Operators & Strings

🎯 Session Objectives

- 🔧 Understand what operators are and why they are used
- 🧮 Explore different types of operators in Python
- 🔢 Learn about operator precedence and order of execution
- ⚠️ Understand constraints in programming
- 🔤 Understand string indexing and slicing
- 🔧 Explore common string methods and operations

```
  1010
^ 0110
------
  1100
------
```

bit1	bit2	Result
1	1	0
1	0	1
0	1	1
0	0	0

'XOR ^'

```
= 1 * 2^3 + 1 * 2^2 + 0 * 2^1 + 0 * 2^0
= 8+4+0+0 = 12
```

NOT ~

~10 = -11

```
00001010
```

Flip the bits

```
11110101
```

1's Complement =
+1
================
2's Complement

leftmost bit [direction]
= 1 [negative]
= 0 [positive]

$$\begin{array}{r} 1010 \\ + \quad \textcircled{1} \\ \hline 1011 \end{array} \xrightarrow{} \boxed{-11}$$

$$\boxed{\sim X = -(X+1)}^{\text{NOT}}$$

$\boxed{X = -17}$ $= -(-17+1) = -(-16) = 16$

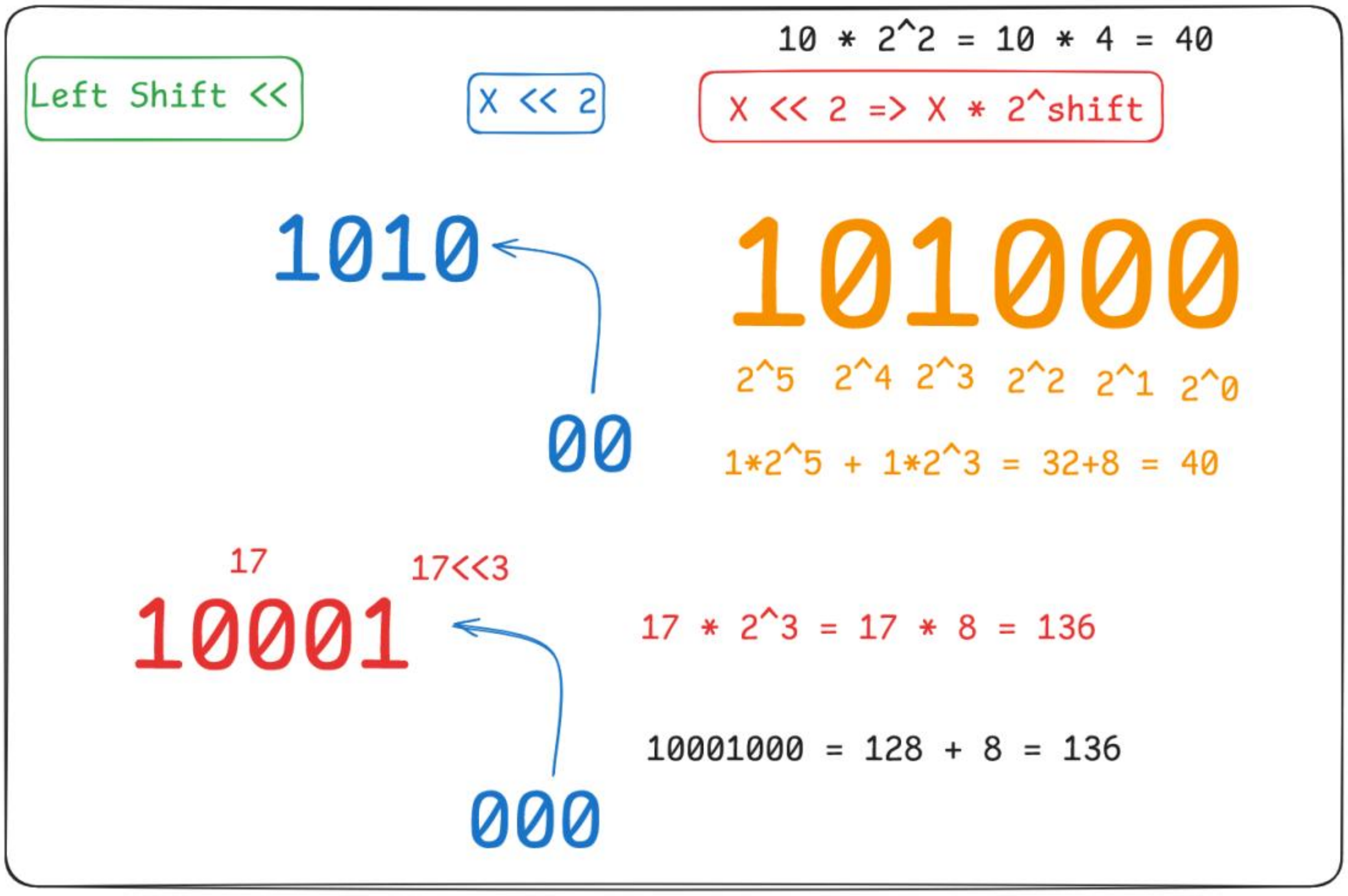
2	17
2	8-1
2	4-0
2	2-0
2	1-0
	0-1

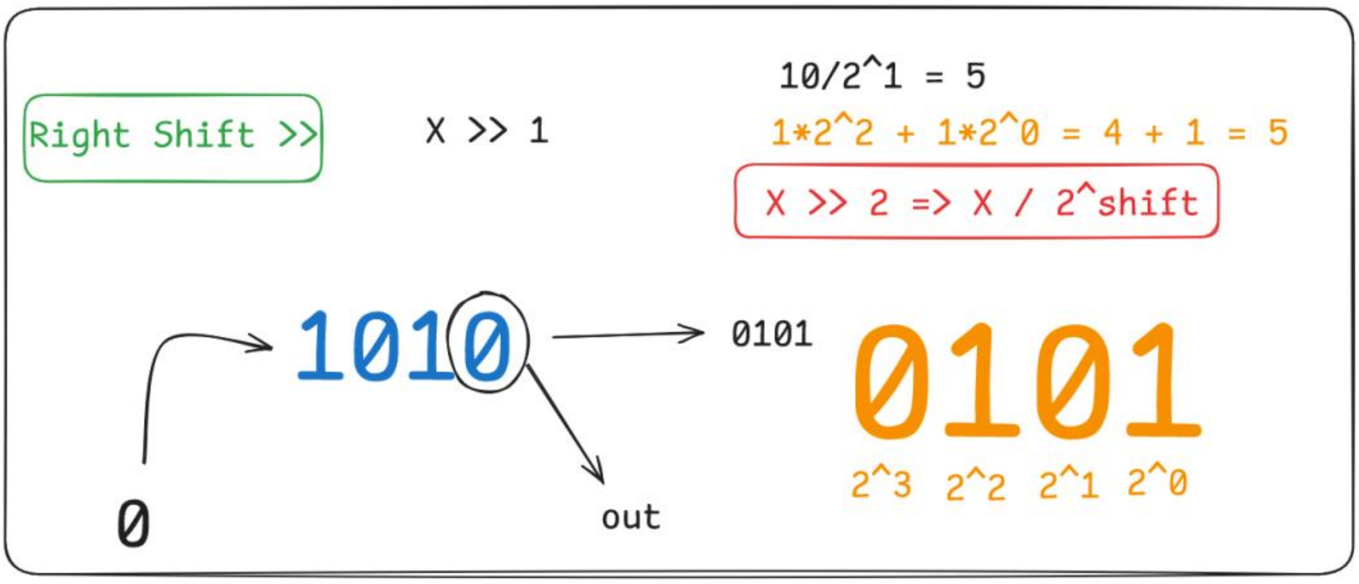
000-10001

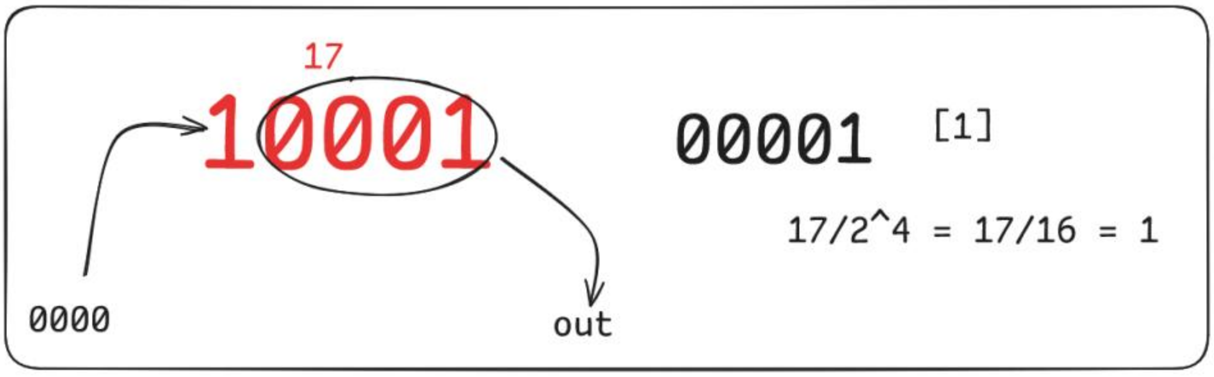
$\textcircled{1}$11-01110 [1's Complement]

-ve

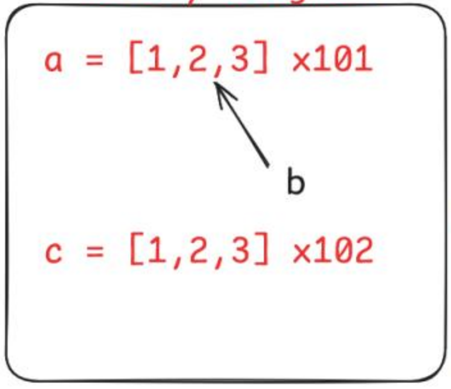
$$\begin{array}{r} 10001 \\ +1 \\ \hline 10000 \end{array} \quad \boxed{16}$$


10 * 2^2 = 10 * 4 = 40
Left Shift <<
X << 2
X << 2 => X * 2^shift
1010
00
101000
2^5 2^4 2^3 2^2 2^1 2^0
1*2^5 + 1*2^3 = 32+8 = 40
17
17<<3
10001
000
17 * 2^3 = 17 * 8 = 136
10001000 = 128 + 8 = 136


10/2^1 = 5
Right Shift >>
X >> 1
1*2^2 + 1*2^0 = 4 + 1 = 5
X >> 2 => X / 2^shift
1010
0101
0101
2^3 2^2 2^1 2^0
0
out

