**Aim:** Write a program to demonstrate different number datatypes in python.

**IDE:**

Data types in Python refer to classifying or categorizing data objects based on their characteristics and behavior. They determine the type of values variables can hold and specify the operations that can be performed on those values. For instance, Python has several built-in data types, including numeric types (int, float, complex), string (str), Boolean (bool), and collection types (list, tuple, dict, set). Moreover, each data type has its own set of properties, methods, and behaviors that allow programmers to manipulate and process data effectively in their programs.

**Built-in Data Types in Python**

Built-in data types in Python are fundamental data structures provided by the Python programming language. Pre-defined and available for use without requiring any additional libraries or modules. Python offers several built-in data types, including:

**Numeric Data Types:** Numeric data types in Python are used to represent numerical values. Python provides three primary numeric datatypes in python:

* Integer (int): Integers are whole numbers without any decimal points. They can be positive or negative.
* Floating-Point (float): Floating-point numbers represent decimal values. They can be positive or negative and may contain a decimal point.
* Complex (complex): People use complex numbers to represent numbers with a real and imaginary part. You write them in the form of a + bj, where a is the real part and b is the imaginary part.

**String Data Type(str):** Represents a sequence of characters enclosed in single quotes (‘ ‘) or double quotes (” “), such as “Hello, World!”, ‘Python’.

**Boolean Data Type(bool):** Represents either True or False, used for logical operations and conditions.

**Collection Data Types:**

* list: Represents an ordered and mutable collection of items, enclosed in square brackets [].
* tuple: Represents an ordered and immutable collection of items, enclosed in parentheses ().
* dict: Represents a collection of key-value pairs enclosed in curly braces {} with unique keys.
* set: Represents an unordered and mutable collection of unique elements, enclosed in curly braces {} or using the set() function.

**Results:**

Attach the screenshot of each task along with the output

**Numeric Data Types**

**Python Code:**

num1 = 10

print(num1)

print(“Datatype of num1 is”, type(num1))

num2 = 2.5

print(num2)

print(“Datatype of num1 is”, type(num2))

num3 = 2+6j

print(num3)

print(“Datatype of num1 is”, type(num3))

**Example 1**

x = 5

y = -6

# Performing arithmetic operations

sum\_result = x + y

difference\_result = x - y

multiplication\_result = x \* y

division\_result = x / y

# Printing the results

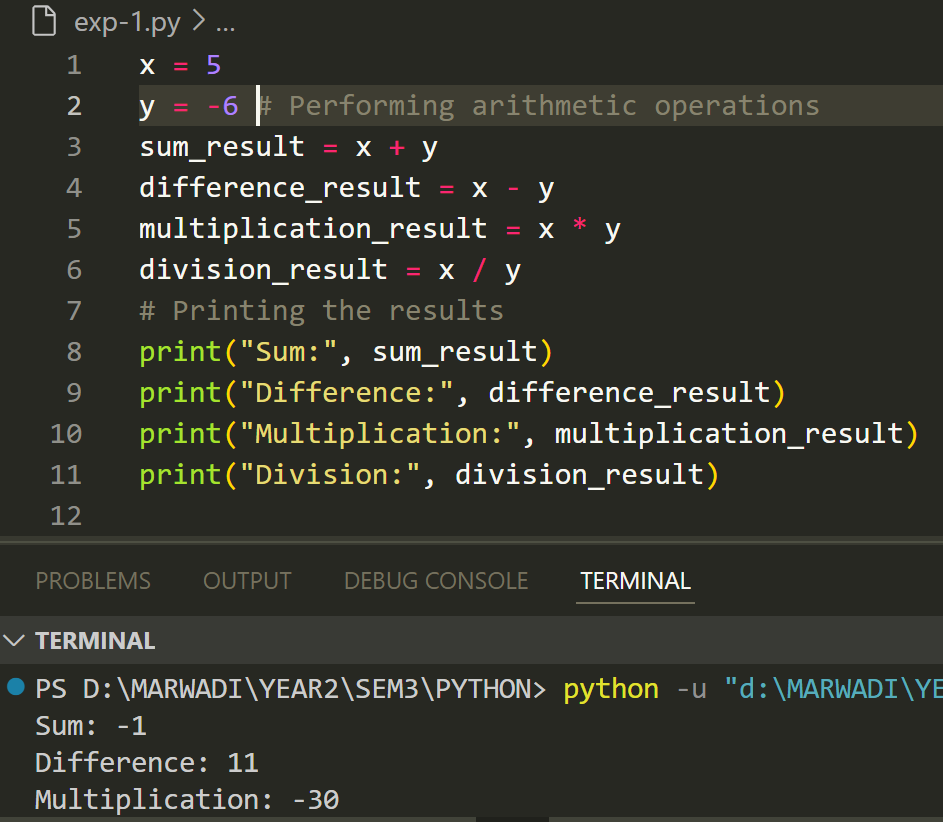
print("Sum:", sum\_result)

print("Difference:", difference\_result)

print("Multiplication:", multiplication\_result)

print("Division:", division\_result)

Output:



**Example 2:**

a = 10

b = 20

# Comparing the values

greater\_than = a > b

less\_than\_or\_equal = a <= b

equal\_to = a == b

not\_equal\_to = a != b

# Printing the results

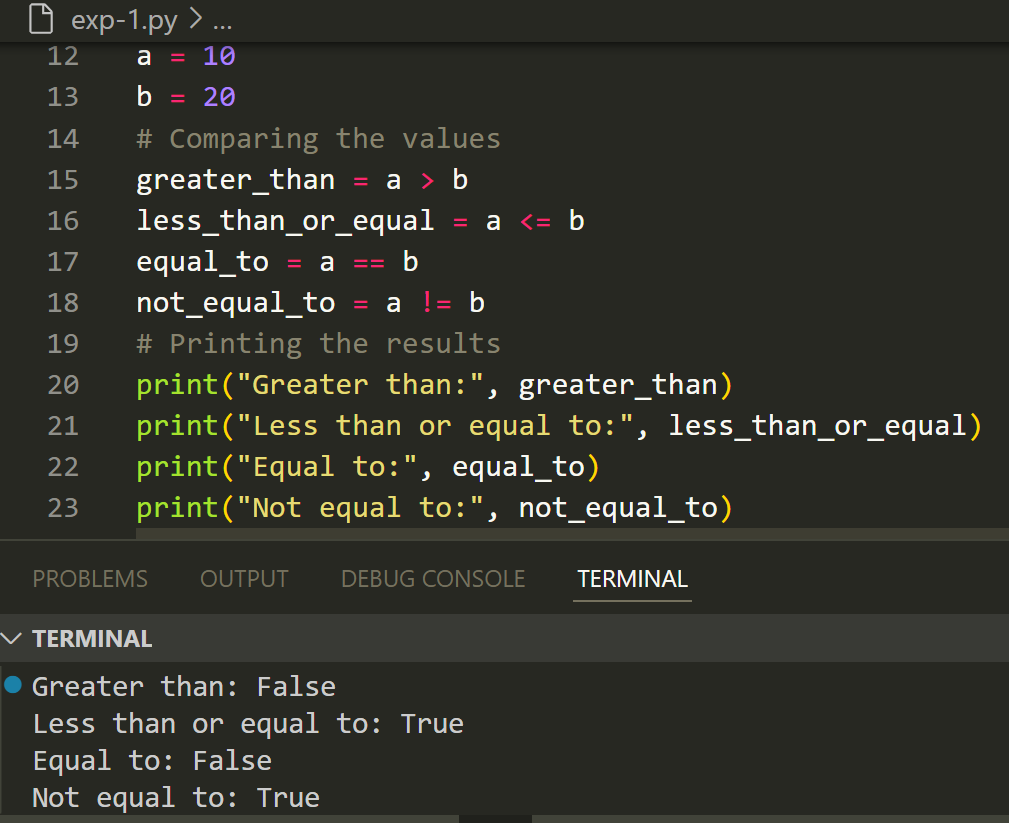
print("Greater than:", greater\_than)

print("Less than or equal to:", less\_than\_or\_equal)

print("Equal to:", equal\_to)

print("Not equal to:", not\_equal\_to)

Output



**Example 3**

x = 3.14

y = 2.5

# Performing arithmetic operations

sum\_result = x + y

difference\_result = x - y

multiplication\_result = x \* y

division\_result = x / y

# Printing the results

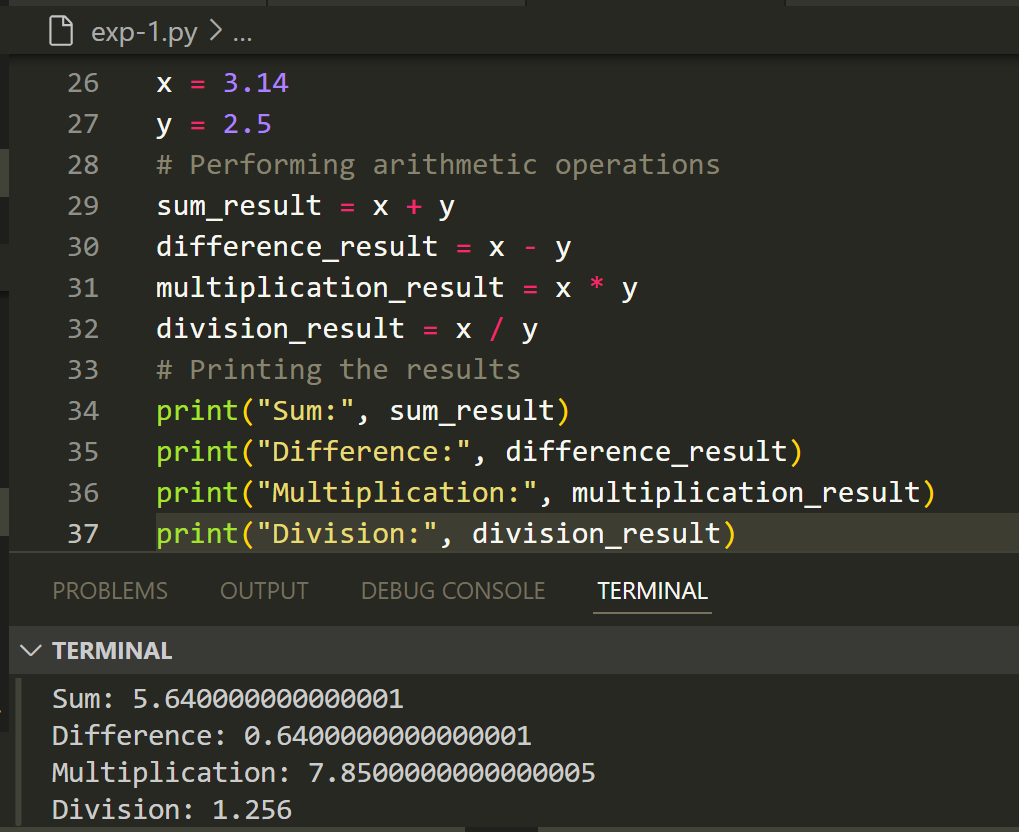
print("Sum:", sum\_result)

print("Difference:", difference\_result)

print("Multiplication:", multiplication\_result)

print("Division:", division\_result)

Output



**Example 4**

a = 1.2

b = 2.7

# Comparing the values

greater\_than = a > b

less\_than\_or\_equal = a <= b

equal\_to = a == b

not\_equal\_to = a != b

# Printing the results

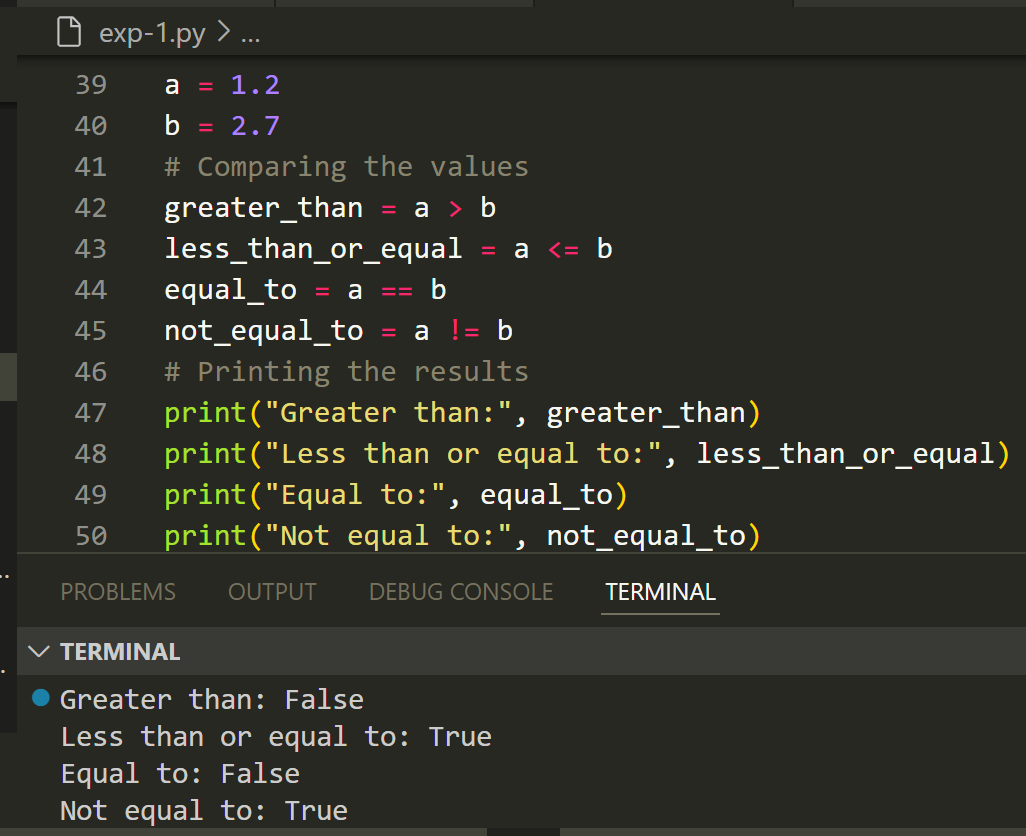
print("Greater than:", greater\_than)

print("Less than or equal to:", less\_than\_or\_equal)

print("Equal to:", equal\_to)

print("Not equal to:", not\_equal\_to)

Output



**Example 5**

x = 2 + 3j

y = -1 + 2j

# Performing arithmetic operations

sum\_result = x + y

difference\_result = x - y

multiplication\_result = x \* y

division\_result = x / y

# Printing the results

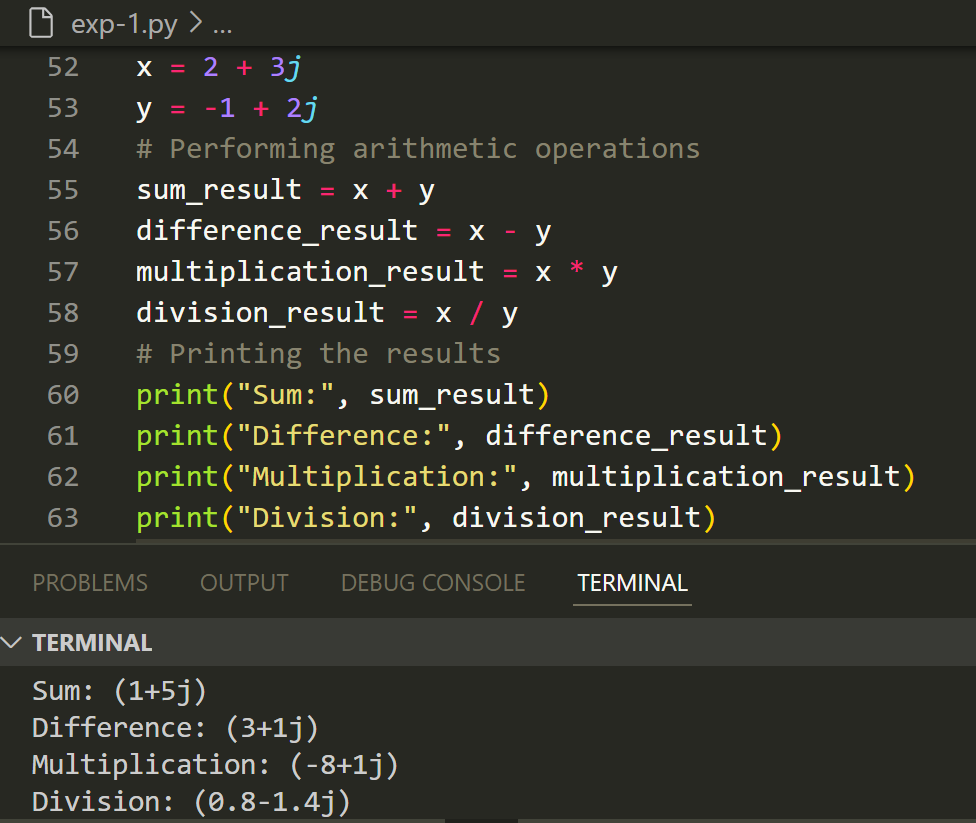
print("Sum:", sum\_result)

print("Difference:", difference\_result)

print("Multiplication:", multiplication\_result)

print("Division:", division\_result)

Output



**Example 6**

a = 1 + 2j

b = 3 + 4j

# Comparing the values

equal\_to = a == b

not\_equal\_to = a != b

# Printing the results

print("Equal to:", equal\_to)

print("Not equal to:", not\_equal\_to)

Output

A screenshot of a computer program

AI-generated content may be incorrect.

**Example 7**

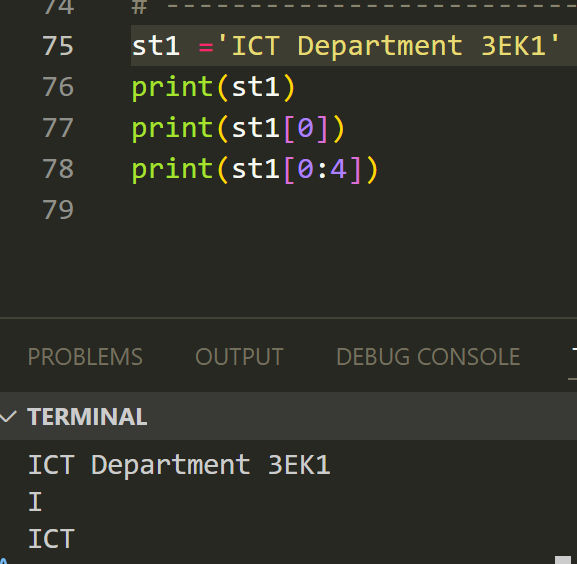
st1 = “ICT Department 3EK1”

print(st1)

print(st1[0])

print(st1[0:4])

Output



**Example 8**

st1 = “ICT”

st2 = “Department”

st3 = “3EK1”

print(st1+st2+st3)

**Repetitions:** Python allows us to repeat a given string with the help of ‘ \* ‘ operator.

print(4\*st1)

Output

A screenshot of a computer program

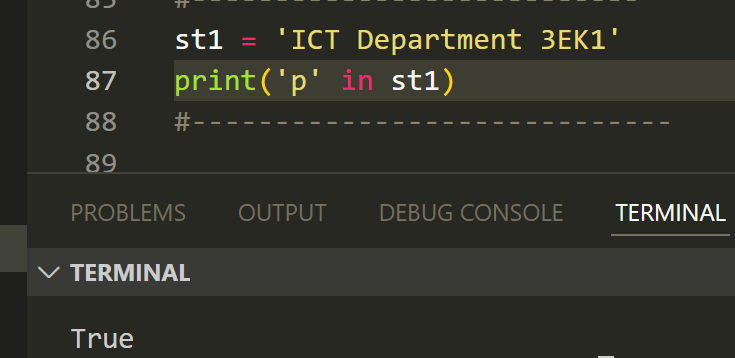
AI-generated content may be incorrect.

**Membership:** The Membership operator helps to check whether a given character is present in the string or not with the help of two operators in and not in. In and not in operator returns the Boolean value True or False.

st1 = “ICT Department 3EK1”

print(“p” in st1)

Output



**Collection Data Types**

Collection data types in Python are used to store and organize multiple values into a single entity. Python provides several built-in collection data types, including lists, tuples, dictionaries, and sets.

list1=[123,567,89]

print(list1)

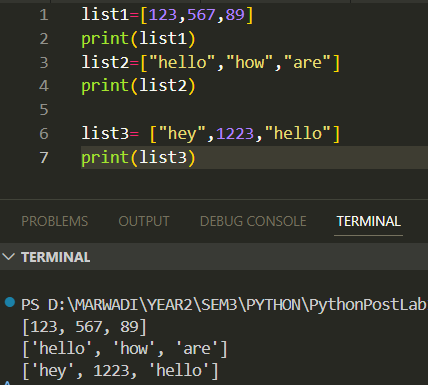
list2=[“hello”,”how”,”are”]

print(list2)

list3= [“hey”,1223,”hello”]

print(list3)

Output



list1=["apple","mango",123,345]

list2 = ["grapes"]

print(list1+ list2)

Output

A screen shot of a computer program

AI-generated content may be incorrect.

dict1={“comp”: “computer” , “sci” : “science”}

print(dict[“comp”])

dict2={“123”:”computer”,456 : “maths”}

print(dict2[“123”])

print(dict1[“comp”]+ dict2[“123”])

Check

print(dict1+ dict2)

print(dict1[“computer”]+ dict2[“computer”])

Output

Dictionaries cannot be added directly.

my\_set = {1, 2, 3, 4, 5}

print(my\_set)

set1 = {1, 2, 3, 4, 5}

set2 = {4, 5, 6, 7, 8}

check

print(set1 + set2)

Output

You cannot use the + operator to combine two sets in Python

my\_tuple = (1, 2, 3, 4, 5)

t1 = (2,3,4)

t2 = (5,6,7)

print(t1+t2)

Output

A screen shot of a computer program

AI-generated content may be incorrect.

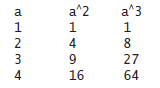
**Post Lab Exercise:**

1. Write a program that displays “Welcome to Python” five times.

**A="Welcome To Python"**

**print(5\*A)**

1. Write a program that displays the following table:



print("Number\tSquare\tCube")

for i in range(1, 5):

print(f"{i}\t{i\*\*2}\t{i\*\*3}")

1. Write a program that displays the result of



**D=(9.5\*4.5)-(2.5\*3)/45.5-3.5**

**print(D)**

**Github :** [**PythonPostLab/1 at main · Om-Lathigara/PythonPostLab**](https://github.com/Om-Lathigara/PythonPostLab/tree/main/1)