PYTHON TUTORIAL FOR BEGINNERS

Source: www.youtube.com/@RishabhMishraOfficial

Chapter - 08

Operators in Python

- What are Operators
- Types of Operators
- Operators Examples

Operators in Python

Operators in Python are **special symbols or keywords** used to perform operations on operands (variables and values).

Operators: These are the special symbols/keywords. Eg: + , * , /, etc.

Operand: It is the value on which the operator is applied.

```
# Examples
    Addition operator '+': a + b
    Equal operator '==':   a == b
    and operator 'and':    a > 10 and b < 20
```

Types of Operators

Python supports various types of operators, which can be broadly categorized as:

1. Arithmetic Operators
2. Comparison (Relational) Operators
3. Assignment Operators
4. Logical Operators
5. Bitwise Operators
6. Identity Operators
7. Membership Operators

Operators Cheat Sheet

Operator	Description
()	Parentheses
**	Exponentiation
+, -, ~	Positive, Negative, Bitwise NOT
*, /, //, %	Multiplication, Division, Floor Division, Modulus
+, -	Addition, Subtraction
==, !=, >, >=, <, <=	Comparison operators
is, is not, in, not in	Identity, **Membership** Operators
NOT, AND, OR	Logical NOT, Logical AND, Logical OR
<<, >>	Bitwise Left Shift, Bitwise Right Shift
&, ^, |	Bitwise AND, Bitwise XOR, Bitwise OR

1. Arithmetic Operators

Arithmetic operators are used with numeric values to perform mathematical operations such as addition, subtraction, multiplication, and division.

Operator	Name	Example (a = 5, b = 3)
+	Addition	a + b # o/p: 8
-	Substraction	a – b # o/p: 2
*	Multiplication	a * b # o/p: 15
/	Division	a / b # o/p: 1.6666
%	Modulus (returns remainder)	a % b # o/p: 2
//	Floor division	a // b # o/p: 1
**	Exponentiation	a ** b # o/p: 125

Precedence of **Arithmetic Operators** in Python:

P – Parentheses
E – Exponentiation
M – Multiplication
D – Division
A – Addition
S – Subtraction

2. Comparison (Relational) Operators

Comparison operators are used to compare two values and return a Boolean result (True or False).

Operator	Name	Example
==	Equal	a == b
!=	Not equal	a != b
>	Greater than	a > b
<	Less than	a < b
>=	Greater than or equal to	a >= b
<=	Less than or equal to	a <= b

3. Assignment Operators

Assignment operators are used to assign values to variables.

Operator	Eaample	Also written As
=	a = 5	a = 5
+=	a += 3	a = a + 3
-=	a -= 3	a = a - 3
*=	a *= 3	a = a * 3
/=	a /= 3	a = a / 3
%=	a %= 3	a = a % 3
//=	a //= 3	a = a // 3
**=	a **= 3	a = a ** 3
&=	a &= 3	a = a & 3 ... etc

4. Logical Operators

Logical operators are used to combine conditional statements.

Operator	Decription	Eaample
and	Returns True if both statements are true	a > 2 and a < 5
or	Returns True if one of the statements is true	a > 5 or a < 10
not	Reverse the result, returns False if the result is true	not(a > 2 and a < 10)

5. Identity & Membership Operators

Identity operators are used to compare the memory locations of two objects, not just equal but if they are the same objects.

Membership operators checks whether a given value is a member of a sequence (such as strings, lists, and tuples) or not.

Tbpe	Operator	Eaample	Also written As
Identity	is	Returns True if both variables are the same object	a is b
Identity	is not	Returns True if both variables are not the same object	a is not b
Membership	in	Returns True if a sequence with the specified value is present in the object	a in b
Membership	not in	Returns True if a sequence with the specified value is not present in the object	a not in b

6. Bitwise Operators

Bitwise operators perform operations on binary numbers.

Operator	Name	Description	Also written As
&	AND	Sets each bit to 1 if both bits are 1	a & b
|	OR	Sets each bit to 1 if one of two bits is 1	a | b
^	XOR	Sets each bit to 1 if only one of two bits is 1	a ^ b
~	NOT	Inverts all the bits	~a
<<	Zero fill left shift	Shift left by pushing zeros in from the right and let the leftmost bits fall off	a << 2
>>	Signed right shift	Shift right by pushing copies of the leftmost bit in from the left, and let the rightmost bits fall off	a >> 2

Bitwise Operators Example:

```
# Compare each bit in these numbers.
0101  (This is 5 in binary)
0011  (This is 3 in binary)
-------
0001  (This is the result of 5 & 3)

# Rules: 0 - False, 1 - True
True + True = True
True + False = False
False + False = False
```

```
Eg:1
a = 5           # 0101
b = 3           # 0011
print(a & b)
# Output: 1     # 0001

Eg:2
a = 5           # 0101
b = 8           # 1000
print(a & b)
# Output: 0     # 0000
```

Python Tutorial Playlist: **Click Here**

https://www.youtube.com/playlist?list=PLdOKnrf8EcP384Ilxra4UlK9BDJGwawg9