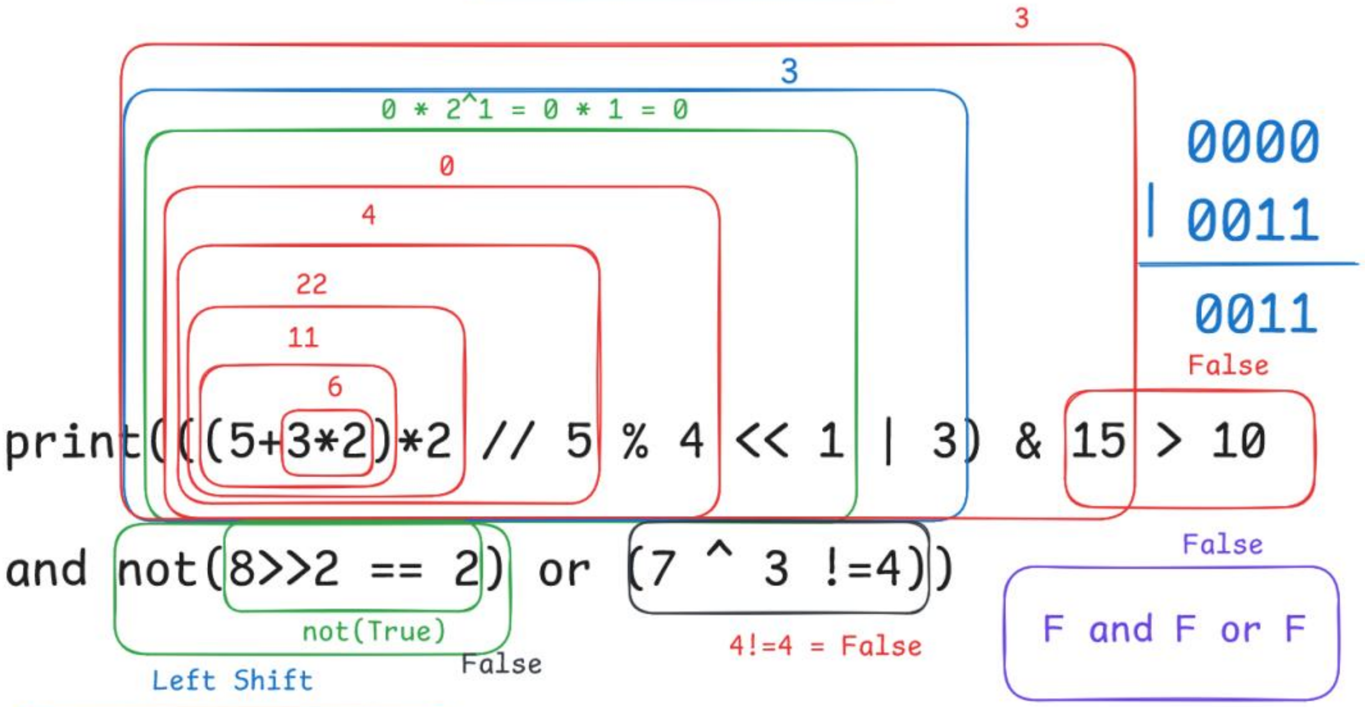


Memory Diagram

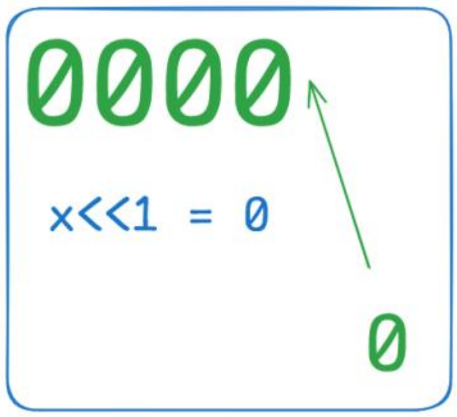


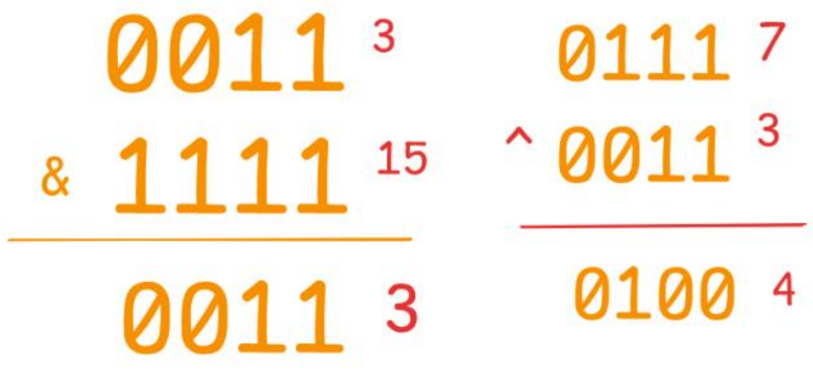
```
a = [1,2,3]
b = a
c = [1,2,3]
```

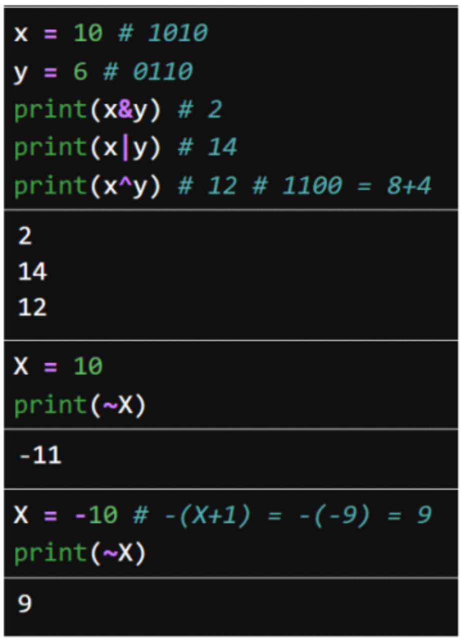
```
a==b[Data Check] [True]
a is b [address] [True]
a==c[Data Check] [True]
a is c [Address] [False]
```



```
print(((5+3*2)*2 // 5 % 4 << 1 | 3) & 15 > 10
and not(8>>2 == 2) or (7 ^ 3 !=4))
```







```
x = 10 # 1010
y = 6 # 0110
print(x&y) # 2
print(x|y) # 14
print(x^y) # 12 # 1100 = 8+4
```

