UNIT IV

In Python programming, Operators in general are used to perform operations on values and variables. These are standard symbols used for the purpose of logical and arithmetic operations. In this article, we will look into different types of **Python operators.**

- OPERATORS: These are the special symbols. Eg- + , * , /, etc.
- OPERAND: It is the value on which the operator is applied.

Types of Operators in Python

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

Arithmetic Operators in Python

Python Arithmetic operators are used to perform basic mathematical operations like **addition, subtraction, multiplication**, and **division**.

In Python 3.x the result of division is a floating-point while in Python 2.x division of 2 integers was an integer. To obtain an integer result in Python 3.x floored (// integer) is used.

Operator	Description	Syntax
+	Addition: adds two operands	x + y
–	Subtraction: subtracts two	x – y

Operator	Description	Syntax
	operands	
*	Multiplication: multiplies two operands	x * y
/	Division (float): divides the first operand by the second	x / y
//	Division (floor): divides the first operand by the second	x // y
%	Modulus: returns the remainder when the first operand is divided by the second	x % y
**	Power: Returns first raised to power second	x ** y

Example of Arithmetic Operators in Python

Division Operators

Division Operators allow you to divide two numbers and return a quotient, i.e., the first number or number at the left is divided by the second number or number at the right and returns the quotient.

Comparison Operators in Python

In Python Comparison of Relational operators compares the values. It either returns **True** or **False** according to the condition.

Operator	Description	Syntax
>	Greater than: True if the left operand is greater than the right	x > y
<	Less than: True if the left	x < y

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

Logical Operators in Python

Python Logical operators perform **Logical AND**, **Logical OR**, and **Logical NOT** operations. It is used to combine conditional statements.

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

Bitwise Operators in Python

Python Bitwise operators act on bits and perform bit-by-bit operations. These are used to operate on binary numbers.

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<<

Assignment Operators in Python

Python Assignment operators are used to assign values to the variables.

Operator	Description	Syntax
=	Assign the value of the right side of the expression to the left side operand	x = y + z
+=	Add AND: Add right-side operand with left-side operand and then assign to left operand	a+=b a=a+b
-=	Subtract AND: Subtract right operand from left operand and then assign to left operand	a-=b a=a-b
*=	Multiply AND: Multiply right operand with left operand and then assign to left operand	a*=b a=a*b
/=	Divide AND: Divide left operand with right operand and then assign to left operand	a/=b a=a/b
%=	Modulus AND: Takes modulus	a%=b a=a%b

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

Identity Operators in Python

In Python, **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

Python Loops

Python has two primitive loop commands:

- while loops
- for loops

The while Loop

With the while loop we can execute a set of statements as long as a condition is true.

The while loop requires relevant variables to be ready, in this example we need to define an indexing variable, i, which we set to 1.

The break Statement

With the break statement we can stop the loop even if the while condition is true:

Example

Exit the loop when i is 3:

```
i = 1
while i < 6:
  print(i)
```

```
  if i == 3:
    break
  i += 1
```

The continue Statement

With the continue statement we can stop the current iteration, and continue with the next:

Example

Continue to the next iteration if i is 3:

```
i = 0
while i < 6:
  i += 1
  if i == 3:
    continue
  print(i)
```

The else Statement

With the else statement we can run a block of code once when the condition no longer is true:

Example

Print a message once the condition is false:

```
i = 1
while i < 6:
  print(i)
  i += 1
else:
  print("i is no longer less than 6")
```

Exercise:

Print i as long as i is less than 6.

```
i = 1
[ ] i < 6[ ]
  print(i)
  i += 1
```