In this article, we will learn about operators in python. Operators are extremely useful in mathematical operations in any pythonic code. It is very useful to know the proper know-how of these operators. Starting from the basics, we will first see what operators are, then their types, and the subsequent codes. So let's start.

What are Python Operators?

Operators, as the name suggests, operate the calculation, i.e., the basics of python coding. There are various operators used, and all have different and important functions to write any code.

When learned in-depth, the operator gives a correct understanding of what a python code is trying to calculate and this also helps the coder in predicting the result of the code, even before the code is executed.

Types of Operators in Python

1. Comparison Operators in Python

These operators are for comparing any two significant values in a python code. They take two values in them and then compare them with each other. And accordingly, they display the result.

Operator	Meaning
>	Returns True if x is greater than y
<	Returns True if x is less than y
==	Returns True if both x,y are equal
!=	Returns True if x and y not equal
>=	Returns True if x is greater than or equal to y
<=	Returns True if x is less than or equal to the y

Example of Comparison Operator in Python:

```
a=int(input("enter number"))
b=int(input("enter number"))
if a>b:
print("greater")
else:
print("smaller")
```

Output:

```
enter number 56
enter number 78
smaller
```

2. Logical Operators in Python

These operators work on logic, i.e., they check the conditions and give a straight logical output to it. A logical operator is a must use in a one-way code, where the coder has no idea of what the user is going to give as input.

Operator	Meaning
and	Returns True if both the x and y are equal
or	Returns True if either of one condition is true
not	True if not the whole operator is false

Example of Logical Operator in Python

```
a=int(input("enter number"))
b=int(input("enter number"))
if a and b==12:
print("true")
else:
print("false")
```

Output:

```
enter number 12
enter number 12
true
```

3. Arithmetic Operators in Python

These are the basic mathematical operators. They perform basic functions like addition, subtraction, etc in any code. They are considered the building blocks of the code.

Operator	Meaning	Exam
+	Addition	a+b
–	subtraction	a-b
*	multiplication	a*b
/	division	a/b
%	modulus	a%b
**	Exponential	a**
//	Floor division	a//

Example of Arithmetic Operators in Python

```
a=int(input("enter number"))
b=int(input("enter number"))
c=a+b
print(c,"the sum of the numbers")
```

Output:

```
enter number 56
enter number 78
134 the sum of the numbers
```

4. Identity Operators in Python

These are the gaming defined operators in any code which helps the coder to know the identity of the functional group. These operators, by default, return only true or false. However, they may accept new output as generated by the coder.

Operator	Meaning
is	Returns true if all conditions are correct
Is not	Returns false on one condition being incorrect

Example of Identity Operators in Python

```
a=int(input("enter number"))
b=int(input("enter number"))
if a is b:
print("both are equal")
```

Output:

```
enter number 56
enter number 56
both are equal
```

5. Bitwise Operators in Python

They act on alphanumeric operators in the form of bits. They only give raw output in the form of 0 and 1. They are easily interpreted by the code as they work on the language of on and off, i.e., 0 and 1 only.

Operator	Meaning	Example
&	Bitwise AND	x & y = 0 (0000 0000)
|	Bitwise OR	x | y = 14 (0000 1110)
~	Bitwise NOT	~x = -11 (1111 0101)
^	Bitwise XOR	x ^ y = 14 (0000 1110)

>>	Bitwise right shift	x >> 2 = 2 (0000 0010)
<<	Bitwise left shift	x << 2 = 40 (0010 1000

Example of Bitwise Operators in Python:

```
a=int(input("enter number"))
b=int(input("enter number"))
c=a^b
print(c)
```

Output:

```
enter number 67
enter number 78
13
```

6. Assignment Operators in Python

They are the mathematical operators, which perform two functions at the same time. They easily interpret the difficult codes by breaking them in small lines.

Operator	Example	Equivalent to
=	X = 1	X = 1
+=	X += 1	X = x + 1
-=	x -= 1	x = x – 1
*=	x *= 1	x = x * 1
/=	x /= 1	x = x / 1
%=	x %= 1	x = x % 1
//=	x /= 1	x = x / 1
**=	x **= 1	x = x ** 1
&=	x &= 1	x = x & 1
|=	x |= 1	x = x | 1
^=	x ^= 1	x = x ^ 1
>>=	x >>= 1	x = x >> 1

Example of Assignment Operators in Python:

```
a=int(input("enter number"))
b=int(input("enter number"))
a*=3
print(a)
print(b)
```

Output:

```
enter number 900
enter number 87
2700
87
```

7. Special Operators in Python

They are generally identity operators or membership operators. They are termed special operators because they are generally not much in use for any simple python code.

Operator	Meaning
is	True if x and y are same
Is not	True if x and y are not same or comparable

Example of Special Operators in Python

```
a=int(input("enter number"))
b=int(input("enter number"))
if a is not b:
print("true")
else:
print("false")
```

Output:

```
enter number 78
enter number 89
true
```

8. Membership Operators in Python

These are used to test whether a value or variable is found in a sequence or not. These operators are mostly used for the Fibonacci series or the stepwise calculations in sequential data formats.

Operator	Meaning
in	True if values are in the order of list.
Not in	True if value/variable is not in the order of list

Example of Membership Operators in Python:

```
a=int(input("enter number"))
b=[78,90,8]
if a in b:
print("true")
else:
print("false")
```

Output:

```
enter number 8
true
```

Operator's Precedence and Associativity in Python

This means that though python serves a vast library of operators, sometimes we use many operators together. Then how will we identify which operator functions first? This can be done if we know the priority that python has given to each operator.

A table regarding the same has been listed below:

Operator	Meaning	As
( )	Parentheses	le
**	Exponent	r
* / %	Multiplication/division/modulus	le
+ -	Addition/subtraction	le
<< >>	Bitwise shift left, Bitwise shift right	le
< <=	Relational less than/less than or equal to	le
> >=	Relational greater than/greater than or equal to	A
== !=	Relational is equal to/is not equal to	le

Example of Operators Precedence in Python:

```
a=int(input("enter number"))
b=int(input("enter number"))
d=int(input("enter number"))
p=int(input("enter number"))
c=a+b/d%p
print(c)
```

Output:

```
enter number 56
enter number 78
enter number 89
enter number 45
56.87640449438202
```

Conclusion

In this article, we learned where and how to use different types of operators in python. It is actually very important to know the correct

knowledge of any pythonic operator. Practice codes and you'll get your hands set on it soon. Happy pythonning!