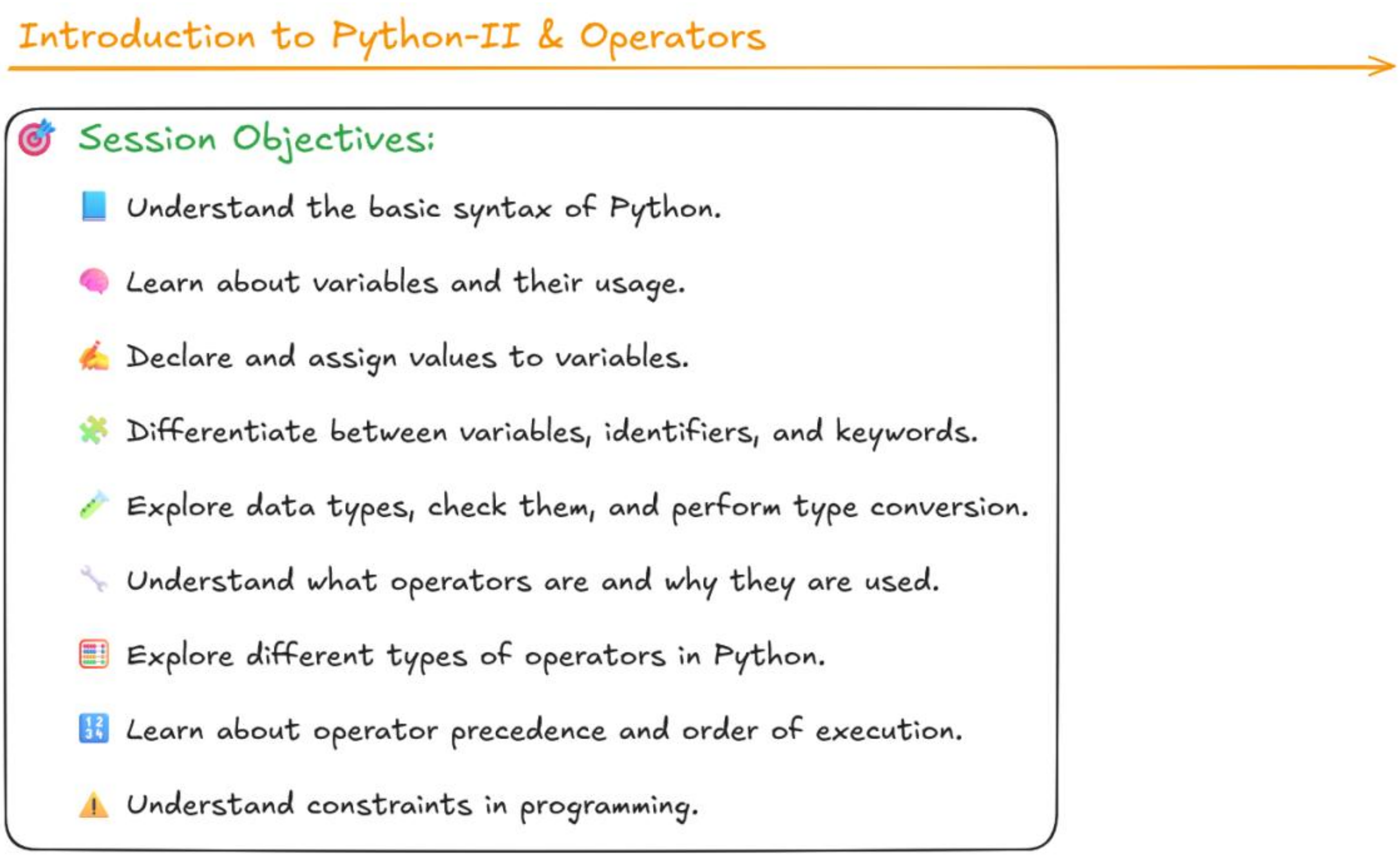
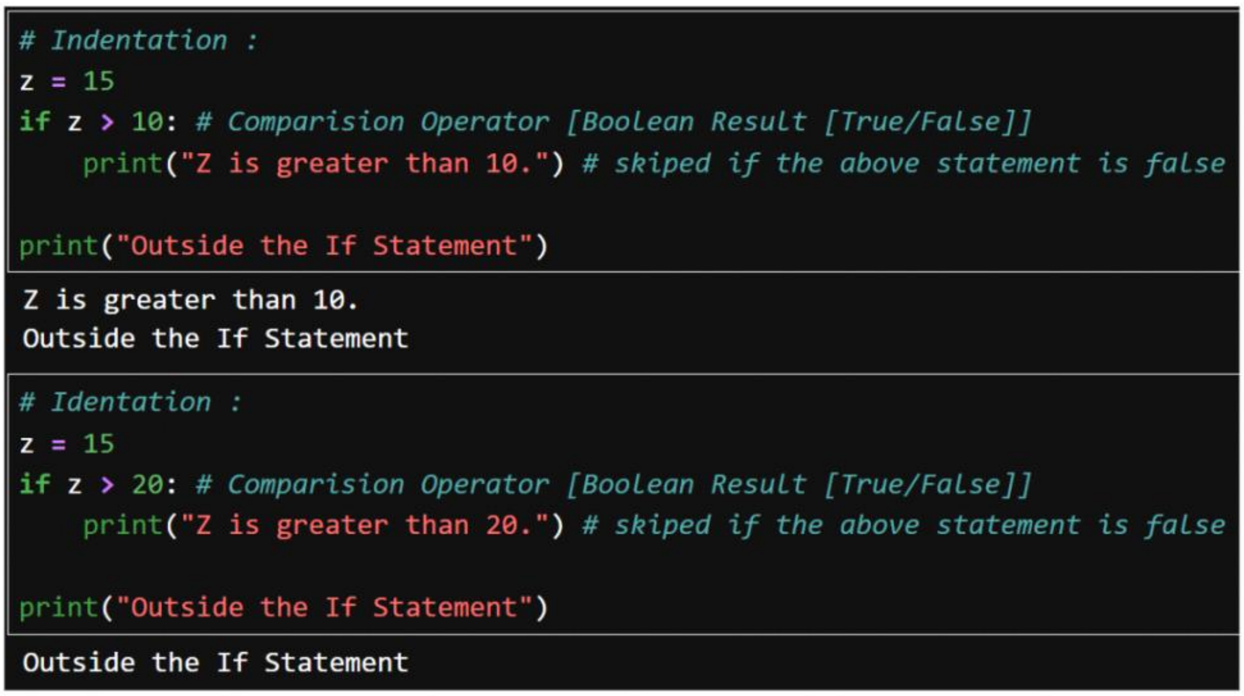


Introduction to Python-II & Operators

🎯 Session Objectives:

- 📘 Understand the basic syntax of Python.
- 🧠 Learn about variables and their usage.
- ✍️ Declare and assign values to variables.
- 🧩 Differentiate between variables, identifiers, and keywords.
- 🧪 Explore data types, check them, and perform type conversion.
- 🔧 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.

