8. PYTHON – OPERATORS

Table of Contents

8. PYTHON – OPERATORS

1. Operator

- ✓ An operator is a symbol that performs an operation.
- ✓ An operator acts on some variables, those variables are called as operands.
- ✓ If an operator acts on a single operand, then it is called as unary operator
- ✓ If an operator acts on 2 operands, then it is called as binary operator.
- ✓ If an operator acts on 3 operands, then it is called as ternary operator.

Program Name: operator and operands
demo1.py

```
a = 10
b = 20
c = a + b
print(c)
```

output

```
30
```

- ✓ Here a, b and c are called as operands.
- ✓ + symbol is called as operator.

2. Type of operators

- ✓ Arithmetic Operators: +, -, *, /, %, **, //
- ✓ Assignment Operator =
- ✓ Unary minus Operator -

- ✓ Relational Operator >, <, >=, <=, ==, ! =
- ✓ Logical Operators and, or, not
- ✓ Membership operators in, not in

Make a note:

- ✓ Python does not have increment and decrement operators.

1. Arithmetic Operators: (+, -, *, /, %, **, //)

- ✓ These operators will do their usual job, so please don't expect any surprises.

Assume that,

a = 13
b = 5

Operator	Meaning	Example	Result
+	Addition	a + b	18
-	Subtraction	a - b	8
*	Multiplication	a * b	65
/	Division	a / b	2.6
%	Modulus (Remainder of division)	a % b	3
**	Exponent operator (exponential power value)	a ** b	371293

//	Integer division (gives only integer quotient)	a // b	2

Make a note

- ✓ Division operator / always performs floating point arithmetic, so it returns float values.
- ✓ Floor division (//) can perform both floating point and integral as well,
  - If values are int type, then result is int type.
  - If at least one value is float type, then result is float type.

Program Name: Arithmetic Operators
demo2.py

```
a = 13
b = 5
print(a+b)
print(a-b)
print(a*b)
print(a/b)
print(a%b)
print(a**b)
print(a//b)
```

Output

```
18
8
65
2.6
3
371293
2
```

2. Assignment operator: (=, +=, -=, *=, /=, %=, **=, //=)

✓ By using these operators, we can assign values to variables.

Assume that,

a = 13

Operator	Example	Equals to	Result
=	a = 13	a = 13	13
+=	a += 6	a = a + 6	19
-=	a -= 6	a = a - 6	7

Program Name	Assignment operator demo3.py

```
a = 13
print(a)

a += 5
print(a)
```

Output

```
13
18
```

3. Unary minus operator: (-)

- ✓ Symbol of unary minus operator is **–**

Program Name	Unary minus operator demo4.py
	a = 10 print(a) print(-a)
Output	
	10 -10

4. Relational operators (>, >=, <, <=, ==, !=)

- ✓ By using these operators, we can create simple conditions.
- ✓ These operators are used to compare two values.
- ✓ These operators' returns result as Boolean type either True or False.

Assume that,

a = 13
b = 5

Operator	Example	Result
>	a > b	True
>=	a >= b	True
<	a < b	False
<=	a <= b	False
==	a == b	False
!=	a != b	True

Program Name: Relational operators
demo4.py

```
a = 13
b = 5

print(a > b)
print(a >= b)
print(a < b)
print(a <= b)
print(a == b)
print(a != b)
```

Output

```
True
True
False
False
False
True
```

5. Logical operators (and, or, not)

- ✓ In python there are three logical operators those are,
    - and
    - or
    - not

- ✓ Logical operators are useful to create compound conditions.
- ✓ Compound condition is a combination of more than one simple condition.
- ✓ Each simple condition brings the Boolean result finally the total compound condition evaluates either True or False.

For Boolean types behaviour

- ✓ and
    - If both arguments are True, then only result is True
- ✓ or
    - If at least one argument is True, then result is True
- ✓ not
    - complement

Program Name	Logical operators. demo5.py

```
a = True
b = False

print(a and a)
print(a or b)
print(not a)
```

Output

```
True
True
False
```

Program Name: Logical operators
demo7.py

```
a = 1
b = 2
c = 3
print((a>b) and (b>c))
print((a<b) and (b<c))
```

Output

```
False
True
```

6. Membership operators (in, not in)

- ✓ There are two membership operators,
  - in
  - not in

- ✓ Membership operators are useful to check whether the given value is available in sequence or not. (It may be string, list, set, tuple and dict)

The **in** operator

- ✓ **in** operator returns True, if element is found in the collection or sequences.
- ✓ **in** operator returns False, if element is not found in the collection or sequences.

The **not in** operator

- ✓ **not in** operator returns True, if an element is not found in the sequence.
- ✓ **not in** operator returns False, if an element is found in the sequence.

Program Name: in, not in operators
demo8.py

```
values = [10, 20, 30]

print(20 in values)
print(22 in values)
```

output

```
True
False
```

Program Name: in, not in operators
demo9.py

```
text = "Welcome to python programming"

print("Welcome" in text)
print("welcome" in text)
print("nireekshan" in text)
print("Daniel" not in text)
```

output

```
True
False
False
True
```