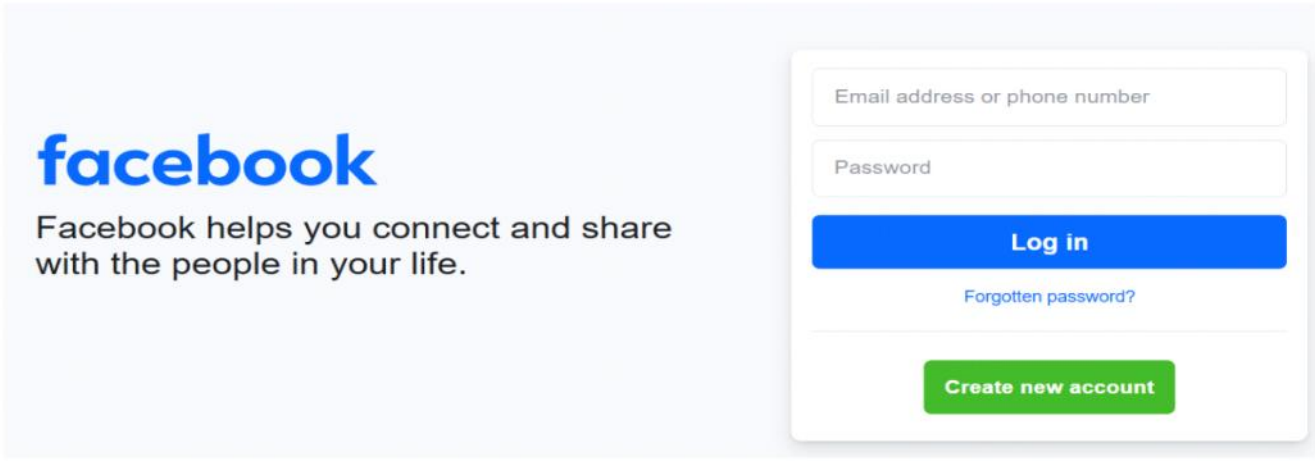
# Introduction to Python-2 & Operators

## 🎓 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

## Input



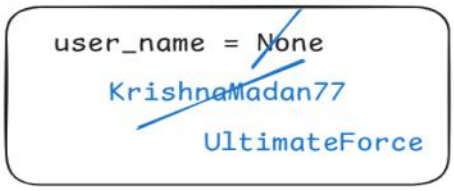

```
user_name = input("Enter the UserName: ")
print(user_name)
```

```
Enter the UserName:  KrishnaMadan77
KrishnaMadan77
```

```
user_name = input("Enter the UserName: ")
print(user_name)
```

```
Enter the UserName:  UltimateForce
UltimateForce
```

Memory Diagram




Console

## *Input()* :

- It is used to accept user input from the keyboard as a string, which can be further type-casted accordingly.

```
user_name = input("Enter the UserName: ")
print(user_name)
```

```
Enter the UserName:  KrishnaMadan77
KrishnaMadan77
```

```
user_name = input("Enter the UserName: ")
print(user_name)
```

```
Enter the UserName:  UltimateForce
UltimateForce
```

```
val = input("Enter the number between 1 to 100: ")
print(val)
print(type(val))
```

```
Enter the number between 1 to 100:  77
77
<class 'str'>
```

```
x = '99'
y = int(x) # int('99') - 99
print(y)
print(type(y))
print(type(x))
```

```
99
<class 'int'>
<class 'str'>
```

```
val = int(input("Enter the number between 1 to 100: "))
print(val)
print(type(val))
```

```
Enter the number between 1 to 100:  91
91
<class 'int'>
```

```
x = input("Enter the number between 1 to 100: ")
x = x + 10 # 'str' + 10 : error
print(x) # TypeError: can only concatenate str (not "int") to str
```

```
# String Patterns 'F-Strings' formatted prints
employee_name = input("Enter the Employee Name : ")
designation = input("Enter the Designation of the Employee: ")
print(f"Welcome {employee_name}, You are hired as a {designation} role.")
```

```
Enter the Employee Name :  Rajat Singh Thakur
Enter the Designation of the Employee:  Senior Analyst
Welcome Rajat Singh Thakur, You are hired as a Senior Analyst role.
```

```python
# Single Line Comment
'''
    This is a multiline Comments
'''
print("Hi, EveryOne. Welcome to the python Module!") # Single line print Statement
```

```
Hi, EveryOne. Welcome to the python Module!
```

```python
# Multiline String input -> <pre> tag in HTML
print("""
    This is a     Python Series 🌀.
            Welcome to the Awesome Session 🔥.....
                    I hope you learn Something Valueable 💞
""")
```

```
    This is a     Python Series 🌀.
            Welcome to the Awesome Session 🔥.....
                    I hope you learn Something Valueable 💞
```

```python
# Variables for dynamic input
candidate_name = input("Enter candidate name: ")
position = input("Enter job position: ")
company = input("Enter the Company Name: ")
joining_date = input("Enter joining date (e.g., 1 Dec 2025): ")
salary = int(input("Enter salary package : "))
location = input("Enter job location: ")

# Multiline print statement using f-strings
print(f"""
    Dear {candidate_name},

    We are pleased to offer you the position of {position} at our {company}
    Your expected joining date will be {joining_date}.

    You will be provided a salary package of {salary}.
    The position is based at our {location} office.

    Please confirm your acceptance of this offer by replying to this email.

    Best regards,
    HR Team
""")
```

```
Enter candidate name:  Shyam Sundar
Enter job position:  Data Analyst
Enter the Company Name:  PhonePe
Enter joining date (e.g., 1 Dec 2025):  11th Jan 2026
Enter salary package :  1100000
Enter job location:  Gurugram

    Dear Shyam Sundar,

    We are pleased to offer you the position of Data Analyst at our PhonePe.
    Your expected joining date will be 11th Jan 2026.

    You will be provided a salary package of 1100000.
    The position is based at our Gurugram office.

    Please confirm your acceptance of this offer by replying to this email.

    Best regards,
    HR Team
```

```python
# Arithematic Operators : [Alt + Shift + down key -> Duplicate row]
num1 = int(input("Enter your First Value : "))
num2 = int(input("Enter your Second Value : "))
add = (num1 + num2)
mul = (num1 * num2)
sub = (num1 - num2)
div = (num1 / num2)
print(add)
print(mul)
print(sub)
print(div)
```

```
Enter your First Value :  11
Enter your Second Value :  7
18
77
4
1.5714285714285714
```

```python
fav_car = input("Enter Your Favourite Car: ")
print("My Dream Car is : ", fav_car)
print(f"My Dream Car is : {fav_car}")
```

```
Enter Your Favourite Car:  Mercedes G63 AMG
My Dream Car is :  Mercedes G63 AMG
My Dream Car is : Mercedes G63 AMG
```

```python
# Area of Circle
pi = 3.14
r = float(input("Enter the radius value: ")) # radius
circle_area = pi * r * r
print(r)
print(type(r))
print(circle_area)
print(type(circle_area))
```

```
Enter the radius value:  2.75
2.75
<class 'float'>
23.74625
<class 'float'>
```

```
float >> int [Upcasting]
```

```
# Indentation:
Java {}
if(cond){
    print("Something")
}
Python ":"
if cond:
    print("Something")
```

```
# Indentation :
if condition : # Return Boolean [True/False]
    print("") # if the condition is false this line would be skipped ❌
```

```
# Indentation
z = 15
if z > 10: # True
    print("Z is Greater than 10.")

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

```
Z is Greater than 10.
Outside the If Condition
```

```
# Indentation
z = 7
if z > 10: # False
    print("Z is Greater than 10.") # Skipped

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

```
Outside the If Condition
```

```
# Indentation
z = 7
if z > 10: # False [IndentationError: expected an indented block after 'if' statement on line 3]
print("Z is Greater than 10.") # Error

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

## Variables : *What & Why?*

**Variables acts as a container for storing data. They help with:**

- Storing Data
- Manipulating Values
- Reusability
- Improving the Readability:
  - total_sum >> t_s
  - compound_interest > c_i
  - customer_age >> ca
  - 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
```

```python
# Unpacking a tuple
a,b,c  = 10,20,30 # 3 LHS = 3 RHS.
print(a) # 10
print(b) # 20
print(c) # 30
```

```
10
20
30
```

```python
# *arbitary value [Only one person has to take all the other elements [Container <list>]]
a,b,*c = 10,20,30,40,50,60,70,80,90 # 3LHS = 9 RHS
print(a) # 10
print(type(a)) # 'int'
print(b) # 20
print(type(b)) # 'int'
print(c) # [30,40,50,60,70,80,90]
print(type(c)) # 'list'
```

```
10
<class 'int'>
20
<class 'int'>
[30, 40, 50, 60, 70, 80, 90]
<class 'list'>
```

```python
a,*b,*c = 10,20,30,40,50,60,70,80,90 # 3LHS = 9 RHS
print(a) # 10
print(b) # 'Confused'
print(c) # 'Confused'
# '*b , *c'  -> Error [SyntaxError: multiple starred expressions in assignment]
```

```python
a,*b,c = 10,20,30,40,50,60,70,80,90 # 3LHS = 9 RHS
print(a) # 10
print(type(a)) # 'int'
print(b) # [20,30,40,50,60,70,80]
print(type(b)) # 'list'
print(c) # 90
print(type(c)) # 'int'
```

```
10
<class 'int'>
[20, 30, 40, 50, 60, 70, 80]
<class 'list'>
90
<class 'int'>
```

```python
*a,b,c = 10,20,30,40,50,60,70,80,90 # 3LHS = 9 RHS
print(a) # [10,20,30,40,50,60,70]
print(type(a)) # 'list'
print(b) # 80
print(type(b)) # 'int'
print(c) # 90
print(type(c)) # 'int'
```

```
[10, 20, 30, 40, 50, 60, 70]
<class 'list'>
80
<class 'int'>
90
<class 'int'>
```

```python
p=q=r=s = "Programming"
print(p)
print(q)
print(r)
print(s)
```

```
Programming
Programming
Programming
Programming
```

## *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 Keywords
  - for, while, if, else, in, list, elif, int, float,
  - tuple, return, class, dict, None, True, False...
- Can't have space or any special characters (except _)
- Variables name can't start with numbers

```
123_number ❌
user name ❌
findingSum ✅
boolean ✅
bool ❌ [Keyword]
profileName123$ ❌
profile123 ✅
customer_name_1 ✅
product-sum ❌
```

```
# String -> Escape Characters ('\')
print('Hi I'm good, What about you?') # SyntaxError: unterminated string literal (detected at line 2)
```

```
print('Hi I\'m good, What about you?')
```

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

```
print("Hi I'm good, What about you?")
```

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

```
print("an "apple" a day, keeps the doctor away")
SyntaxError: invalid syntax. Perhaps you forgot a comma?
```

```
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
```

```
# Quick Challenge:
# Escape Characters ('\') , (\n)['Next Line'] (\t)['tab space']
print('Hey, What\'s your name: \t \'Aditya Sharma\'')
print("Hello\nWorld")
print('x', end= " ")
print('y', end= " ")
print('z', end= "\n")
print("Welcome Back!")
```

```
Hey, What's your name:    'Aditya Sharma'
Hello
World
x y z
Welcome Back!
```

```
print('x', end= " ")
print('y', end= " ")
print('z', end= "\n")
x y z
Welcome Back!
```

```
print('x', 'y', 'z', end = "\t")
```

```
x y z
```

```python
print('x', 'y', 'z', end = '\n')
print('a', 'b', 'c')
```

```
x y z
a b c
```

```python
# type -> Conversion
x = 10.0
print(type(x))
```

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

```python
num1 = 99.99 # float
print(int(num1)) # 'int' -> round down -> 99
result = int(num1)
print(type(num1)) # 'float'
print(type(result)) # 'int'
```

```
99
<class 'float'>
<class 'int'>
```

```python
x = 77
y = str(x) # '77'
print(y)
print(type(y)) # 'str'
```

```
77
<class 'str'>
```

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

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

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

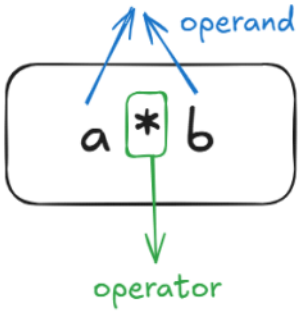
```
0
<class 'int'>
<class 'bool'>
```

## 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': '/' (floating)
- 'Modulus': '%' (remainder)
- 'Exponentiation': '**' (power)
- 'floor division': '//'

```
ceil : Rounding Up -> 7.79 -> ceil(7.79) - 8
floor : Rounding Down -> 7.99 -> floor(7.99) - 7
```

```
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(z//2) # 7
```

```
18
4
77
1.5714285714285714
4
1
4
121
7
```

### Comparision Operators:

Compare 2 values and Returns Boolean (True/False)

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

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

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

## Logical Operators: ¶

- 'and' : Return True If both the conditions are 'True' Else 'False'
- 'or' : Return False If both the conditions are 'False' Else 'True'
- 'not' : Return the Opposite
- 'not' > 'and' > 'or'

```python
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) # (T or F) # True
print((x>=y) and (z==y) and (z>x)) # (T and F and T) # (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(F or T) # not(T) # False
print(not(x<=y) and (z>=y) or (y!=x)) # not(F) and T or T # (T and T or T) # True
print(not(x<=y) or (z>=y) and (y==x)) # (not(F) or T and F) # (T or T and F) # (T or F) # True
```

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