St. Francis Institute of Technology, Mumbai-400 103
Department Of Information Technology

A.Y. 2023-2024
Class: SE-ITA/B, Semester: IV
Subject: **Python Lab**

Experiment – 1: Python Data Types

Aim: To implement a Python program for the following data types:

a. Write a Python program to demonstrate numeric data types.
b. Write a Python program to demonstrate any 10 methods of List.
c. Write a Python program to demonstrate working on tuples and any methods of Tuple.
d. Write a Python program to demonstrate any 10 methods of Dictionary.

1. **Objectives:** After performing this experiment, a student will be able to write a basic Python program using Python data types like Numeric, Sequences, Sets and Dictionaries.
2. **Prerequisite:** Python basics
3. **Requirements:** PC, Python 3.9, Windows 10/ MacOS/ Linux, IDLE IDE
4. **Pre-Experiment Exercise:**
5. **Theory:**

A data type represents the type of data stored into a variable or memory. The datatypes which are already available in Python are called built-in data types. The datatypes created by the programmer is called user-defined datatypes.

The built-in data types are as follows:

- **Numeric**: The numeric data types represents numbers. There are three sub categories of numeric data types
- **int** : int are positive or negative whole numbers with no decimal point. Integers in Python 3 are of unlimited size.
- **float** : represent real numbers and are written with a decimal point dividing the integer and the fractional parts. Floats may also be in scientific notation, with E or e indicating the powerof 10 (2.5e2 = 2.5 x 10^2 = 250).
- **complex** : are of the form a + bJ, where a and b are floats and J (or j) represents the squareroot of -1 (which is an imaginary number). The real part of the number is a, and theimaginary part is b. Complex numbers are not used much in Python programming.
- **Sequences:** A sequence represents a group of elements or items. There are six types of sequences in Python.
- **str:** represents string datatype which is string of characters. Strings are constructed by using single or double quotes.
- **bytes:** represents a group of byte numbers. A byte is any positive number between 0 to 255.
- **Byte array:** similar to array of bytes. But the difference is array of bytes cannot be modified but byte array type array can be modified.

- **List:** represents a group of elements. Lists can grow dynamically in memory. Lists are represented using square brackets and its elements are separated by commas.
- **Tuple:** contains a group of elements which can be of different types. The elements in the tuple are separated by commas and enclosed in parenthesis. Whereas the elements of a list can be modified, it is not possible to modify the tuple elements. A tuple can be treated as a read-only list.
- **range:** represents a sequence of numbers. The numbers in the range are not modifiable.
- **Sets:** A set is an unordered, mutable collection of elements. Common uses include membership testing, removing duplicates from a sequence, and computing standard math operations on sets such as intersection, union, difference *etc.* To create a set, elements separated by commas are entered using curly braces. the same notation is used in Python.
- **frozenset:** is similar to set except elements of frozenset cannot be modified.
- **Dictionary**: represents a group of elements arranged in the form of key value pairs. In the dictionary, the first element is considered as a 'key' and the immediate next value is considered as its 'value'. The key and its value is separated by a colon. All the key value pairs are inserted in curly braces. Various methods are available to access and process the elements of a dictionary.

6. Laboratory Exercise

A. Procedure

i. Open IDE for Python programming
ii. Open new Python file from menu file-new
iii. Type Python code with proper syntax
iv. Save file with .py extension
v. Execute the command statements inside the saved file using cntr+enter key and explore results in other windows of IDE.
vi. Add relevant comments in your programs and execute the code.

Test it for various cases.

a. Write a Python program to demonstrate numeric data types.
b. CODE:

7. Post-Experiments Exercise:

A. Extended Theory:

a. How can we determine the datatype of any python variable?
b. List down various naming conventions in Python.

B. Post Lab Program:

a. Write a Python program to implement three different syntaxes of range function.
b. Write a Python program to implement any 10 methods of Set.

