PD LAB

ASSIGNMENT - 9

Name: Raunak Thanawala

Registration Number: 231070051

Branch: Computer Engineering

Batch: 3

**Aim:-**

Study all the datatypes in python.

**Theory:-**

Python provides a variety of built-in data types that are commonly used to represent and manipulate data in programming. Here’s a rundown of the primary built-in data types in Python, excluding user-defined types:

1. **Numeric Types:**
   * **int:**
     + Represents integers, i.e., whole numbers without a decimal point (e.g., 5, -10, 2024).
     + Python’s int type supports arbitrary precision, so you can work with very large integers.
   * **float:**
     + Represents floating-point numbers, i.e., numbers with a decimal point (e.g., 3.14, -0.001, 2.0).
     + Floats are used to represent real numbers and are based on double-precision, 64-bit IEEE 754 floating-point format.
   * **complex:**
     + Represents complex numbers, which have a real and an imaginary part (e.g., 3 + 4j).
     + In Python, j is used to denote the imaginary unit.
2. **Sequence Types**
   * **str:**
     + Represents a sequence of characters (e.g., "Hello, World!").
     + Strings in Python are immutable, meaning they cannot be changed after creation.
   * **list:**
     + Represents an ordered, mutable collection of items (e.g., [1, 2, 3] or ['a', 'b', 'c']).
     + Lists are versatile and can store items of different data types, including other lists.
   * **tuple:**
     + Represents an ordered, immutable collection of items (e.g., (1, 2, 3)).
     + Once created, the elements of a tuple cannot be changed, making it a "read-only" version of a list.
3. **Mapping Type:**
   * **dict:**
     + Represents a collection of key-value pairs (e.g., {'name': 'Alice', 'age': 25}).
     + Dictionaries are mutable, allowing you to add, modify, and remove items.
     + Keys are unique, while values can be of any data type.
4. **Set Types:**
   * **set:**
     + Represents an unordered collection of unique items (e.g.{1, 2, 3}).
     + Sets are mutable, allowing you to add or remove items, but they do not allow duplicates.
   * **frozenset:**
     + Represents an immutable version of a set.
     + Once created, items in a frozenset cannot be added or removed, which makes it useful in situations where a constant set of unique items is needed.
5. **Boolean Type:**
   * **bool:**
     + Represents the truth values True and False.
     + Booleans are often used in conditional statements and can result from comparison operators (e.g., 5 > 3 evaluates to True).
6. **Binary Types:**
   * **bytes:**
     + Represents an immutable sequence of bytes, typically used for binary data (e.g., b'hello').
     + Bytes are particularly useful for handling files or network data.
   * **bytearray:**
     + Represents a mutable sequence of bytes, similar to bytes but can be modified.
   * **memoryview:**
     + Provides a memory-efficient way to access the buffer protocol of an object (e.g., memoryview(bytes\_obj)).
     + It allows you to access slices of binary data without creating copies, which is useful for large binary data processing.

**Code and Output:**

*#All Datatypes*

x = "Hello World"

x = 50

x = 60.5

x = 3j

x = ["geeks", "for", "geeks"]

x = ("geeks", "for", "geeks")

x = range(10)

x = {"name": "Suraj", "age": 24}

x = {"geeks", "for", "geeks"}

x = frozenset({"geeks", "for", "geeks"})

x = True

x = b"Geeks"

x = bytearray(4)

x = memoryview(bytes(6))

x = None

a = 10

print("a = ",a)

print("Type of a: ", type(a))

b = 10.5

print("b = ",b)

print("Type of b: ",type(b))

c = 10 + 5j

print("c = ",c)

print("Type of c: ",type(c))

str1 = 'Hello1'

str2 = "Hello2"

str3 = '''Hello3'''

mstr = ''' Multiline

"And no need for special characters" '''

print(str1)

print(type(str1))

print(str2)

print(type(str2))

print(str3)

print(type(str3))

print(mstr)

print(type(mstr))

List = [1,2,3,4,5]

MultiList = [[1,2],[3,4],[5]]

print(List)

print(type(List))

print(MultiList)

print(type(MultiList))

Tuple = (1,2,3,4,5,6)

Tuple1 = ('a','b','c')

Tuple2 = (10,20,30)

NesTuple = (Tuple1, Tuple2)

print(Tuple)

print(type(Tuple))

print(NesTuple)

print(type(NesTuple))

t = True

f = False

print(t)

print(type(t))

print(f)

print(type(f))

Set = set([1,1,1,1,'a',3,43,'a',4,2,3,234])

print(Set)

print(type(Set))

SSet = set("Geeks for Geeks for")

print(SSet)

print(type(SSet))

FSet = frozenset([1,1,2,3,4,5,6,'a',122,50.5,'5'])

print(FSet)

print(type(FSet))

