Aim: To implement the basic data types and control structures in python.

Theory:

Python has the following data types built-in by default, in these categories

- Text Type: Str
- Numeric Types: int, float, complex
- Sequence Types: list, tuple, range
- Mapping Type: Dict
- Set Types: set, frozenset
- Boolean Type: Bool
- Binary Types: bytes, bytearray, memoryview

Getting the Data Type

You can get the data type of any object by using the type() function.

Casting

There can be two types of Type Casting in Python –

• Implicit Type Casting

• Explicit Type Casting

Implicit Type Conversion

In this, methods, Python converts data type into another data type automatically. In this process, users don't have to involve in this process.

Explicit Type Casting

In this method, Python need user involvement to convert the variable data type into certain data type in order to the operation required.

Mainly in type casting can be done with these data type function:

• **Int() :** Int() function take float or string as an argument and return int type object.

• **float() :** float() function take int or string as an argument and return float type object.

• **str() :** str() function take float or int as an argument and return string type object.

Implementation:

Strings in Python

In [5]:

```
# Assigning string to a variable
a = 'This is a string'
print (a)
b = "This is a string"
print (b)
c= '''This is a string'''
print (c)
```

```
This is a string
This is a string
This is a string
```

Lists in Python

In [6]:

```
L = [1, "a" , "string" , 1+2]
print (L)
#Adding an element in the list
L.append(6)
print (L)
#Deleting last element from a list
L.pop()
print (L)
#Displaying Second element of the list
print (L[1])
```

```
[1, 'a', 'string', 3]
[1, 'a', 'string', 3, 6]
[1, 'a', 'string', 3]
a
```

Tuples in Python

In [7]:

```
tup = (1, "a", "string", 1+2)
print(tup)
print(tup[1])
```

```
(1, 'a', 'string', 3)
a
```

Dictionaries in Python

A Python dictionary is a data structure that stores the value in key: value pairs. Values in a dictionary can be of any data type and can be duplicated, whereas keys can't be repeated and must be immutable.

In [6]:

```
d = {1: 'Lorem', 2: 'Ipsum', 3: 'Dolerum'}
print(d)
```

```
{1: 'Lorem', 2: 'Ipsum', 3: 'Dolerum'}
```

Create a Dictionary

In [7]:

```
# create dictionary using { }
d1 = {1: 'Game', 2: 'of', 3: 'Thrones'}
print(d1)

# create dictionary using dict() constructor
d2 = dict(a = "House", b = "of", c = "Cards")
print(d2)
```

```
{1: 'Game', 2: 'of', 3: 'Thrones'}
{'a': 'House', 'b': 'of', 'c': 'Cards'}
```

Accessing Dictionary Items

In [8]:

```
d = { "name": "Alice", 1: "Python", (1, 2): [1,2,4] }

# Access using key
print(d["name"])

# Access using get()
print(d.get("name"))
```

```
Alice
Alice
```

Adding and Updating Dictionary Items

In [8]:

```
d = {1: 'Game', 2: 'of', 3: 'Thrones'}

# Adding a new key-value pair
d[4] = "age"

# Updating an existing value
d[1] = "Python dict"

print(d)
```

```
{1: 'Python dict', 2: 'of', 3: 'Thrones', 4: 'age'}
```

Conclusion:

Understanding basic data types and control structures in Python is fundamental for effective programming. Python offers a wide variety of built-in data types such as integers, floats, strings, lists, dictionaries, and more, allowing flexibility in data handling. Using the type() function helps identify an object's data type, which is useful in debugging and development. Type casting plays a key role in converting between data types—either implicitly, handled by Python automatically, or explicitly, where the user manually converts types using functions like int(), float(), and str(). Mastering these basics lays a strong foundation for building more complex Python programs.