C. Conclusion:

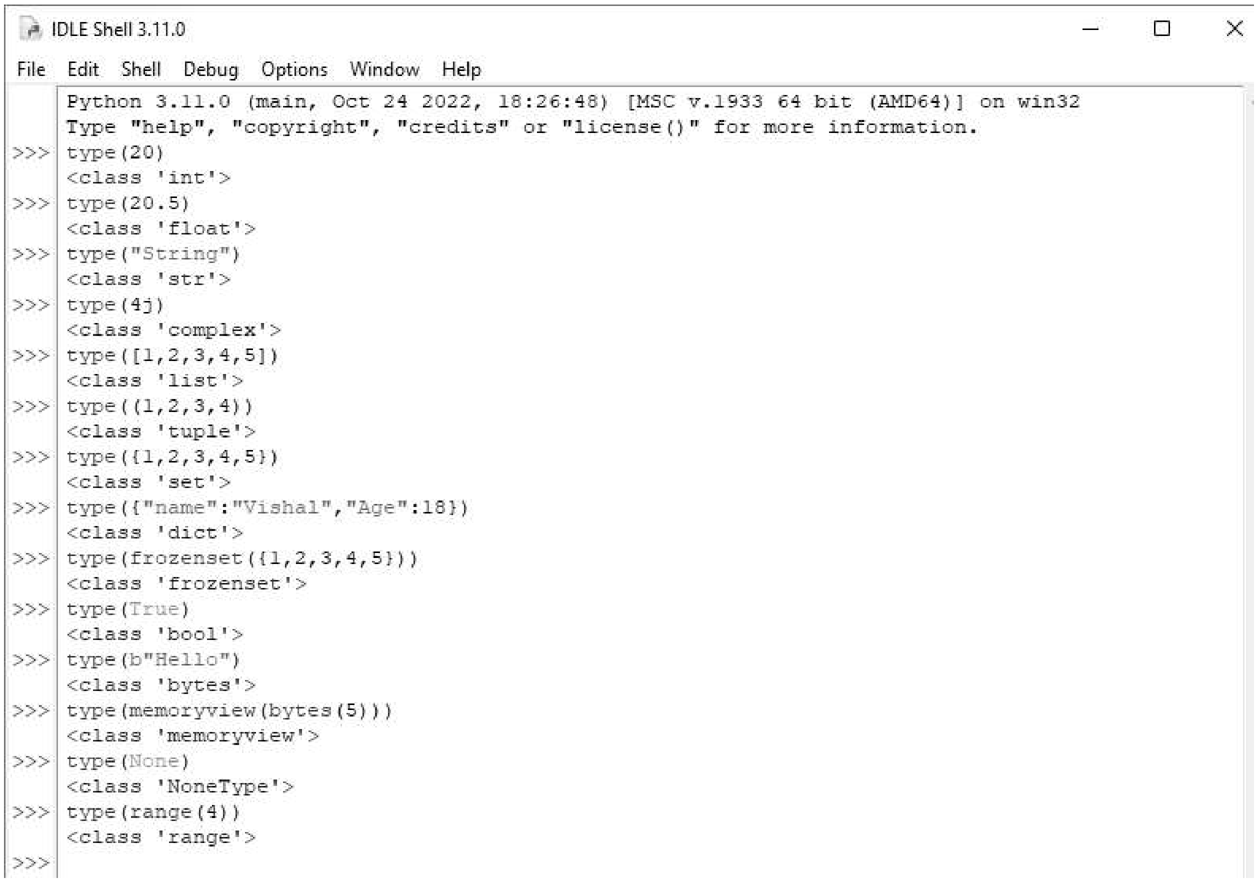
1. Write what was performed in the program (s).
2. What is the significance of program (s)?

D. References:

[1] Dr. R. Nageswara Rao," Core Python Programming" , Dreamtech Press, Wiley Publication
[2] https://www.pythonforbeginners.com/