Dict = {1:'G', 2:'e', 3:'k', 4:'e', 5:'s'}

print(Dict)

print(type(Dict))

non = None

print(non)

print(type(non))

bite = b"Hi"

print(bite)

print(type(bite))

bitearr = bytearray(b"Hi")

print(bitearr)

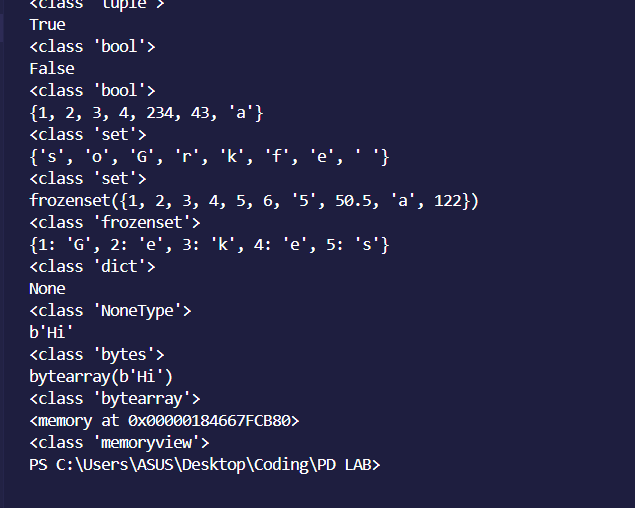
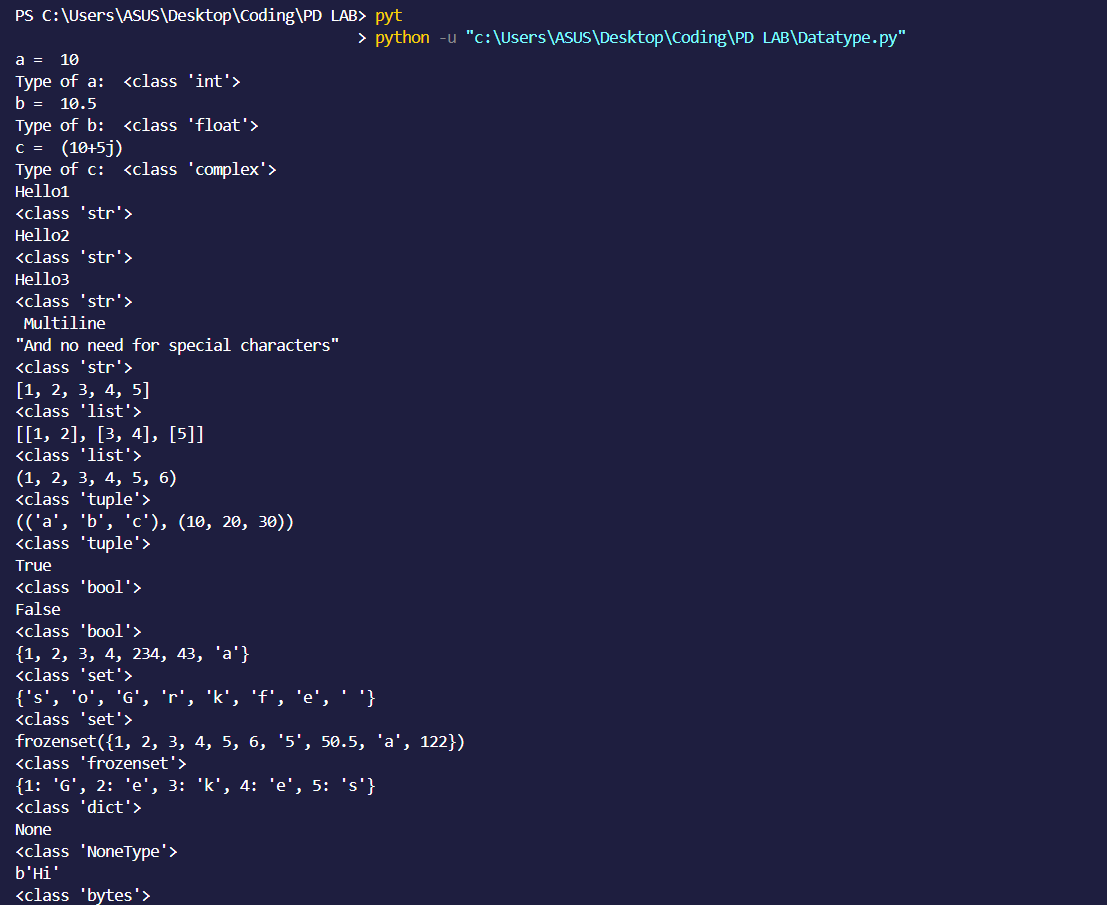
print(type(bitearr))

memview = memoryview(bite)

print(memview)

print(type(memview))

**OUTPUT:**



**CONCLUSION:**

Thus we have written a program that shows the various datatypes illustrated in python.

We have also given a brief explanation about all the datatypes in python.