Operator

1. Arithmetic Operators

Used for mathematical operations.

1. Addition(+)
2. sutraction(-)
3. Multiplication(*)
4. Division(/)
5. Floor Division(//)
6. Modulus(%)
7. Exponentiation(**)

In [1]:
```
8+5
```

Out[1]: 13

In [2]:
```
8-5
```

Out[2]: 3

In [3]:
```
8*5
```

Out[3]: 40

In [4]:
```
8/5
```

Out[4]: 1.6

In [5]:
```
8//5
```

Out[5]: 1

In [6]:
```
8%5
```

Out[6]: 3

In [7]:
```
8**5
```

Out[7]: 32768

In [8]:
```
a=10
b=15
print(a+b)
print(a-b)
print(a*b)
print(a/b)
print(a//b)
```

```
print(a%b)
print(a**b)
```

```
25
-5
150
0.6666666666666666
0
10
1000000000000000
```

2. Comparison (Relational) Operators

```
Compare two values and return True or False.
```

1. Equal to (==) 2. Not equal to(!=) 3. Greater than(<) 4. Less than(>) 5. Greater than or equal to(<=) 6. Less than or equal to(>=)

In [9]:
```
a=15
b=20
print(a==b)
print(a!=b)
print(a<b)
print(a>b)
print(a<=b)
print(a>=b)
```

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

3.Logical Operators

```
Used to combine conditional statements.
1. AND(*)
2. OR(+)
3. NOT(opp)
```

AND

In [21]:
```
a=True
b=False
a&b
a and b
```

Out[21]: False

In [22]:
```
a=0
b=1
a&b
```

Out[22]: 0

In [23]:
```
a=1
b=1
a&b
```

Out[23]: 1

OR

In [27]:
```
x=1
y=0
x|y
x or y
```

Out[27]: 1

In [28]:
```
x=1
y=1
x|y
x or y
```

Out[28]: 1

In [29]:
```
x=0
y=0
x|y
x or y
```

Out[29]: 0

NOT

In [32]:
```
x=1
x=True
not x
```

Out[32]: False

In [33]:
```
x=0
x=False
not x
```

Out[33]: True

4. Assignment Operators

Assign values to variables (sometimes with operations).

1. = x=10

2. += x+=10--> x=x+10
3. -= x-=10--> x=x-10
4. = *x*=10--> x=x*10
5. /= x/=10--> x=x/10
6. //= x//=10--> x=x//10
7. %= x%=10--> x=x%10
8. **= x**10--> x=x**10

In [ ]:
```
x=5
```

In [4]:
```
x=5
x+=10
x
```

Out[4]: 15

In [6]:
```
x-=10
x
```

Out[6]: -5

In [8]:
```
x*=10
x
```

Out[8]: -500

In [10]:
```
x/=10
x
```

Out[10]: -5.0

In [11]:
```
x//=10
x
```

Out[11]: -1.0

In [12]:
```
x%=10
x
```

Out[12]: 9.0

In [13]:
```
x**=10
x
```

Out[13]: 3486784401.0

5. Unary Operator

In [14]:
```
n=7 #negetion
n
```

Out[14]: 7

In [17]:
```
m=7
m=-(m)
m
```

Out[17]: -7

6. Bitwise Operator

1. Bitwise AND(&)
2. Bitwise OR(|)
3. Bitwise XOR(^)
4. Bitwise NOT(~)/ Complement
5. Left shift(<<)
6. Right shift(>>)

Bitwise AND(&)

1 if both the bits are 1 else 0.

In [18]:
```
5&6
```

Out[18]: 4

In [20]:
```
12&8
```

Out[20]: 8

Bitwise OR(|)

1 if atleast 1 bit is 1.

In [24]:
```
45|5
```

Out[24]: 45

In [25]:
```
12|4
```

Out[25]: 12

Bitwise not/Complement(~)

In [26]:
```
~20
```

Out[26]: -21

In [27]:
```
~-20
```

Out[27]: 19

In [28]:
```
~-5
```

Out[28]: 4

Bitwise XOR(^)

1 if bits are different else 0.

In [35]:
```
print(bin(15))
print(bin(5))
```

```
0b1111
0b101
```

In [31]:
```
15^5
```

Out[31]: 10

In [36]:
```
print(bin(20))
print(bin(8))
```

```
0b10100
0b1000
```

In [32]:
```
 20^8
```

Out[32]: 28

Left Shift(<<)

Shifts bits left, fills with 0

In [38]:
```
print(bin(5))
```

```
0b101
```

In [39]:
```
print(bin(10))
```

```
0b1010
```

In [37]:
```
5<<1
```

Out[37]: 10

In [40]:
```
print(bin(12))
```

```
0b1100
```

In [41]:
```
12<<1
```

Out[41]: 24

In [42]:
```
print(bin(24))
```

```
0b11000
```

In [43]:
```
12<<2
```

Out[43]: 48

In [44]:
```
print(bin(48))
```

```
0b110000
```

Right shift(>>)

Shifts bits right, drops last bits

In [47]:
```
print(bin(17))
```

```
0b10001
```

In [49]:
```
17>>1
```

Out[49]: 8

In [50]:
```
print(bin(8))
```

```
0b1000
```

In [51]:
```
17>>2
```

Out[51]: 4

In [52]:
```
print(bin(2))
```

```
0b10
```

Operator Completed