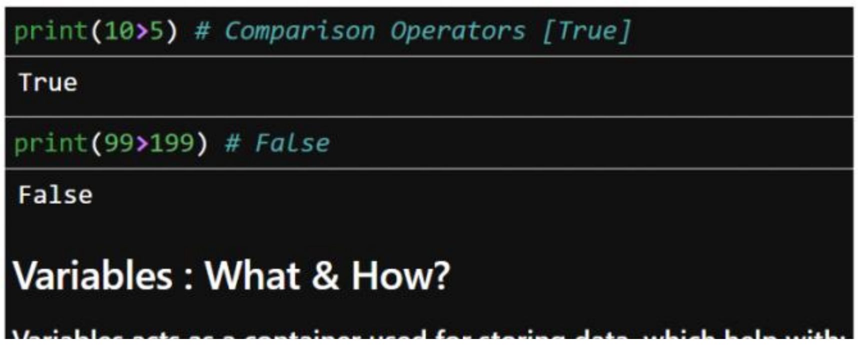


```python
# Indentation :
z = 15
if z > 10: # Comparision Operator [Boolean Result [True/False]]
    print("Z is greater than 10.") # skiped if the above statement is false

print("Outside the If Statement")
```

```
Z is greater than 10.
Outside the If Statement
```

```python
# Identation :
z = 15
if z > 20: # Comparision Operator [Boolean Result [True/False]]
    print("Z is greater than 20.") # skiped if the above statement is false

print("Outside the If Statement")
```

```
Outside the If Statement
```



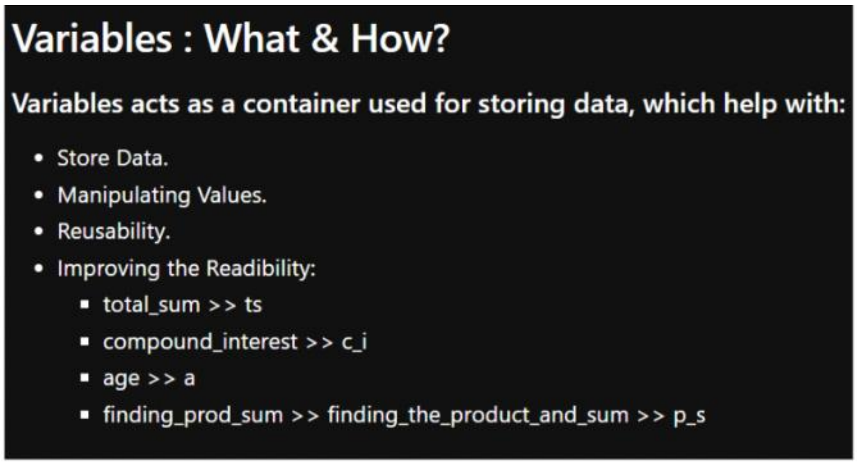
```python
print(10>5) # Comparison Operators [True]
```

```
True
```

```python
print(99>199) # False
```

```
False
```

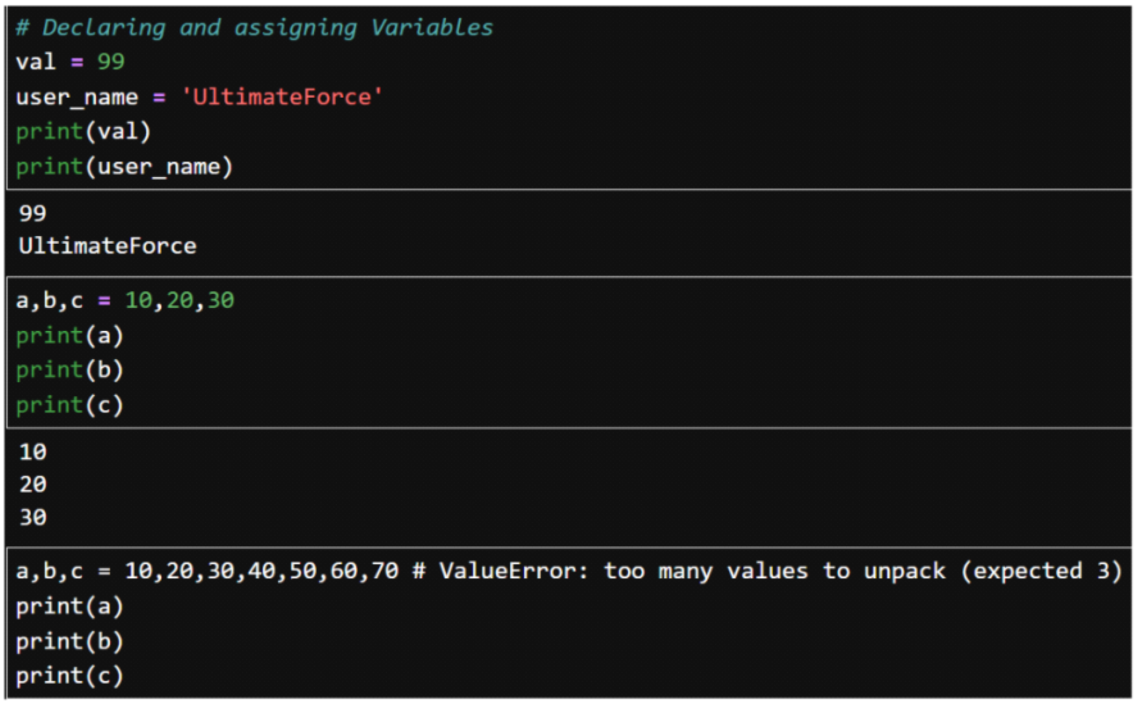
Variables : What & How?



Variables : What & How?

Variables acts as a container used for storing data, which help with:

- Store Data.
- Manipulating Values.
- Reusability.
- Improving the Readibility:
  - total_sum >> ts
  - compound_interest >> c_i
  - age >> a
  - finding_prod_sum >> finding_the_product_and_sum >> p_s



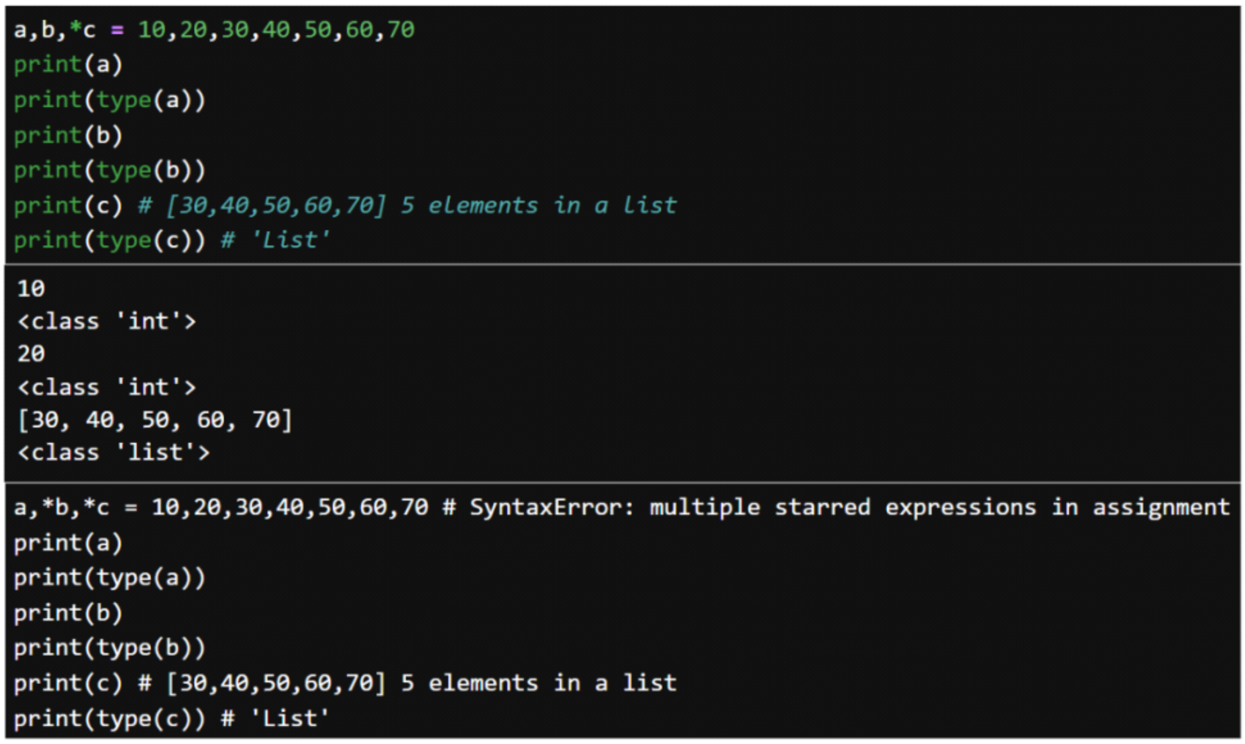
```
# Declaring and assigning Variables
val = 99
user_name = 'UltimateForce'
print(val)
print(user_name)
```

```
99
UltimateForce
```

```
a,b,c = 10,20,30
print(a)
print(b)
print(c)
```

```
10
20
30
```

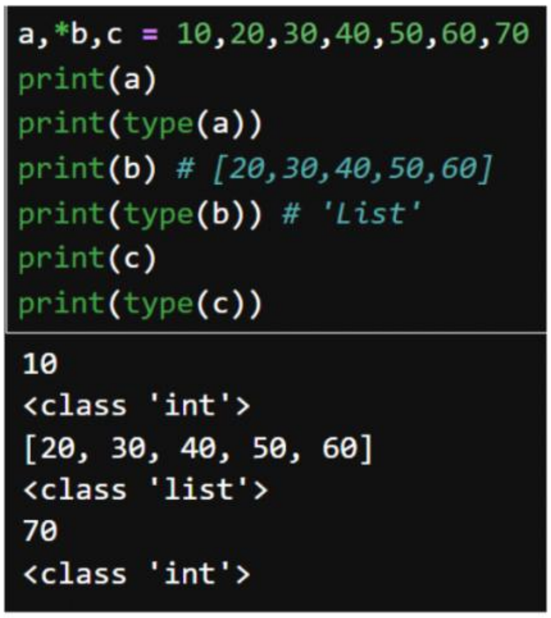
```
a,b,c = 10,20,30,40,50,60,70 # ValueError: too many values to unpack (expected 3)
print(a)
print(b)
print(c)
```



```
a,b,*c = 10,20,30,40,50,60,70
print(a)
print(type(a))
print(b)
print(type(b))
print(c) # [30,40,50,60,70] 5 elements in a list
print(type(c)) # 'List'
```

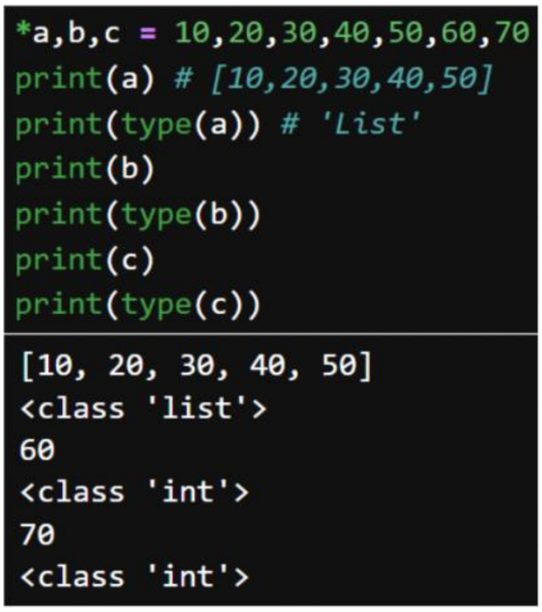
```
10
<class 'int'>
20
<class 'int'>
[30, 40, 50, 60, 70]
<class 'list'>
```

```
a,*b,*c = 10,20,30,40,50,60,70 # SyntaxError: multiple starred expressions in assignment
print(a)
print(type(a))
print(b)
print(type(b))
print(c) # [30,40,50,60,70] 5 elements in a list
print(type(c)) # 'List'
```



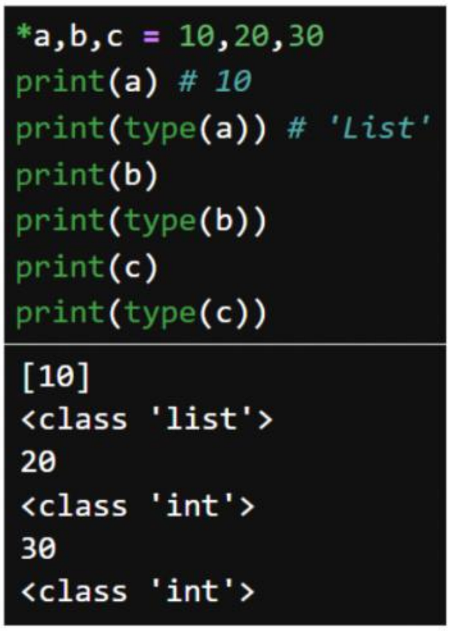
```python
a,*b,c = 10,20,30,40,50,60,70
print(a)
print(type(a))
print(b) # [20,30,40,50,60]
print(type(b)) # 'List'
print(c)
print(type(c))
```

```
10
<class 'int'>
[20, 30, 40, 50, 60]
<class 'list'>
70
<class 'int'>
```



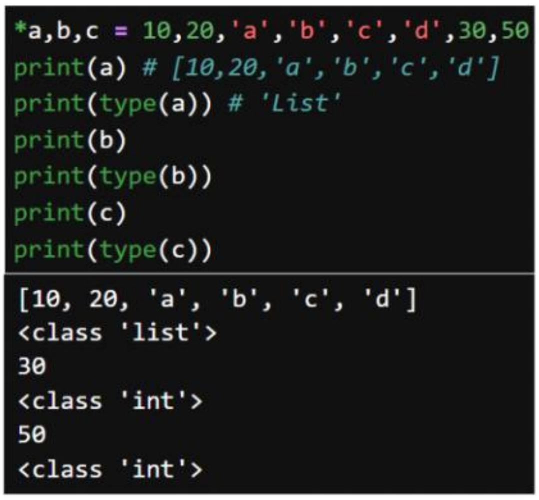
```python
*a,b,c = 10,20,30,40,50,60,70
print(a) # [10,20,30,40,50]
print(type(a)) # 'List'
print(b)
print(type(b))
print(c)
print(type(c))
```

```
[10, 20, 30, 40, 50]
<class 'list'>
60
<class 'int'>
70
<class 'int'>
```



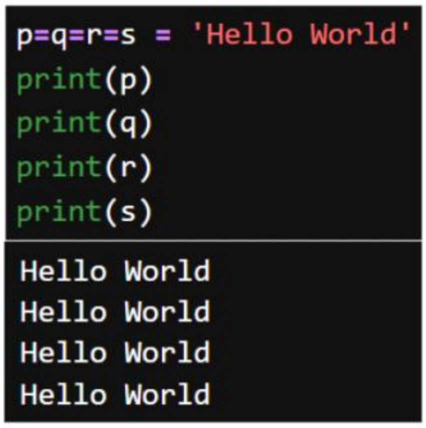
```python
*a,b,c = 10,20,30
print(a) # 10
print(type(a)) # 'List'
print(b)
print(type(b))
print(c)
print(type(c))
```

```
[10]
<class 'list'>
20
<class 'int'>
30
<class 'int'>
```



