

**Lab # 01**

**Student Name: Sahil kumar**

**Roll Number: BIT-23F-006**

**Section: A**

**Subject: Artificial Intelligence**

**Department: BS Information Technology**

**GETTING STARTED WITH PYTHON**

**Lab 1. Introduction to Python**

The Python interpreter is the program that executes Python code. It reads and executes Python scripts or commands line by line.

**Objectives: To get to know about the Variables, Data types, Logical operators (And OR NOT)**

**Variables in python**

Variables in Python are used to store data values. They act as labels that refer to locations in memory where data is stored.

1. Naming: Variable names in Python can consist of letters (both uppercase and lowercase), digits, and underscores. However, they must start with a letter or an underscore. Variable names are case-sensitive.

2. Assignment: Variables are assigned values using the assignment operator `=`. For example:

**x = 10**

**name = "Alice"**

3. Dynamic Typing: Python is dynamically typed, meaning you don't need to declare the type of a variable before assigning a value to it. The type of the variable is inferred based on the value assigned to it. For example:

**x = 10 # x is an integer**

**name = "Alice" # name is a string**

4. Reassignment: Variables can be reassigned to different values of any type. For example:

**x = 10**

**x = "Hello"**

5. Memory Management: Python automatically handles memory allocation and reallocation for variables through its garbage collector.

6. Variable Types: Python supports various data types for variables, including integers, floats, strings, and lists, tuples, dictionaries, sets, and more. Variables can hold values of any of these types.

7. Scope: The scope of a variable determines where in the code that variable can be accessed.

**Data types In Python**

**Numeric Types:**

1. int: Represents integers, e.g., 5, -10, 1000.
2. Float: Represents floating-point numbers, e.g., 3.14, 2.5, -0.1.

**Sequence Types:**

1. Str: Represents strings, which are sequences of characters enclosed in single or double quotes, e.g., 'hello', "world".
2. List: Represents lists, which are ordered collections of items that can be of different types, e.g., [1, 2, 3], ['a', 'b', 'c'].
3. Tuple: Represents tuples, which are similar to lists but are immutable (cannot be modified after creation), e.g., (1, 2, 3), ('a', 'b', 'c').

**Mapping Type:**

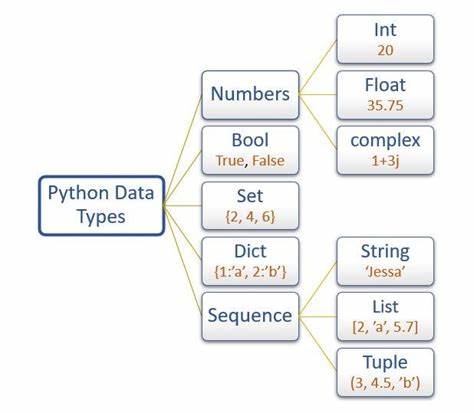
1. Dict: Represents dictionaries, which are collections of key-value pairs, e.g., {'name': 'Alice', 'age': 30}.

**Set Types:**

1. set: Represents sets, which are unordered collections of unique elements, e.g., {1, 2, 3}, { 'apple', 'banana', 'orange'}.

**Boolean Type:**

1. bool: Represents boolean values, which can be either True or False.



**Assignment:**

1. Make 2 programs of each data type.
2. Make 10 shapes using string

**Lab 01**

**1: Make 2 programs of each data type.**

**Numeric Types**:

* **Integer (int)**:

1)

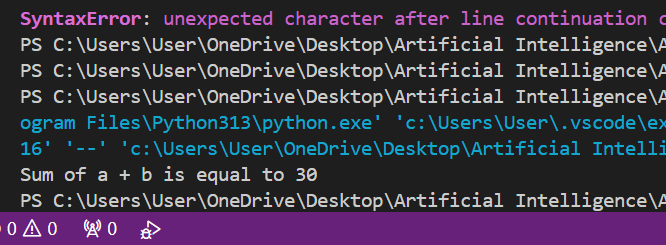
a = 10

b = 20

sum = a + b

print( "Sum of a + b is equal to", sum )

**Output:**



2)

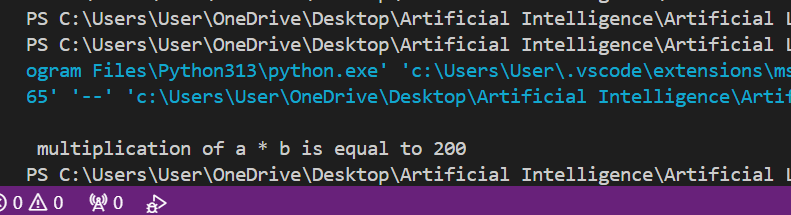
a = 10

b = 20

result = a \* b

print( "multiplication of a \* b is equal to", result )

Output:



* **Float (float)**

**1)**

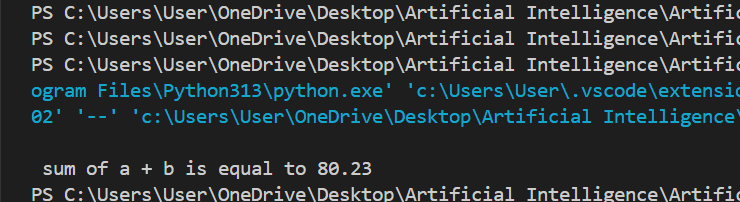
a = 45.56

b = 34.67

result = a + b

print( "\n sum of a + b is equal to", result)

**output:**



**2)**

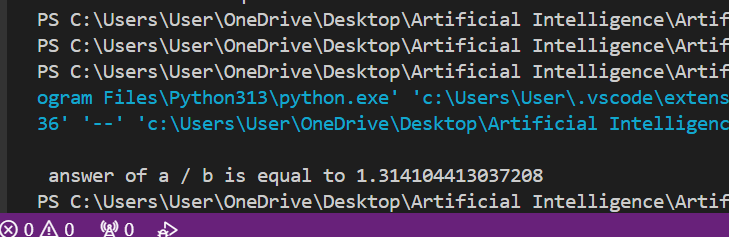
a = 45.56

b = 34.67

result = a / b

print( "\n answer of a / b is equal to", result)

**output:**



* **Complex (complex)**:

**1)**

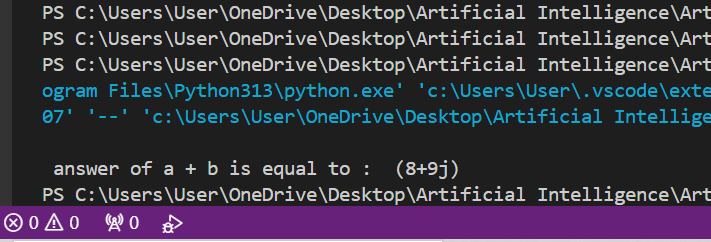
a = 3 + 5j

b = 5 + 4j

result = a + b

print( "\n answer of a + b is equal to : ", result)

output:



2)

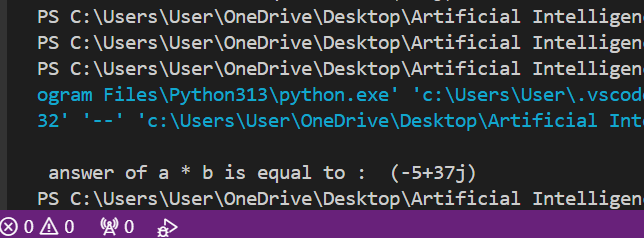
a = 3 + 5j

b = 5 + 4j

result = a \* b

print( "\n answer of a \* b is equal to : ", result)

output:



**Boolean Type**:

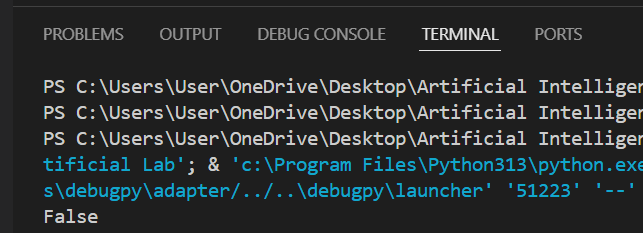
1)

a = 5

b = 10

print(a > b ) # Flase

output:



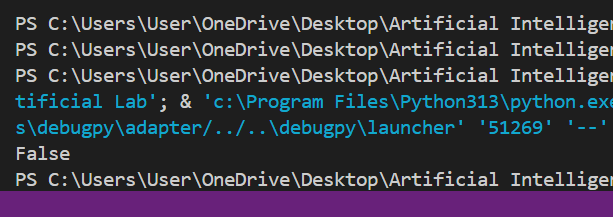
2)

a = 5

b = 10

print(a == b ) # Flase

output:



**Sequence Types**:

* **String (str)**

**1)**

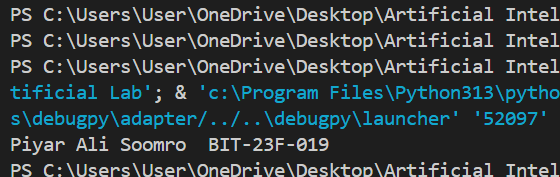
frist\_stirng = "Piyar Ali"

second\_string = "Soomro"

third\_string = "BIT-23F-019"

print(frist\_stirng + " "+ second\_string + "  " + third\_string )

Output:



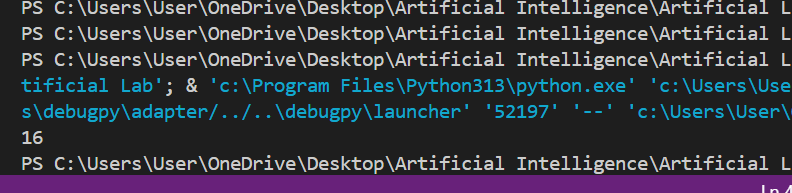
2)

a = "Piyar Ali "

b = "Soomro"

print(len(a) + len(b))

output:



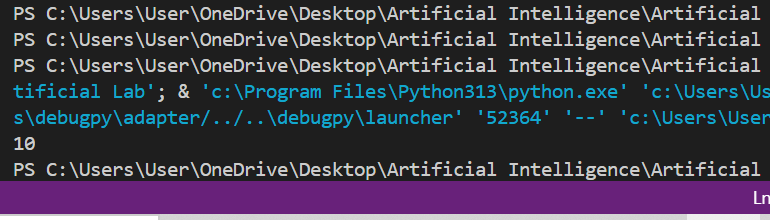
**Tuple (tuple):**

**1)**

my\_tuple = (10, 20, 30)

print(my\_tuple[0])

**output:**

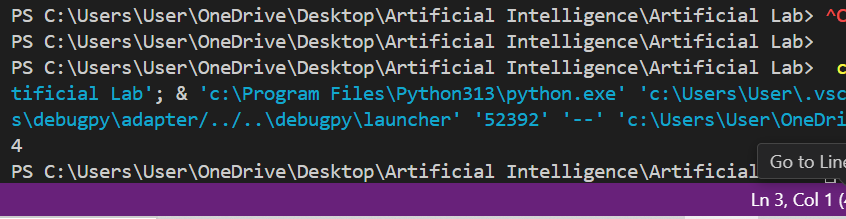


**2)**

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

print(len(my\_tuple))

**output:**



**Mapping Type:**

**1)**

# Creating a simple dictionary

person = {"name": "Piyar Ali Soomro", "age": 21 , "id": 19  }

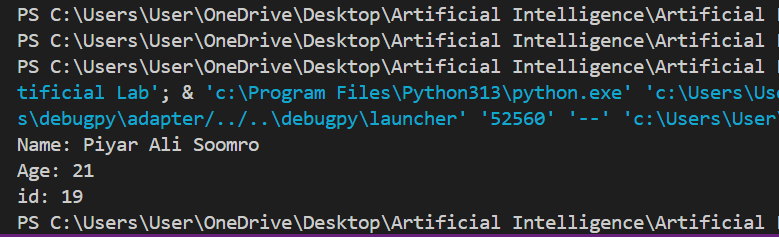
# Accessing values using keys

print("Name:", person["name"])

print("Age:", person["age"])

print("id:",person["id"])

**output:**

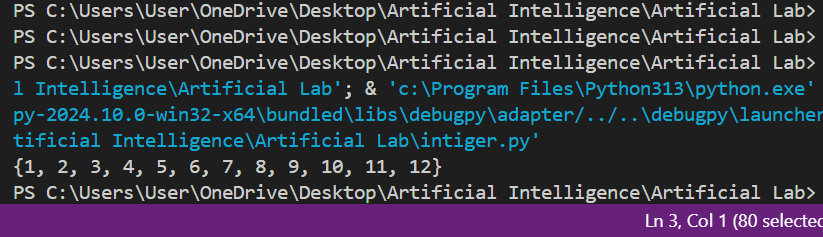


**Set Types**:

set\_of\_Numbers\_set = {1, 2, 3,4,5,6,7,8,9,10,11,12}

print(set\_of\_Numbers\_set)

**output:**



**Boolean (bool):**

a = 5

b = 10

print(a > b)  # False

**2: Make 10 shapes using string:**

**Code:**

# Define number of rows for each shape

n = 5

# 1. Square

print("Square:")

for i in range(n):

    print("\* " \* n)

# 2. Right Triangle

print("\nRight Triangle:")

for i in range(1, n + 1):

    print("\* " \* i)

# 3. Isosceles Triangle

print("\nIsosceles Triangle:")

for i in range(n):

    print(" " \* (n - i - 1) + "\* " \* (i + 1))

# 4. Diamond

print("\nDiamond:")

for i in range(n):

    print(" " \* (n - i - 1) + "\* " \* (i + 1))

for i in range(n - 1):

    print(" " \* (i + 1) + "\* " \* (n - i - 1))

# 5. Hollow Square

print("\nHollow Square:")

for i in range(n):

    if i == 0 or i == n - 1:

        print("\* " \* n)

    else:

        print("\*" + "  " \* (n - 2) + " \*")

# 6. Hollow Right Triangle

print("\nHollow Right Triangle:")

for i in range(n):

    if i == 0 or i == n - 1:

        print("\* " \* (i + 1))

    else:

        print("\*" + "  " \* (i - 1) + " \*")

# 7. Hollow Isosceles Triangle

print("\nHollow Isosceles Triangle:")

for i in range(n):

    if i == 0:

        print(" " \* (n - 1) + "\*")

    elif i == n - 1:

        print("\* " \* (2 \* n - 1))

    else:

        print(" " \* (n - i - 1) + "\*" + " " \* (2 \* i - 1) + "\*")

# 8. X Shape

print("\nX Shape:")

for i in range(n):

    print(" " \* i + "\*" + " " \* (2 \* (n - i - 1) - 1) + "\*" \* (i < n - 1))

# 9. Plus Shape

print("\nPlus Shape:")

for i in range(n):

    if i == n // 2:

        print("\* " \* n)

    else:

        print(" " \* (n - 1) + "\*")

# 10. Hourglass

print("\nHourglass:")

for i in range(n):

    print(" " \* i + "\* " \* (n - i))

for i in range(1, n):

    print(" " \* (n - i - 1) + "\* " \* (i + 1))

**Output:**