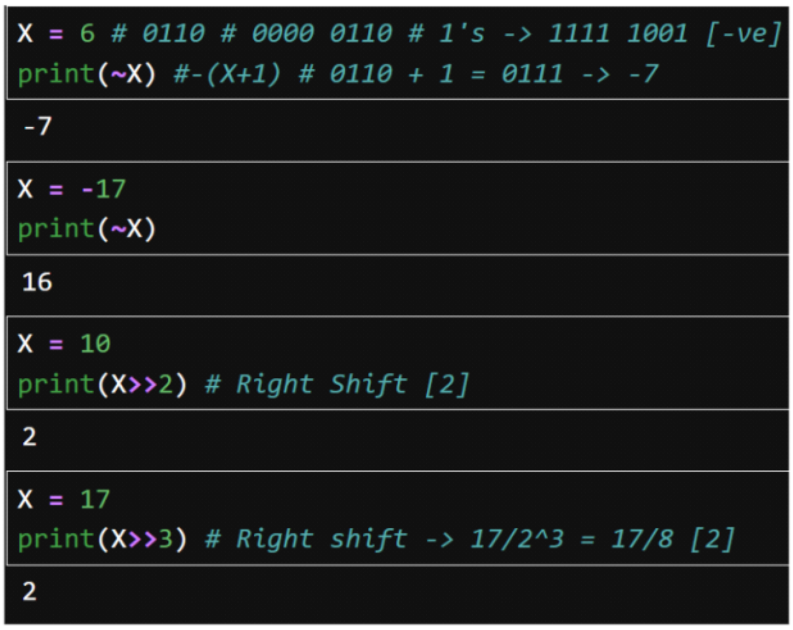
```
2
14
12
```

```
X = 10
print(~X)
```

```
-11
```

```
X = -10 # -(X+1) = -(-9) = 9
print(~X)
```

```
9
```



```
X = 6 # 0110 # 0000 0110 # 1's -> 1111 1001 [-ve]
print(~X) #-(X+1) # 0110 + 1 = 0111 -> -7
```

```
-7
```

```
X = -17
print(~X)
```

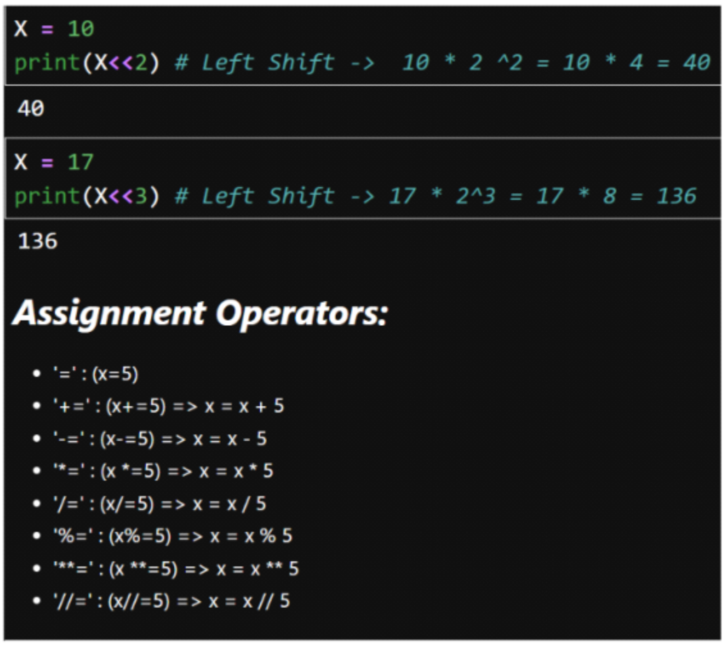
```
16
```

```
X = 10
print(X>>2) # Right Shift [2]
```

```
2
```

```
X = 17
print(X>>3) # Right shift -> 17/2^3 = 17/8 [2]
```

```
2
```



```
X = 10
print(X<<2) # Left Shift ->  10 * 2 ^2 = 10 * 4 = 40
```

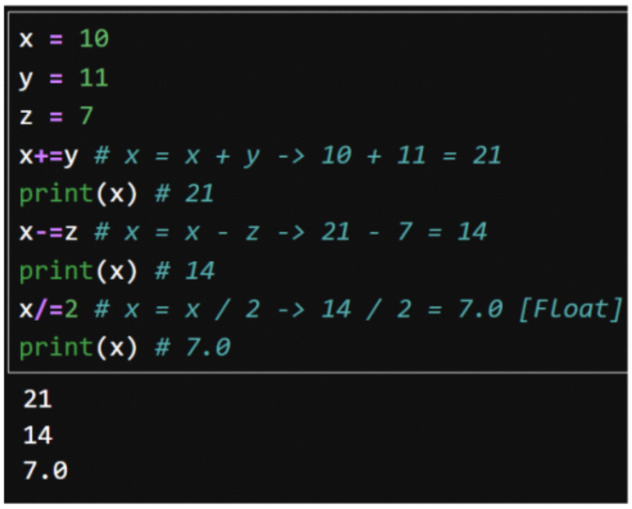
```
40
```

```
X = 17
print(X<<3) # Left Shift -> 17 * 2^3 = 17 * 8 = 136
```

```
136
```

