assignment-2

July 18, 2025

[1]:
```python
# Python program to
# demonstrate numeric value
a = 6
print("Type of a: ", type(a))
b = 8.0
print("\nType of b: ", type(b))
c = 2 + 9j
print("\nType of c: ", type(c))
```

```
Type of a:  <class 'int'>

Type of b:  <class 'float'>

Type of c:  <class 'complex'>
```

[2]:
```python
# Creating a String
# with single Quotes
String1 = 'Welcome Sudip'
print("Representing : ")
print(String1)
```

```
Representing :
Welcome Sudip
```

[3]:
```python
# Creating a String
# with double Quotes
String1 = " Sudip Madhu"
print("\n A student of GCECT: ")
print(String1)
print(type(String1))
```

```

 A student of GCECT:
 Sudip Madhu
<class 'str'>
```

[4]:
```python
#Creating String with triple
# Quotes allows multiple lines
```

```
String1 = '''Welcome
Sudip '''
print("\nCreating a multiline String. ")
print(String1)
```

```

Creating a multiline String.
Welcome
Sudip
```

[5]:
```
# Python Program to Access
# characters of String
String1 = "Abc"
print("Initial String: ")
print(String1)
# Printing First character
print("\nFirst character of String is: ")
print(String1[0])
# Printing Last character
print("\nLast character of String is: ")
print(String1[-1])
```

```
Initial String:
Abc

First character of String is:
A

Last character of String is:
c
```

[6]:
```
# Creating a List
List = []
print("Initial blank List: ")
print(List)
# Creating a List with
# the use of multiple values
List = ["A", "B", "C", "D"]
print("\nList containing multiple values: ")
print(List[0])
print(List[3])
# Creating a Multi-Dimensional List
# (By Nesting a list inside a List)
List = [['A', 'B'], ['C']]
print("\nMulti-Dimensional List: ")
print(List)
```

```
Initial blank List:
```

```
[]

List containing multiple values:
A
D

Multi-Dimensional List:
[['A', 'B'], ['C']]
```

[7]:
```python
List = ["A", "B", "C"]
# accessing a element
print("Accessing element from the list")
print(List[0])
print(List[2])
# negative indexing
print("Accessing element using negative indexing")
# print the last element of list
print(List[-1])
# print the third last element of list
print(List[-3])
```

```
Accessing element from the list
A
C
Accessing element using negative indexing
C
A
```

[8]:
```python
# Creating a tuple
tuple1 = tuple([1, 2, 3, 4, 5])
# Accessing element using indexing
print("First element of tuple")
print(tuple1[0])
# negative indexing
print("\nLast element of tuple")
print(tuple1[-1])
print("\nThird last element of tuple")
print(tuple1[-3])
```

```
First element of tuple
1

Last element of tuple
5

Third last element of tuple
3
```

```
[9]: print(type(True))
     print(type(False))
     print(type(true))
```

```
<class 'bool'>
<class 'bool'>
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
/tmp/ipython-input-9-2996133441.py in <cell line: 0>()
      1 print(type(True))
      2 print(type(False))
----> 3 print(type(true))

NameError: name 'true' is not defined
```

```
[10]: # Creating an empty Dictionary
      Dict = {}
      print("Empty Dictionary: ")
      print(Dict)
      # with Integer Keys
      Dict = {1: 'A', 2: 'B', 3: 'C'}
      print("\nDictionary with the use of Integer Keys: ")
      print(Dict)
      # with Mixed keys
      Dict = {'Name': 'A', 1: [1, 2, 3, 4]}
      print("\nDictionary with the use of Mixed Keys: ")
      print(Dict)
      # with dict() method
      Dict = dict({1: 'A', 2: 'B', 3:'C'})
      print("\nDictionary with the use of dict(): ")
      print(Dict)
      # with each item as a Pair
      Dict = dict([(1, 'A'), (2, 'B')])
```

```
Empty Dictionary:
{}

Dictionary with the use of Integer Keys:
{1: 'A', 2: 'B', 3: 'C'}

Dictionary with the use of Mixed Keys:
{'Name': 'A', 1: [1, 2, 3, 4]}

Dictionary with the use of dict():
{1: 'A', 2: 'B', 3: 'C'}
```

```
[11]: # Creating a Dictionary
Dict = {1: 'A', 'name': 'B', 3: 'C'}
# accessing a element using key
print("Accessing a element using key:")
print(Dict['name'])
# accessing a element using get()
# method
print("Accessing a element using get:")
print(Dict.get(3))
```

```
Accessing a element using key:
B
Accessing a element using get:
C
```

```
[12]: a = 9
b = 4
add = a + b
sub = a - b
mul = a * b
div1 = a / b
div2 = a // b
mod = a % b
p = a ** b
print(add)
print(sub)
print(mul)
print(div1)
print(div2)
print(mod)
print(p)
```

```
13
5
36
2.25
2
1
6561
```

```
[13]: # Relational Operators
a = 13
b = 33

print(a > b)
print(a < b)
print(a == b)
print(a != b)
print(a >= b)
print(a <= b)
```

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

[14]:
```
# Examples of Logical Operator
a = True
b = False
# Print a and b is False
print(a and b)
# Print a or b is True
print(a or b)
# Print not a is False
print(not a)
```

```
False
True
False
```

[15]:
```
# Assignment Operators
a = 10
# Assign value
b = a
print(b)
# Add and assign value
b += a
print(b)
# Subtract and assign value
b -= a
print(b)
# multiply and assign
b *= a
print(b)
```

```
10
20
10
100
```

[16]:
```
a = 10
b = 20
```

```
c = a
print(a is not b)
print(a is c)
```

```
True
True
```

[18]:
```
#membership operator
x = 24
y = 20
list = [10, 20, 30, 40, 50]
if (x not in list):
  print("x is NOT present in given list")
else:
  print("x is present in given list")
if (y in list):
  print("y is present in given list")
else:
  print("y is NOT present in given list")
```

```
x is NOT present in given list
y is present in given list
```