Operators in Python

- **Arithmetic**: `+` , `-` , `*` , `/` , `%` , `**` , `//`
- **Comparison**: `==` , `!=` , `>` , `<` , `>=` , `<=`
- **Logical**: `and` , `or` , `not`
- **Bitwise**: `&` , `|` , `^` , `~` , `<<` , `>>`
- **Assignment**: `=` , `+=` , `-=` , `*=` , `/=` , `%=` , `**=` , `//=`
- **Identity**: `is` , `is not`
- **Membership**: `in` , `not in`

Arithmetic Operators

In [14]:
```
x = 4
y = 3
print(x + y)
```

```
7
```

In [2]:
```
print(x - y)
```

```
1
```

In [3]:
```
print(x * y)
```

```
12
```

In [6]:
```
print(x / y)
```

```
1.3333333333333333
```

In [7]:
```
print(x % y)
```

```
1
```

In [8]:
```
print( x ** y)
```

```
64
```

In [9]:
```
print(x // y)
```

```
1
```

Comparison Operators

In [10]:
```
print(x > y) # x = 4 , y = 3
```

```
True
```

In [11]:
```
print(x < y) # x = 4 , y = 3
```

```
False
```

In [13]:
```
print(x >= y)  # x = 4 , y = 3
```

```
True
```

In [15]:
```
print(x <= y) # x = 4 , y = 3
```

```
False
```

In [16]:
```
print(x == y) # x = 4 , y = 3
```

```
False
```

In [17]:
```
print(x != y) # x = 4 , y = 3
```

```
True
```

Logical Operators

In [18]:
```
x = False   # ------ 0
y = True    #------- 1
print( x or y)
```

```
True
```

In [19]:
```
print(x and y)
```

```
False
```

In [20]:
```
print(not x)
print(not y)
```

```
True
False
```

Bitwise Operators

In [1]:
```
x = 2
y = 3
```

In [16]:
```
print(x & y) # AND

# in binary
# 2 ----> 010
# 3 ----> 110
#         -----
#          010
```

2

In [17]:
```
print(x | y) # OR
# in binary
# 2 ----> 010
# 3 ----> 110
#         -----
#          110
```

3

In [18]:
```
print(2 ^ 3) # XOR

'''Convert the numbers to binary:

Perform the XOR operation:
  10  (binary for 2)
^ 11  (binary for 3)
------
  01  (binary result)

XOR (exclusive OR) compares each bit of two numbers. The rule is:
If the bits are the same, the result is 0.
If the bits are different, the result is 1'''
```

1

Out[18]:
```
'Convert the numbers to binary:\n\nPerform the XOR operation:\n  10  (binary
for 2)\n^ 11  (binary for 3)\n------\n  01  (binary result)\n\nXOR (exclusiv
e OR) compares each bit of two numbers. The rule is:\nIf the bits are the sa
me, the result is 0.\nIf the bits are different, the result is 1'
```

In [5]:
```
print(x >> 2) # Right Shift

# The right shift effectively divides the number by 2**n, where n is the numbe
# For x >> 2, it is equivalent to dividing x by 2**2=4 and taking the integer
```

0

In [4]:
```
print(y << 3) # Left Shift
# The left shift operator effectively multiplies the number by 2**n ,where n i

# Result=3×2**3 = 3×8 = 24
```

24

In [5]:
```
print(~ x) # NOT

# For any number x, the bitwise NOT result is:
# ~x=-(x+1)
```

```
-3
```

Assignment Operators

In [5]:
```
a = 6
print(a)
```

```
6
```

In [6]:
```
a += 3   # a = a + 3
print(a)  # a = 6 + 3 = 9
```

```
9
```

In [7]:
```
a -= 3   # a = a - 3
print(a) # a = 9 - 3 = 6
```

```
6
```

In [8]:
```
a *= 3   # a = a * 3
print(a) # a = 6 * 3 = 18
```

```
18
```

In [9]:
```
a /= 3   # a = a / 3
print(a) # a = 18 / 3 = 6
```

```
6.0
```

In [10]:
```
a++
++a
```

Identity Operators

In [12]:
```
a = 3
b = 3
print(a is b)
# check wether they are on same memory location
```

```
True
```

In [13]:
```
a = "Hello"
b = "Hello"
print(a is b)
```

```
True
```

In [14]:
```
a = [1, 2, 3]
b = [1, 2, 3]
print(a is b)
```

```
False
```

In [15]:
```
a = "Hello-World"
b = "Hello-World"
print(a is b)
print(a is not b)
```

```
False
True
```

Membership Operators

in & not in

In [26]:
```
x = "Delhi"
print("D" in x)
print("D" not in x)
```

```
True
False
```

In [54]:
```
x = [1, 2, 3]
print(1 in x)
print(5 in x)
```

```
True
False
```

In [20]:
```
x = (1, 2, 3)
print(1 in x)
print(4 in x)
```

```
True
False
```

In [25]:
```
# Sum of digits of a 3-digit number
number = int(input('Enter a 3 digit number: '))

# Units digit
a = number % 10   # a = 3
number //= 10    # 123//10 = 12

# Tens digit
b = number % 10   # b = 2
number //= 10     #  12//10 = 1

# Hundreds digit
c = number % 10

print(a + b + c)
```

```
Enter a 3 digit number: 253
10
```

In [16]:

Out[16]: '0b1100100'

In [ ]: