and B = 20 for the below table.

Operator	Description	Example
+ Addition	Adds values on either side of the operator	A + B = 30
- Subtraction	Subtracts the right hand operator with left hand operator	A - B = -10
* Multiplication	Multiplies values on either side of the operator	A * B = 200
/ Division	Divides left hand operand with right hand operator	A / B = 0.5
% Modulus	Divides left hand operand by right hand operand and returns remainder	B % A = 0
** Exponent	Performs exponential (power) calculation on operators	A ** B = 10 to the power 20
// Floor Division	Performs division and return floor value	a= 31, b=5 a // b Return 6

Output = 31, 11, 210, 2.1, 1, 8

example :

```
a = 21
b = 10
c = 0

c = a + b
print ( c )

c = a - b
print ( c )

c = a * b
print ( c )

c = a / b
print ( c )

c = a % b
print ( c )
a = 2
b = 3
c = a**b
print ( c )
```

Comparison Operators:

These Operators compare the values on either sides of them and decide the relation among them. Assume A = 10 and B = 20.

Operator	Description	Example
==	If the values of two operands are equal, then the condition becomes true.	(A == B) is not true
!=	If values of two operands are not equal, then condition becomes true.	(A != B) is true
>	If the value of left operand is greater than the value of right operand, then condition becomes true.	(a > b) is not true
<	If the value of left operand is less than the value of right operand, then condition becomes true.	(a < b) is true
>=	If the value of left operand is greater than or equal to the value of right operand, then condition becomes true.	(a >= b) is not true
<=	If the value of left operand is less than or equal to the value of right operand, then condition becomes true.	(a <= b) is true

example:

```
a = 21
b = 10
c = 0
if ( a == b ):
        print ("a is equal to b")
else:
        print ("a is not equal to b")

if ( a != b ):
        print ("a is not equal to b")
else:
        print ("a is equal to b")

if ( a < b ):
        print ("a is less than b")
else:
        print ("a is not less than b") if ( a > b ):
        print ("a is greater than b")
else:
        print ("a is not greater than b")
```

```
a = 5
b = 20
if ( a <= b ):
        print ("a is either less than or equal to b")
else:
        print ("a is neither less than nor equal to b")
if ( a => b ):
        print ("a is either greater than  or equal to b")
else:
    print ("a is neither greater than  nor equal to b")
```

Output:

a is not equal to b
a is not equal to b
a is not less than b
a is greater than b
a is either less than or equal to b
b is either greater than or equal to b

Assignment Operators:

An Assignment Operator is the operator used to assign a new value to a variable. Assume A = 10 and B = 20 for the below table.

Operator	Description	Example

example :

```
a = 21
b = 10
c = 0
c = a + b
print ( c )

c += a
print ( c )

c *= a
print ( c )

c /= a
print ( c )

c  = 2
c %= a
print ( c )

c **= a
print ( c )
```

Output = 31, 52, 1092, 52.0, 2, 2097152, 99864

Logical Operators:

Operator	Description	Example
and	True if both the operands are true	X and Y
or	True if either of the operands are true	X or Y
not	True if operand is false (complements the operand)	not X

```
x = True
y = False

print('x and y is',x and y)

print('x or y is',x or y)

print('not x is',not x)
```

Membership Operators:

These Operators are used to test whether a value or a variable is found in a sequence (Lists, Tuples, Sets, Strings, Dictionaries) or not. The following are the Membership Operators:

Operator	Description	Example
in	True if value/variable is found in the sequence	5 in x
not in	True if value/variable is not found in the sequence	5 not in x

```
X = [1, 2, 3, 4]
A = 3
print(A in X)
print(A not in X)
```

Output = True

False

Identity Operators:

These Operators are used to check if two values (or variables) are located on the same part of the memory. Two variables that are equal does not imply that they are identical.
Following are the Identity Operators in Python:

Operator	Description	Example
is	True if the operands are identical	x is True
is not	True if the operands are not identical	x is not True

```
X1 = 'Welcome To NIELIT!'
X2 = 1234
Y1 = 'Welcome To NIELIT!'
Y2 = 1234
print(X1 is Y1)
print(X1 is not Y1)
print(X1 is not Y2)
print(X1 is X2)
```

Output: True
False
True
False

Bitwise Operators:

These operations directly manipulate bits. In all computers, numbers are represented with bits, a series of zeros and ones. In fact, pretty much everything in a computer is represented by bits.

OPERATOR	DESCRIPTION	SYNTAX
&	Bitwise AND	x & y
|	Bitwise OR	x | y
~	Bitwise NOT	~x
^	Bitwise XOR	x ^ y
>>	Bitwise right shift	x>>
<<	Bitwise left shift	x<<

BITWISE AND , OR OPERATOR

Bitwise AND operator: Returns 1 if both the bits are 1 else 0.

Example:

a = 10 = 1010 (Binary)
b = 4 = 0100 (Binary)

```
a & b = 1010
          &
        0100
      = 0000
      = 0 (Decimal)
```

Bitwise or operator: Returns 1 if either of the bit is 1 else 0.

Example:

a = 10 = 1010 (Binary)
b = 4 = 0100 (Binary)

```
a | b = 1010
          |
        0100
      = 1110
      = 14 (Decimal)
```

Bitwise not operator

Bitwise not operator: **Returns one's complement of the number.**

Example:

a = 10 = 1010 (Binary)

~a = ~1010

= -(1010 + 1)

= -(1011)

= -11 (Decimal)

BITWISE XOR OPERATOR

Bitwise xor operator: Returns 1 if one of the bits is 1 and the other is 0 else returns false.

Example:

a = 10 = 1010 (Binary)

b = 4 = 0100 (Binary)

a ^ b = 1010

^

0100

= 1110

= 14 (Decimal)

Shift Operators

These operators are used to shift the bits of a number left or right thereby multiplying or dividing the number by two respectively. They can be used when we have to multiply or divide a number by two.

Bitwise right shift: Shifts the bits of the number to the right and fills 0 on voids left(fills 1 in the case of a negative number) as a result.

Example:

Example 1:
a = 10 = 0000 1010 (Binary)
a >> 1 = 0000 0101 = 5

Shift Operators

Bitwise left shift: Shifts the bits of the number to the left and fills 0 on voids right as a result.

Example:

Example 1:
a = 5 = 0000 0101 (Binary)
a << 1 = 0000 1010 = 10
a << 2 = 0001 0100 = 20