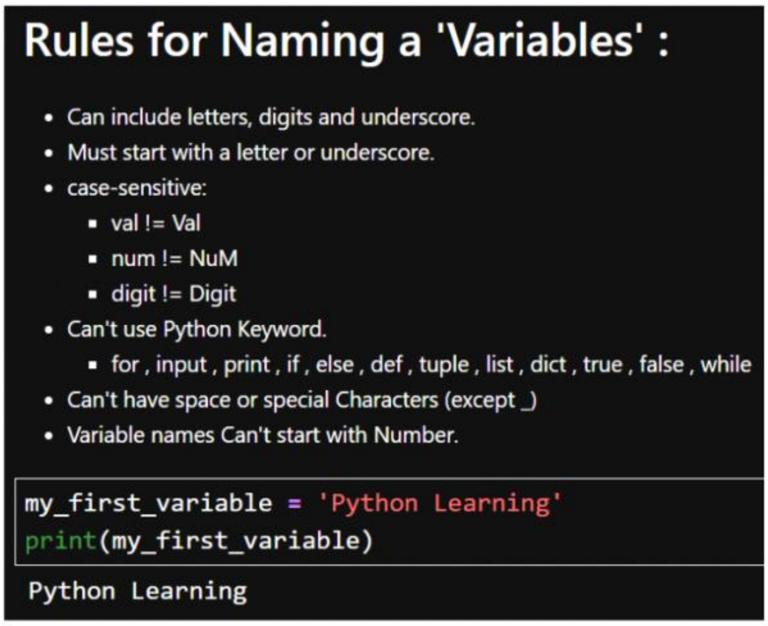
```python
*a,b,c = 10,20,'a','b','c','d',30,50
print(a) # [10,20,'a','b','c','d']
print(type(a)) # 'List'
print(b)
print(type(b))
print(c)
print(type(c))
```

```
[10, 20, 'a', 'b', 'c', 'd']
<class 'list'>
30
<class 'int'>
50
<class 'int'>
```



```python
p=q=r=s = 'Hello World'
print(p)
print(q)
print(r)
print(s)
```

```
Hello World
Hello World
Hello World
Hello World
```

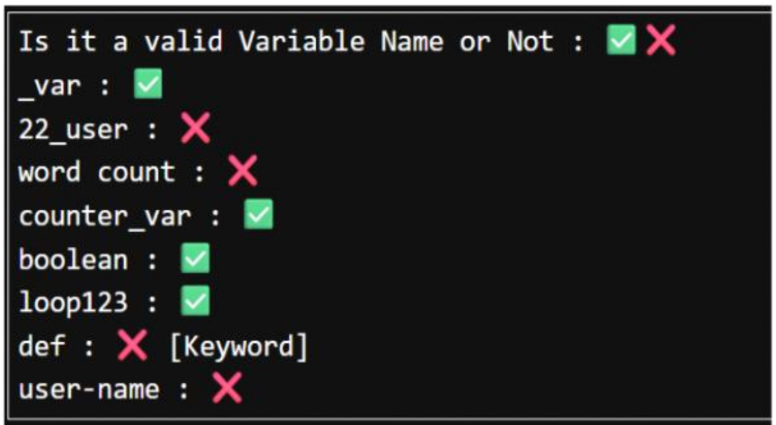


Rules for Naming a 'Variables' :

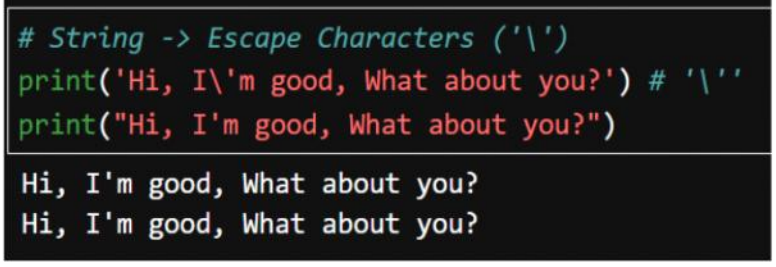
- Can include letters, digits and underscore.
- Must start with a letter or underscore.
- case-sensitive:
  - val != Val
  - num != NuM
  - digit != Digit
- Can't use Python Keyword.
  - for , input , print , if , else , def , tuple , list , dict , true , false , while
- Can't have space or special Characters (except _)
- Variable names Can't start with Number.

```python
my_first_variable = 'Python Learning'
print(my_first_variable)
```

```
Python Learning
```

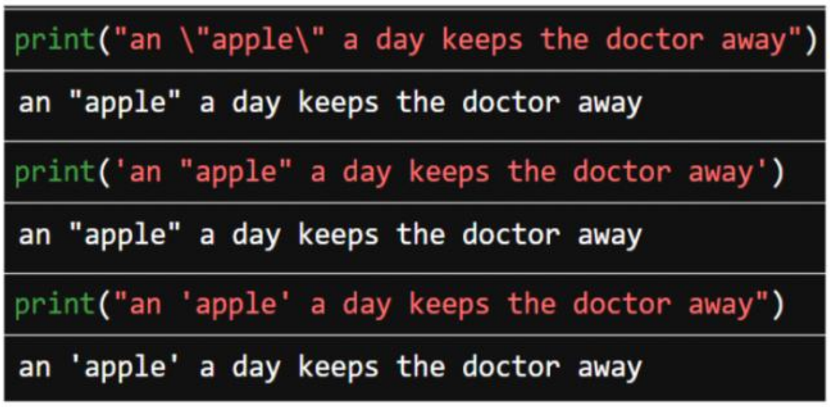


```
Is it a valid Variable Name or Not : ✅❌
_var : ✅
22_user : ❌
word count : ❌
counter_var : ✅
boolean : ✅
loop123 : ✅
def : ❌ [Keyword]
user-name : ❌
```



```
# String -> Escape Characters ('\')
print('Hi, I\'m good, What about you?') # '\''
print("Hi, I'm good, What about you?")
```

```
Hi, I'm good, What about you?
Hi, I'm good, What about you?
```



```
print("an \"apple\" a day keeps the doctor away")
```

