Operators in Python

“Operators in Python can be defined as symbols that assist us to perform certain operations. The operations can be between variable and variable, variable and value, value and value”.

Operators that Python Language supports are:

- Arithmetic Operators
- Assignment Operators
- Comparison Operators
- Logical Operators
- Identity Operators
- Membership Operators
- Bitwise Operators

We must remember most of them from our basic mathematics that we studied in school. May be except the last one, that might be new for some of us. To understand bitwise fully, we must have the basic knowledge of the conversion of decimal into binary. For example, the binary for the first five number is:

0001

0010

0011

0100

0101

And so on.

Now We will give you a theoretical understanding of each of these operators. There is no theory related difference in them and the one we studied during school time. The only difference you will see will be in the syntax i.e. how to write them for perfect execution.

Arithmetic Operators

Basic mathematical operations such as addition, multiplication, subtraction, division, etc. are performed with the help of arithmetic Operations. It contains nearly all operations that we can perform with the help of a calculator.

Operator	Name	Example
+	Addition	x + y
-	Subtraction	x - y
*	Multiplication	x * y
/	Division	x / y
%	Modulus	x % y
**	Exponentiation	x ** y
//	Floor division	x // y

Assignment Operators

The assignment operator is used to assign values to a variable. In some cases, we have to assign a variable's value to another variable, in such cases the value of the right operand is assigned to the left operand. One of the basic signs from which we can recognize an assignment operator is that it must have an equal-to(=) sign. Some commonly used assignment operators include +=, -=, /=, etc.

Operator	Example	Same As
=	x = 5	x = 5
+=	x += 3	x = x + 3
-=	x -= 3	x = x - 3
*=	x *= 3	x = x * 3
/=	x /= 3	x = x / 3
%=	x %= 3	x = x % 3
//=	x //= 3	x = x // 3
**=	x **= 3	x = x ** 3

Comparison Operators

They are also known as relational operators. They compare the values on either side of the operator and decide the relation among them.

- Less than (<) — returns true if the value on the left is less than the value on the right, otherwise it returns false.
- Greater than (>) — returns true if the value on the left is greater than the value on the right, otherwise it returns false.
- Less than or equal to (<=) — returns true if the value on the left is less than or equal to the value on the right, otherwise it returns false.
- Greater than or equal to (>=) — returns true if the value on the left is greater than or equal to the value on the right, otherwise it returns false.
- Equal to (==) — returns true if the value on the left is equal to the value on the right, otherwise it returns false.
- Not equal to (!=) — returns true if the value on the left is not equal to the value on the right, otherwise it returns false.

Operator	Name	Example
==	Equal	x == y
!=	Not equal	x != y
>	Greater than	x > y
<	Less than	x < y
>=	Greater than or equal to	x >= y
<=	Less than or equal to	x <= y

Logical Operators

Logical operators perform logical AND, OR and NOT, operations. They are usually used in conditional statements to join multiple conditions. AND, OR and NOT keywords are used to perform logical operations.

Operator	Description	Example
and	Returns True if both statements are true	x < 5 and x < 10
or	Returns True if one of the statements is true	x < 5 or x < 4
not	Reverse the result, returns False if the result is true	not(x < 5 and x < 10)

Identity Operations

Identity operator checks if two operands share the same identity or not, which means that they share the same location in memory or different. “is” and “is not” are the keywords used for identity operands.

Operator	Description	Example
is	Returns True if both variables are the same object	x is y
is not	Returns True if both variables are not the same object	x is not y

Membership Operators

Membership operand checks if the value or variable is a part of a sequence or not. The sequence could be string, list, tuple, etc. “in” and “not in” are keywords used for membership operands.

Operator	Description	Example
in	Returns True if a sequence with the specified value is present in the object	x in y
not in	Returns True if a sequence with the specified value is not present in the object	x not in y

Bitwise Operators

Bitwise operands are used to perform bit by bit operation on binary numbers. First, we have to change the format of the number from decimal to binary and then compare them using AND, OR, XOR, NOT, etc.