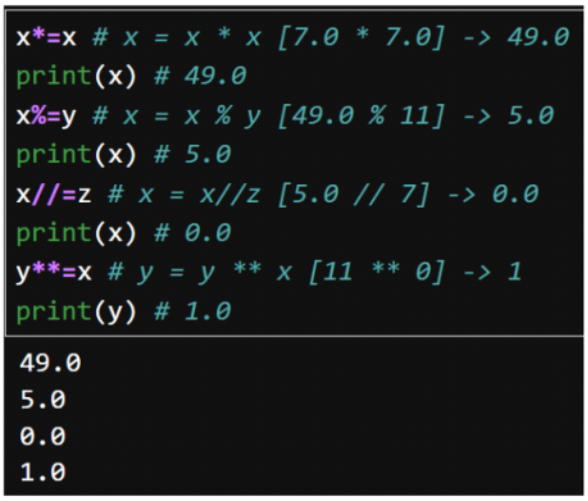
Assignment Operators:

- '=' : (x=5)
- '+=' : (x+=5) => x = x + 5
- '-=' : (x-=5) => x = x - 5
- '*=' : (x *=5) => x = x * 5
- '/=' : (x/=5) => x = x / 5
- '%=' : (x%=5) => x = x % 5
- '**=' : (x **=5) => x = x ** 5
- '//=' : (x//=5) => x = x // 5



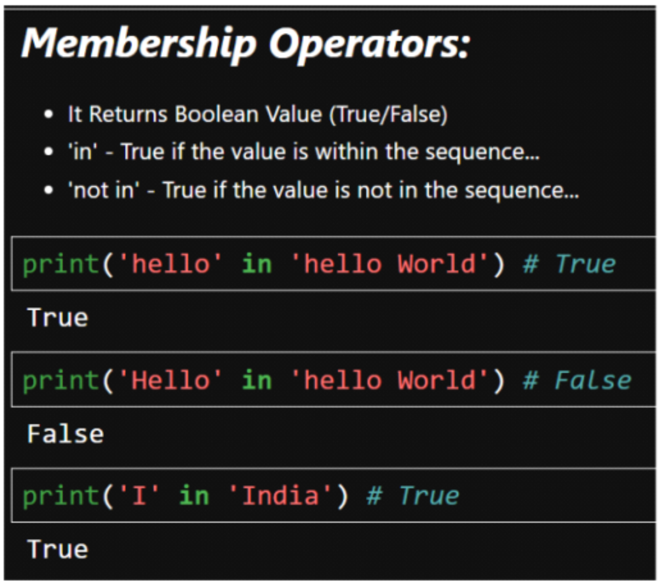
```
x = 10
y = 11
z = 7
x+=y # x = x + y -> 10 + 11 = 21
print(x) # 21
x-=z # x = x - z -> 21 - 7 = 14
print(x) # 14
x/=2 # x = x / 2 -> 14 / 2 = 7.0 [Float]
print(x) # 7.0
```

```
21
14
7.0
```



```
x*=x # x = x * x [7.0 * 7.0] -> 49.0
print(x) # 49.0
x%=y # x = x % y [49.0 % 11] -> 5.0
print(x) # 5.0
x//=z # x = x//z [5.0 // 7] -> 0.0
print(x) # 0.0
y**=x # y = y ** x [11 ** 0] -> 1
print(y) # 1.0
```

```
49.0
5.0
0.0
1.0
```



Membership Operators:

- It Returns Boolean Value (True/False)
- 'in' - True if the value is within the sequence...
- 'not in' - True if the value is not in the sequence...

```python
print('hello' in 'hello World') # True
```

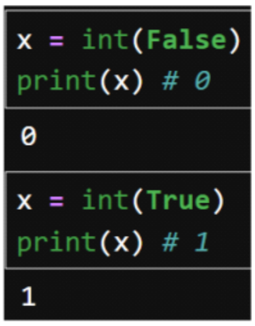
```
True
```

```python
print('Hello' in 'hello World') # False
```

```
False
```

```python
print('I' in 'India') # True
```

```
True
```

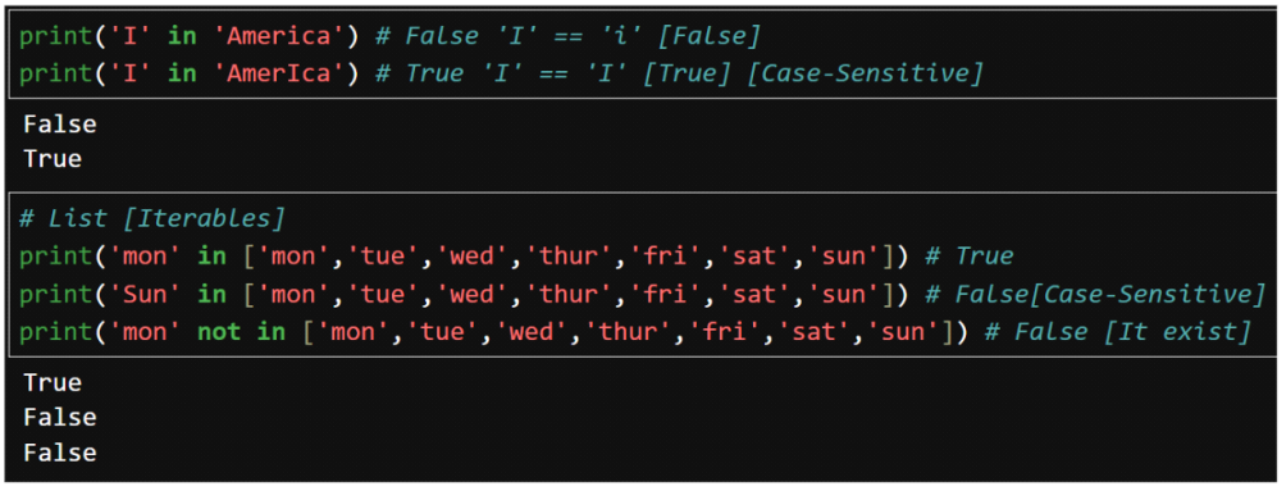


```python
x = int(False)
print(x) # 0
```

```
0
```

```python
x = int(True)
print(x) # 1
```

```
1
```

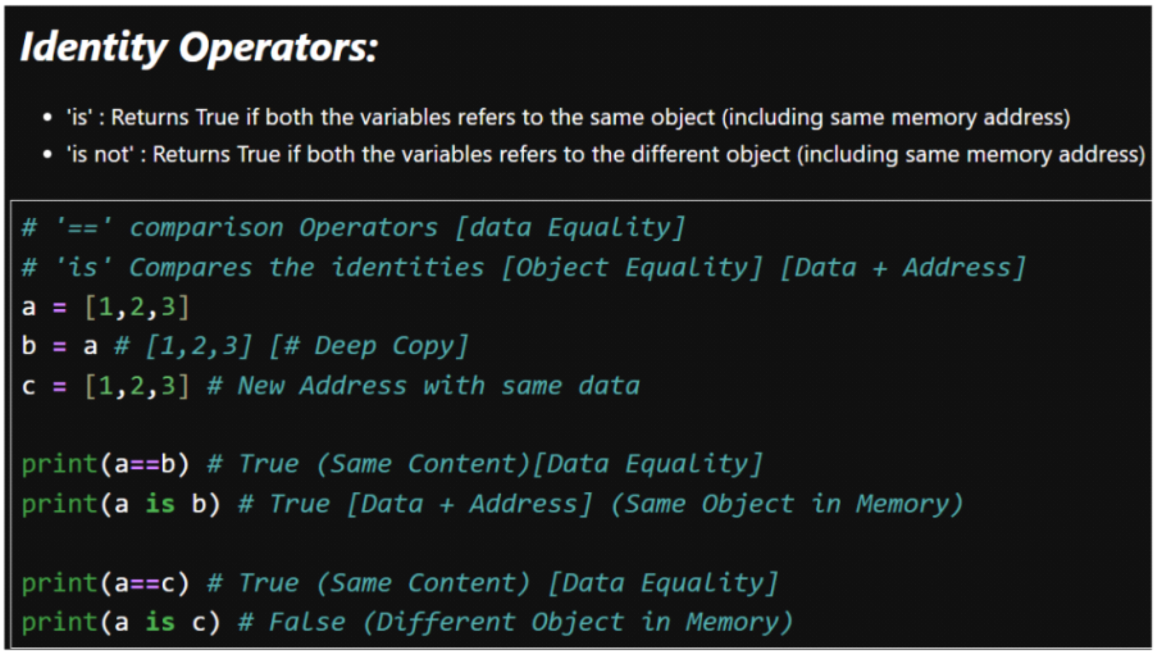


```python
print('I' in 'America') # False 'I' == 'i' [False]
print('I' in 'AmerIca') # True 'I' == 'I' [True] [Case-Sensitive]
```

```
False
True
```

```python
# List [Iterables]
print('mon' in ['mon','tue','wed','thur','fri','sat','sun']) # True
print('Sun' in ['mon','tue','wed','thur','fri','sat','sun']) # False[Case-Sensitive]
print('mon' not in ['mon','tue','wed','thur','fri','sat','sun']) # False [It exist]
```

```
True
False
False
```



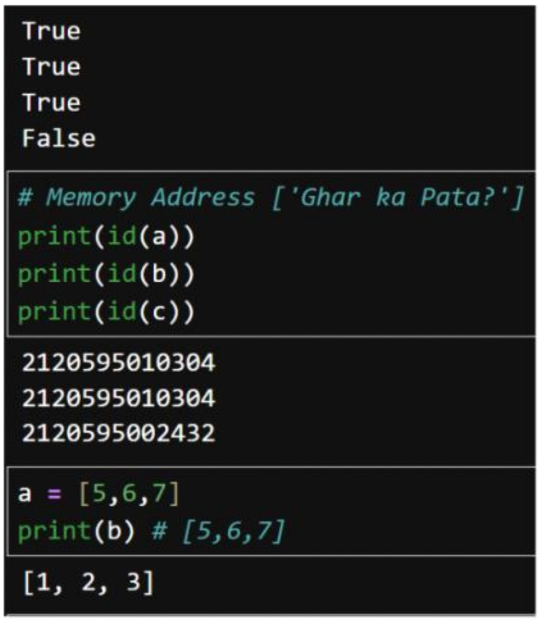
Identity Operators:

- 'is' : Returns True if both the variables refers to the same object (including same memory address)
- 'is not' : Returns True if both the variables refers to the different object (including same memory address)

```python
# '==' comparison Operators [data Equality]
# 'is' Compares the identities [Object Equality] [Data + Address]
a = [1,2,3]
b = a # [1,2,3] [# Deep Copy]
c = [1,2,3] # New Address with same data

print(a==b) # True (Same Content)[Data Equality]
print(a is b) # True [Data + Address] (Same Object in Memory)

print(a==c) # True (Same Content) [Data Equality]
print(a is c) # False (Different Object in Memory)
```



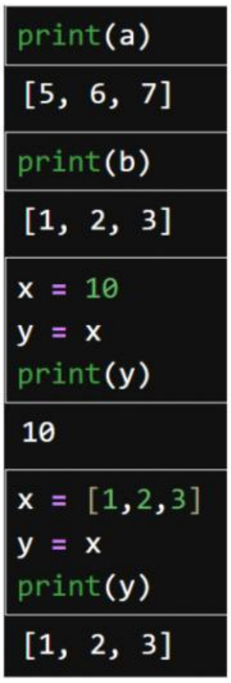
```
True
True
True
False
```

```python
# Memory Address ['Ghar ka Pata?']
print(id(a))
print(id(b))
print(id(c))
```

```
2120595010304
2120595010304
2120595002432
```

```python
a = [5,6,7]
print(b) # [5,6,7]
```

```
[1, 2, 3]
```



```python
print(a)
```

```
[5, 6, 7]
```

```python
print(b)
```

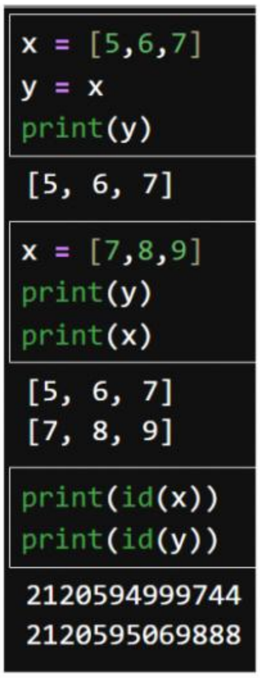
```
[1, 2, 3]
```

```python
x = 10
y = x
print(y)
```

```
10
```

```python
x = [1,2,3]
y = x
print(y)
```

```
[1, 2, 3]
```



```python
x = [5,6,7]
y = x
print(y)
```

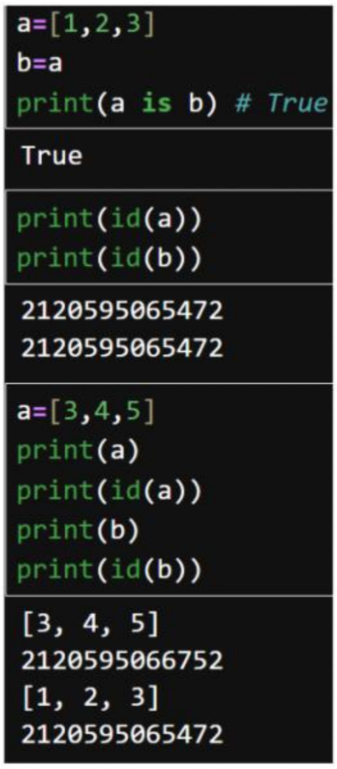
```
[5, 6, 7]
```

```python
x = [7,8,9]
print(y)
print(x)
```

```
[5, 6, 7]
[7, 8, 9]
```

```python
print(id(x))
print(id(y))
```

```
2120594999744
2120595069888
```



```python
a=[1,2,3]
b=a
print(a is b) # True
```

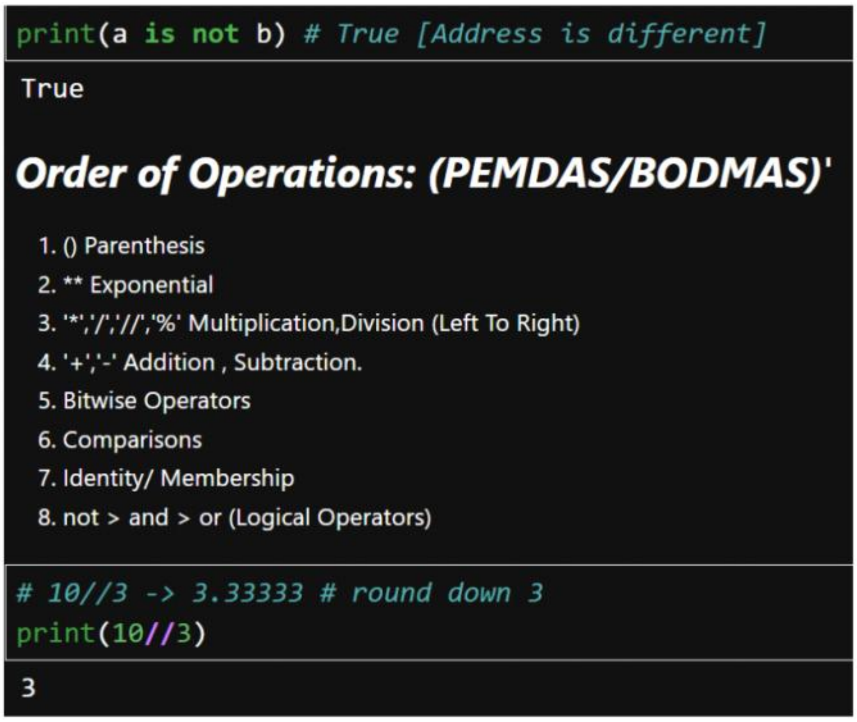
```
True
```

```python
print(id(a))
print(id(b))
```

```
2120595065472
2120595065472
```

```python
a=[3,4,5]
print(a)
print(id(a))
print(b)
print(id(b))
```

```
[3, 4, 5]
2120595066752
[1, 2, 3]
2120595065472
```



```python
print(a is not b) # True [Address is different]
```