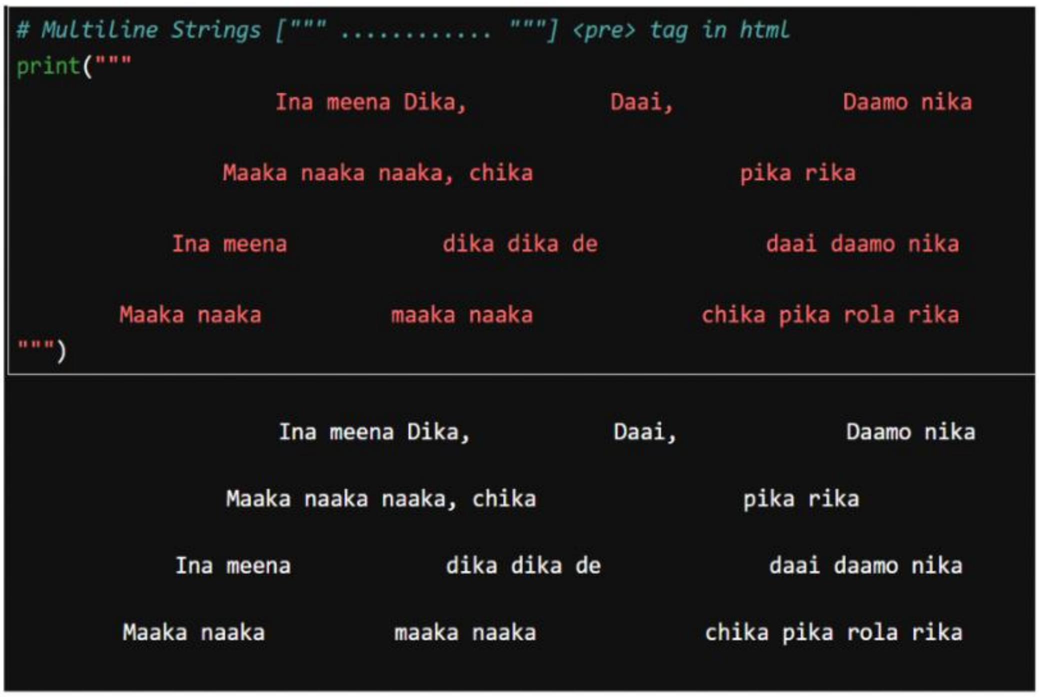
```
an "apple" a day keeps the doctor away
```

```
print('an "apple" a day keeps the doctor away')
```

```
an "apple" a day keeps the doctor away
```

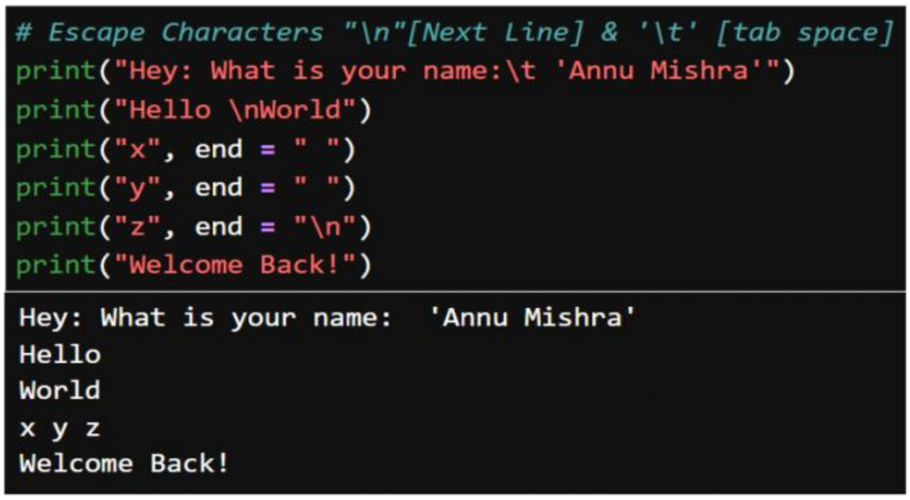
```
print("an 'apple' a day keeps the doctor away")
```

```
an 'apple' a day keeps the doctor away
```



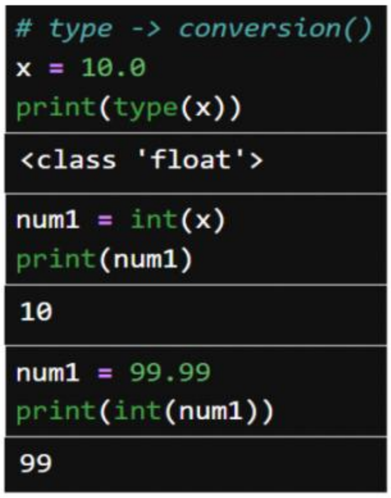
```
# Multiline Strings [""" ............ """] <pre> tag in html
print("""
                       Ina meena Dika,              Daai,               Daamo nika

                  Maaka naaka naaka, chika                     pika rika

              Ina meena              dika dika de                daai daamo nika

         Maaka naaka             maaka naaka                chika pika rola rika
""")
```

```

                       Ina meena Dika,              Daai,               Daamo nika

                  Maaka naaka naaka, chika                     pika rika

              Ina meena              dika dika de                daai daamo nika

         Maaka naaka             maaka naaka                chika pika rola rika
```



```
# Escape Characters "\n"[Next Line] & '\t' [tab space]
print("Hey: What is your name:\t 'Annu Mishra'")
print("Hello \nWorld")
print("x", end = " ")
print("y", end = " ")
print("z", end = "\n")
print("Welcome Back!")
```

```
Hey: What is your name:  'Annu Mishra'
Hello
World
x y z
Welcome Back!
```



```
# type -> conversion()
x = 10.0
print(type(x))
```

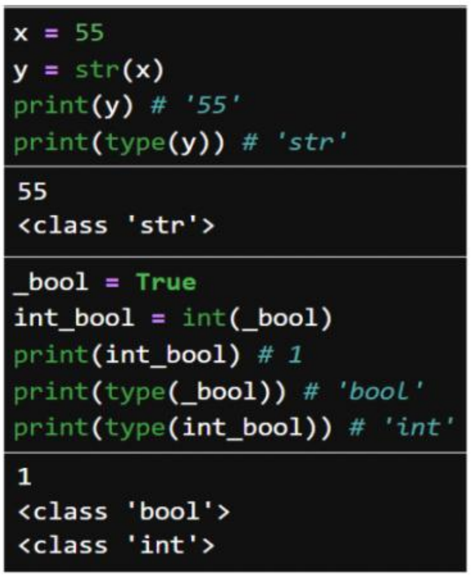
```
<class 'float'>
```

```
num1 = int(x)
print(num1)
```

```
10
```

```
num1 = 99.99
print(int(num1))
```

```
99
```

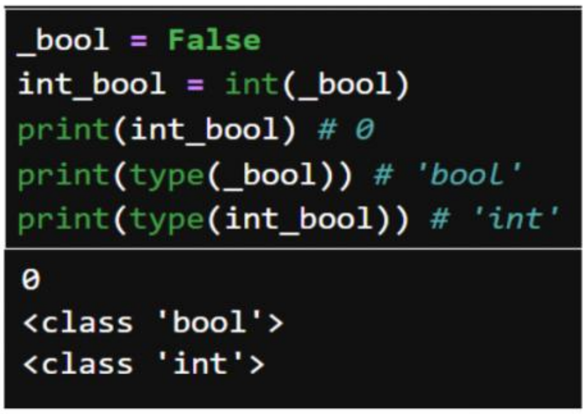


```
x = 55
y = str(x)
print(y) # '55'
print(type(y)) # 'str'
```

```
55
<class 'str'>
```

```
_bool = True
int_bool = int(_bool)
print(int_bool) # 1
print(type(_bool)) # 'bool'
print(type(int_bool)) # 'int'
```

```
1
<class 'bool'>
<class 'int'>
```



```
_bool = False
int_bool = int(_bool)
print(int_bool) # 0
print(type(_bool)) # 'bool'
print(type(int_bool)) # 'int'
```

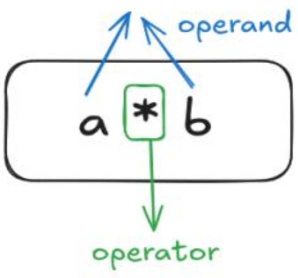
```
0
<class 'bool'>
<class 'int'>
```

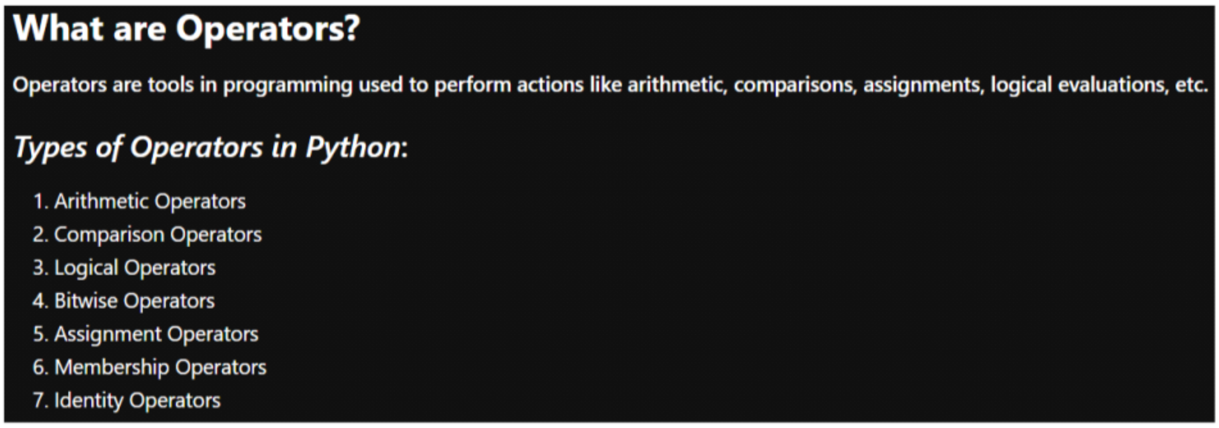
What are Operators?

Operators are tools in programming used to perform actions like arithmetic, comparisons, assignments, logical evaluations, etc.

Types of Operators in Python:

1. Arithmetic Operators
2. Comparison Operators
3. Logical Operators
4. Bitwise Operators
5. Assignment Operators
6. Membership Operators
7. Identity Operators



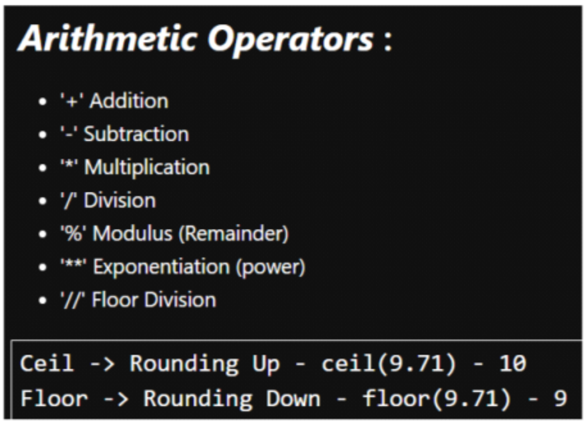


What are Operators?

Operators are tools in programming used to perform actions like arithmetic, comparisons, assignments, logical evaluations, etc.

Types of Operators in Python:

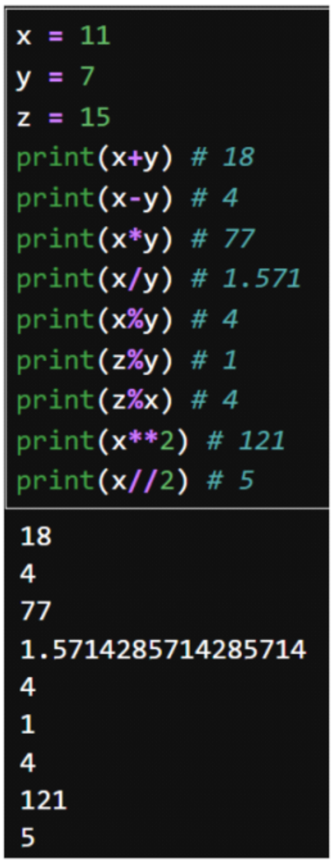
1. Arithmetic Operators
2. Comparison Operators
3. Logical Operators
4. Bitwise Operators
5. Assignment Operators
6. Membership Operators
7. Identity Operators



Arithmetic Operators :

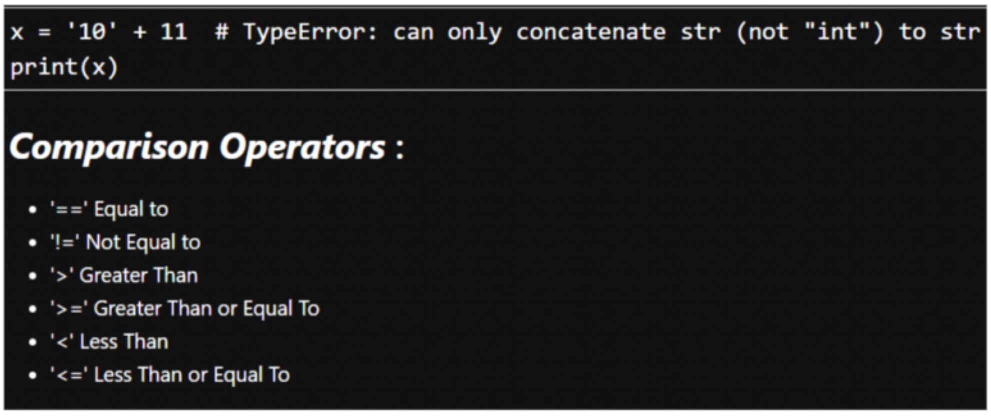
- '+' Addition
- '-' Subtraction
- '*' Multiplication
- '/' Division
- '%' Modulus (Remainder)
- '**' Exponentiation (power)
- '//' Floor Division

```
Ceil -> Rounding Up - ceil(9.71) - 10
Floor -> Rounding Down - floor(9.71) - 9
```



```python
x = 11
y = 7
z = 15
print(x+y) # 18
print(x-y) # 4
print(x*y) # 77
print(x/y) # 1.571
print(x%y) # 4
print(z%y) # 1
print(z%x) # 4
print(x**2) # 121
print(x//2) # 5
```

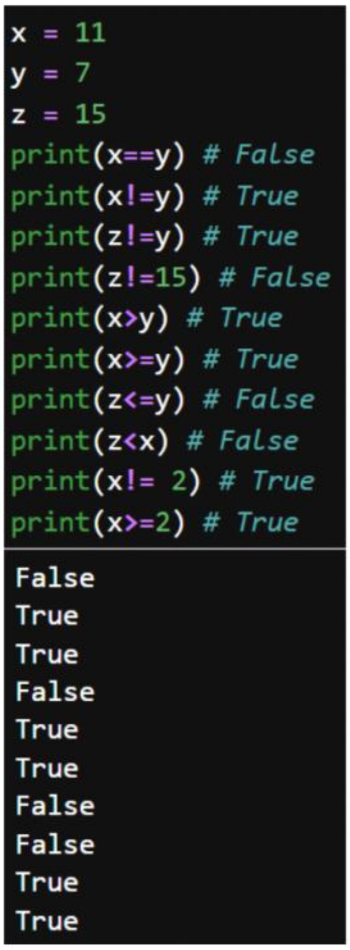
```
18
4
77
1.5714285714285714
4
1
4
121
5
```



```python
x = '10' + 11  # TypeError: can only concatenate str (not "int") to str
print(x)
```

Comparison Operators :

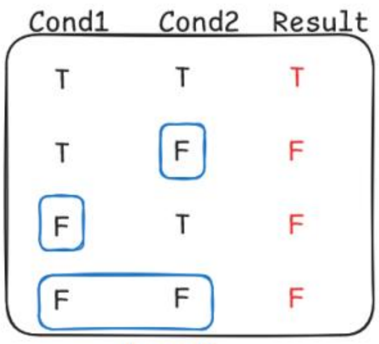
- '==' Equal to
- '!=' Not Equal to
- '>' Greater Than
- '>=' Greater Than or Equal To
- '<' Less Than
- '<=' Less Than or Equal To



```python
x = 11
y = 7
z = 15
print(x==y) # False
print(x!=y) # True
print(z!=y) # True
print(z!=15) # False
print(x>y) # True
print(x>=y) # True
print(z<=y) # False
print(z<x) # False
print(x!= 2) # True
print(x>=2) # True
```

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

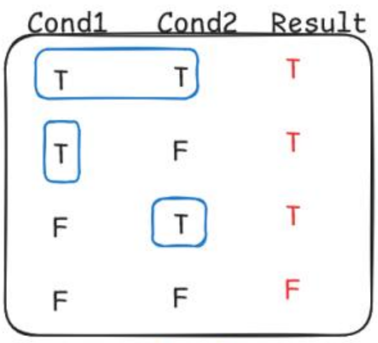
Logical Operators:



Cond1	Cond2	Result
T	T	T
T	$\boxed{F}$	F
$\boxed{F}$	T	F
F	F	F

'and'

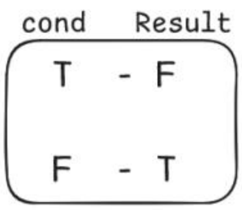
Both Condition needs to be True in order to make the Result True



Cond1	Cond2	Result
T	T	T
$\boxed{T}$	F	T
F	$\boxed{T}$	T
F	F	F

'or'

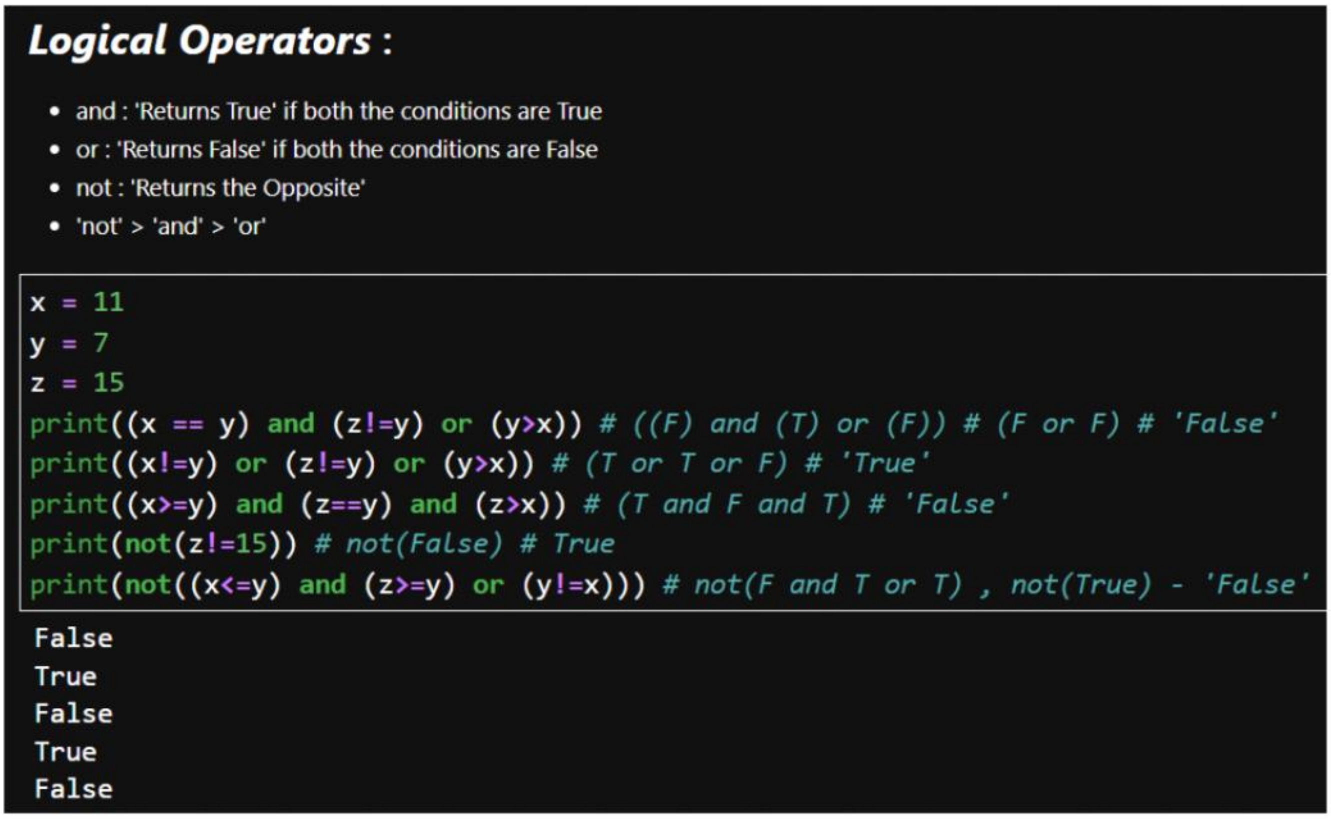
If any condition is True, your final result would be True



cond	Result
T -	F
F -	T

'not'

negate the condition using ~ not logic



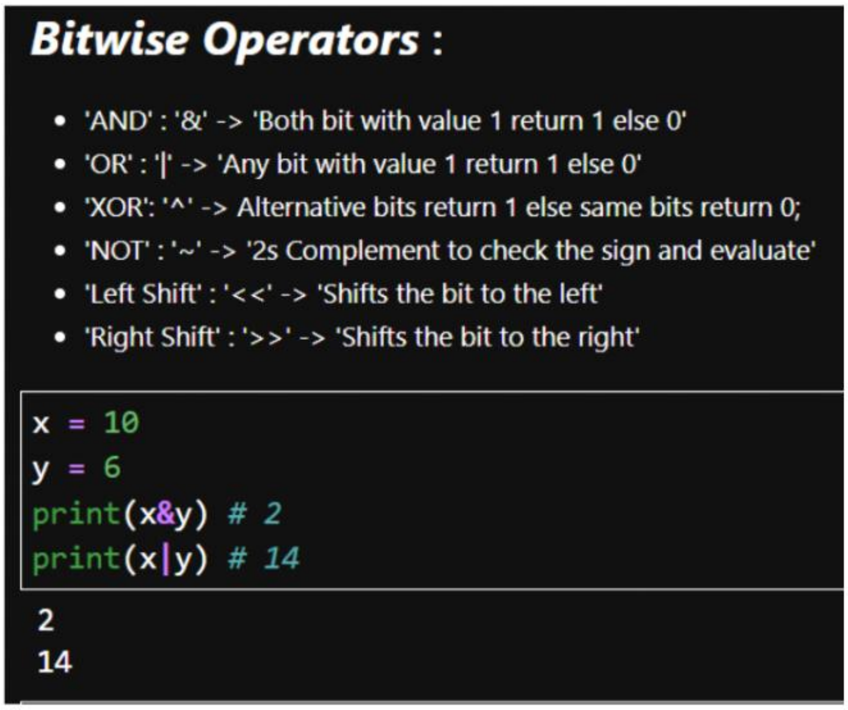
Logical Operators :

- and : 'Returns True' if both the conditions are True
- or : 'Returns False' if both the conditions are False
- not : 'Returns the Opposite'
- 'not' > 'and' > 'or'

```
x = 11
y = 7
z = 15
print((x == y) and (z!=y) or (y>x)) # ((F) and (T) or (F)) # (F or F) # 'False'
print((x!=y) or (z!=y) or (y>x)) # (T or T or F) # 'True'
print((x>=y) and (z==y) and (z>x)) # (T and F and T) # 'False'
print(not(z!=15)) # not(False) # True
print(not((x<=y) and (z>=y) or (y!=x))) # not(F and T or T) , not(True) - 'False'
```

```
False
True
False
True
False
```

Bitwise Operators



Bitwise Operators :

- 'AND' : '&' -> 'Both bit with value 1 return 1 else 0'
- 'OR' : '|' -> 'Any bit with value 1 return 1 else 0'
- 'XOR': '^' -> Alternative bits return 1 else same bits return 0;
- 'NOT' : '~' -> '2s Complement to check the sign and evaluate'
- 'Left Shift' : '<<' -> 'Shifts the bit to the left'
- 'Right Shift' : '>>' -> 'Shifts the bit to the right'

```
x = 10
y = 6
print(x&y) # 2
print(x|y) # 14
```

```
2
14
```

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

'and &'

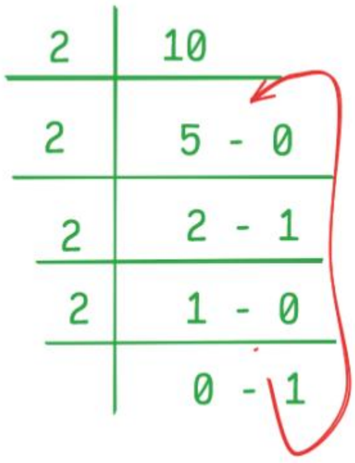
bit1	bit2	Result
1	1	1
1	0	1
0	1	1
0	0	0

'or |'

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

'XOR ^'

10 decimal -> binary



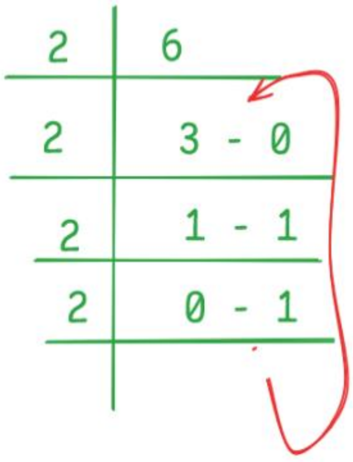
10 -> 1010

1010

2^3 2^2 2^1 2^0

(1*2^3 + 0*2^2 + 1*2^1 + 0*2^0)
= (1*8 + 0*4 + 1*2 + 0*1)
= 8 + 2 = 10

10 & 6

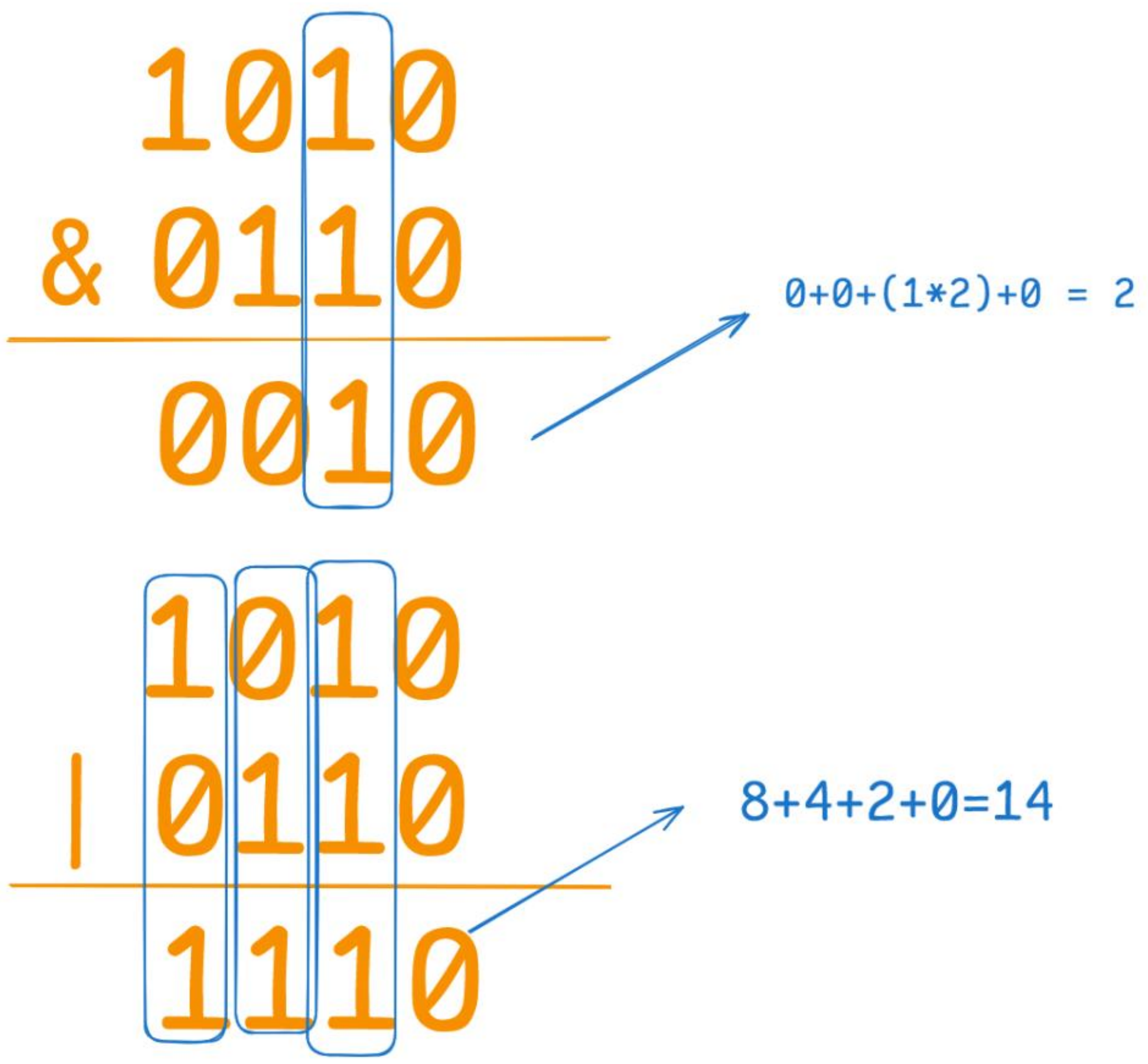


6 -> 0110

0110

2^3 2^2 2^1 2^0

= (0*8 + 1*4 + 1*2 + 0*1)
= (0 + 4 + 2 + 0) = 6


1010
& 0110
0010
0+0+(1*2)+0 = 2
1010
| 0110
1110
8+4+2+0=14