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

IF, ELSE and ELIF Conditions

IF, ELSE and ELIF statement can be defined as a multiway decision taken by our program due to the certain conditions in our code.

For few viewers, the term “elif” is new as they are not familiar with the word and it is also not like most of the other words such as list or loops, etc. that have the same meaning in the English language and in Python programming. In fact, in English “elif” means honest. But if you have ever done programming in any language, you must be familiar with “else-if” statement, well “elif” is just that.

Now coming towards a more formal sort of description. “If and else” are known as decision-making statements for our program. They are very similar to the decision making we apply in our everyday life that depends on certain conditions.

Our compiler will execute the if statement to check whether it is true or false now if it’s true the compiler will execute the code in the “if” section of the program and skip the bunch of code written in “elif” and “else”. But if the “if” condition is false then the compiler will move towards the elif section and keep on running the code until it finds a true statement(there could be multiple elif statements). If this does not happen then it will execute the code written in the “else” part of the program.

An “if” statement is a must because without an if we cannot apply “else” or “else-if” statement. On the other hand else or else if statement are not necessary because if we have to check between only two conditions we use only “if and else” and even though if we require code to run only when the statement returns true and do nothing if it returns false then an else statement is not required at all.

Now Let’s talk about some technical issues related to the working of decision statements:

- There is no limit to the number of conditions that we could use in our program. We can apply as many elif statements as we want but we can only use one “else” and one “if” statement.
- We can use nested if statements i.e. if statement within an if statement. It is quite helpful in many cases.
- Decision statements can be written using logical conditions which are:
    - Equal to
    - Not equal to
    - Less than
    - Greater than
    - Greater than equal to
    - Less than equal to

```
var1 = 5
var2 = 10
if var1 > var2:
    print("Var 1 is greater.")
else:
    print("Var 2 is greater.")

# --------------------------------

var1 = 5
var2 = 5
if var1 > var2:
    print("Var 1 is greater.")
elif var2 > var1:
    print("Var 2 is greater.")
else:
    print("Both Are Equal")


# -----OUTPUT-----
Var 2 is greater.
Both Are Equal
```

Using AND Operator

```
a = 99
b = 33
c = 412
if a > b and c > a:
  print("Both conditions are True")
```

Using OR Operator

```
a = 41
b = 33
c = 500
if a > b or a > c:
  print("At least one of the conditions is True")
```

Nested IF

```
x = 99

if x > 20:
  print("Above Twenty,")
  if x > 40:
    print("and also above 40!")
  else:
    print("but not above 40.")
```

PASS Statement

```
a = 11
b = 55

if b > a:
  pass
```

Questions

1. Take two numbers as input from User and print which one is greater or are they equal.
2. Take three numbers as input from User and print which one is greater or are they equal.
3. Take Salary as input from User and Update the salary of an employee
    - salary less than 10,000, 5 % increment
    - salary between 10,000 and 20, 000, 10 % increment
    - salary between 20,000 and 50,000, 15 % increment
    - salary more than 50,000, 20 % increment

HW

1. Ask the number from User and print whether the number is Odd or Even.
2. A school has following rules for grading system:
    - a. Below 25 - F
    - b. 25 to 45 - E
    - c. 45 to 50 - D
    - d. 50 to 60 - C
    - e. 60 to 80 - B
    - f. Above 80 - A

    Ask user to enter marks and print the corresponding grade.
3. A student will not be allowed to sit in exam if his/her attendance is less than 75%.
    - Take following input from user
        - Number of classes held
        - Number of classes attended.
    - Print percentage of class attended
    - Print Is student is allowed to sit in exam or not.
4. Calculate the purchase amount after deduction criteria on printed price
    - printed price between 500 and 1000, allow 5% discount
    - printed price between 1000 and 5000, allow 10% discount
    - printed price between 5000 and 10000, allow 15% discount
    - printed price more than 1000, allow 20% discount