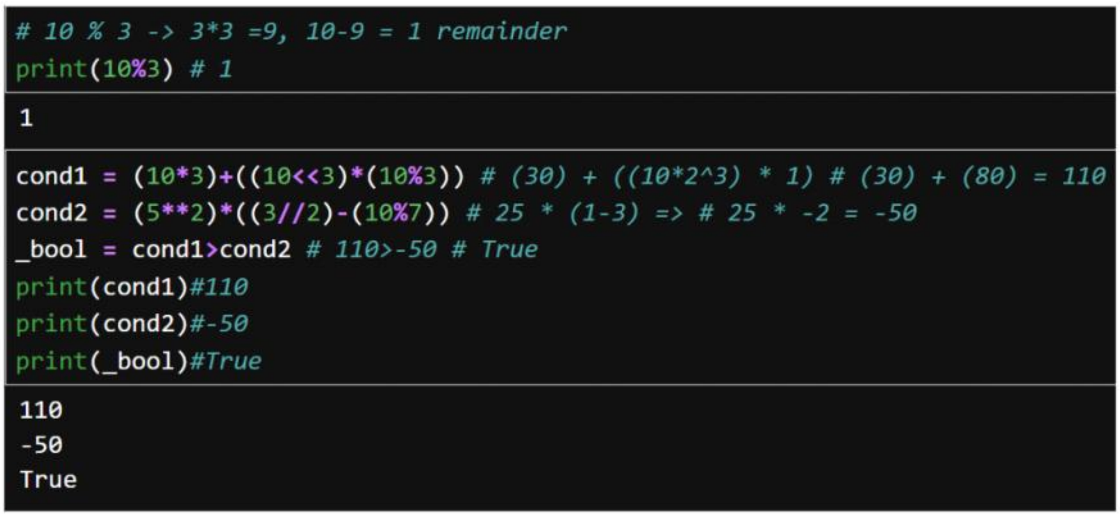
```
True
```

Order of Operations: (PEMDAS/BODMAS)'

1. () Parenthesis
2. ** Exponential
3. '*','/','//','%' Multiplication,Division (Left To Right)
4. '+','-' Addition , Subtraction.
5. Bitwise Operators
6. Comparisons
7. Identity/ Membership
8. not > and > or (Logical Operators)

```python
# 10//3 -> 3.33333 # round down 3
print(10//3)
```

```
3
```

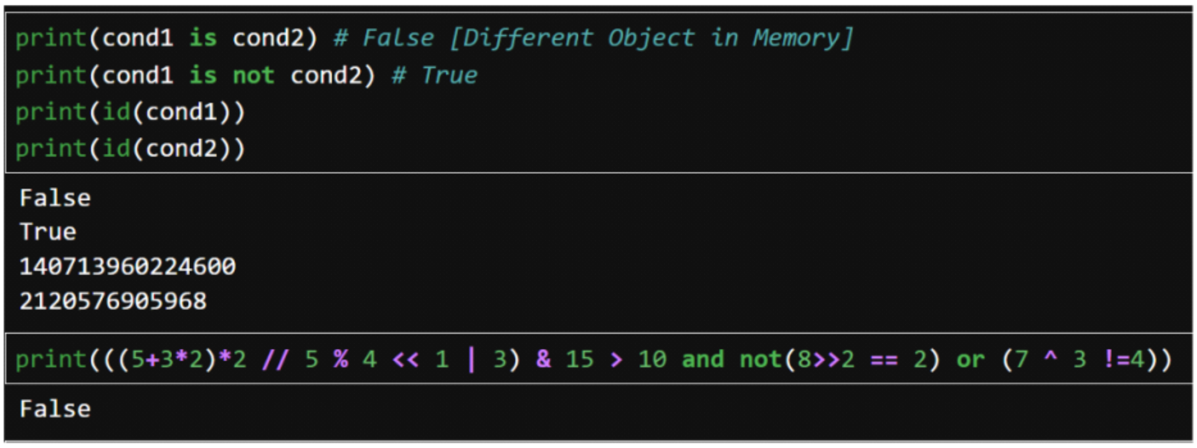


```python
# 10 % 3 -> 3*3 =9, 10-9 = 1 remainder
print(10%3) # 1
```

```
1
```

```python
cond1 = (10*3)+((10<<3)*(10%3)) # (30) + ((10*2^3) * 1) # (30) + (80) = 110
cond2 = (5**2)*((3//2)-(10%7)) # 25 * (1-3) => # 25 * -2 = -50
_bool = cond1>cond2 # 110>-50 # True
print(cond1)#110
print(cond2)#-50
print(_bool)#True
```

```
110
-50
True
```



```python
print(cond1 is cond2) # False [Different Object in Memory]
print(cond1 is not cond2) # True
print(id(cond1))
print(id(cond2))
```

```
False
True
140713960224600
2120576905968
```

```python
print(((5+3*2)*2 // 5 % 4 << 1 | 3) & 15 > 10 and not(8>>2 == 2) or (7 ^ 3 !=4))
```

```
False
```