In-Lab Program

a.Write a Python program to demonstrate numeric data types



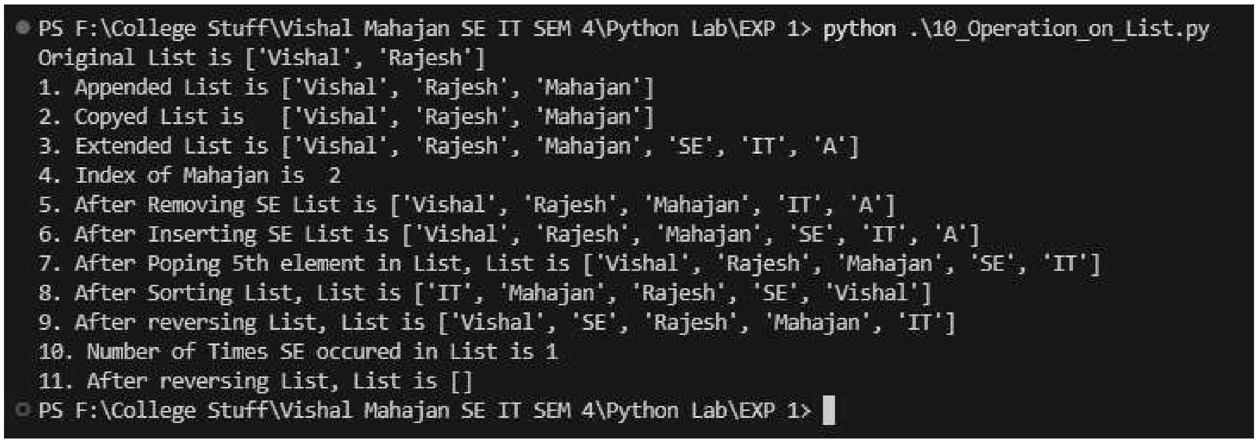
```
Python 3.11.0 (main, Oct 24 2022, 18:26:48) [MSC v.1933 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> type(20)
<class 'int'>
>>> type(20.5)
<class 'float'>
>>> type("String")
<class 'str'>
>>> type(4j)
<class 'complex'>
>>> type([1,2,3,4,5])
<class 'list'>
>>> type((1,2,3,4))
<class 'tuple'>
>>> type({1,2,3,4,5})
<class 'set'>
>>> type({"name":"Vishal","Age":18})
<class 'dict'>
>>> type(frozenset({1,2,3,4,5}))
<class 'frozenset'>
>>> type(True)
<class 'bool'>
>>> type(b"Hello")
<class 'bytes'>
>>> type(memoryview(bytes(5)))
<class 'memoryview'>
>>> type(None)
<class 'NoneType'>
>>> type(range(4))
<class 'range'>
>>>
```

b.Write a Python program to demonstrate any 10 methods of List.

```
#List
#Declearing a List
EXP1= ["Vishal","Rajesh"]
print("Original List is",EXP1)

#Built-in Functions on List
#1. Append() : Use to Add elements to the end of List
EXP1.append("Mahajan")
print("1. Appended List is",EXP1)

#2. Copy() :  Returns a copy of the list
CopyEXP1 = EXP1.copy()
print("2. Copyed List is  ",CopyEXP1)

#3. Extend() :Add the elements of a list (or any iterable) to
the end of the current list
Extended_List=["SE","IT","A"]
EXP1.extend(Extended_List)
print("3. Extended List is",EXP1)

#4. Index() : Return lowest index where elements appears
index_of_mahajan=EXP1.index("Mahajan")
print("4. Index of Mahajan is ",index_of_mahajan)

#5. Remove() : Removes a given object from the List
EXP1.remove("SE")
print("5. After Removing SE List is",EXP1)

#6. Insert() : Adds an element at the specified position
EXP1.insert(3,"SE")
print("6. After Inserting SE List is",EXP1)
```

```
#7 Pop() : Removes the element at the specified position
EXP1.pop(5)
print("7. After Poping 5th element in List, List is",EXP1)

#8 Sort():Sorts the list
EXP1.sort()
print("8. After Sorting List, List is",EXP1)

#9. Reverse(): Reverses the order of the list
EXP1.reverse()
print("9. After reversing List, List is",EXP1)

#10 Count() : Returns the number of elements with specified
value
count_of_SE=EXP1.count("SE")
print("10. Number of Times SE occured in List is",count_of_SE)

#11 Clear() ; Removes all the Elements from the List
EXP1.clear()
print("11. After reversing List, List is",EXP1)
```

Output:



```
PS F:\College Stuff\Vishal Mahajan SE IT SEM 4\Python Lab\EXP 1> python .\10_Operation_on_List.py
Original List is ['Vishal', 'Rajesh']
1. Appended List is ['Vishal', 'Rajesh', 'Mahajan']
2. Copyed List is   ['Vishal', 'Rajesh', 'Mahajan']
3. Extended List is ['Vishal', 'Rajesh', 'Mahajan', 'SE', 'IT', 'A']
4. Index of Mahajan is  2
5. After Removing SE List is ['Vishal', 'Rajesh', 'Mahajan', 'IT', 'A']
6. After Inserting SE List is ['Vishal', 'Rajesh', 'Mahajan', 'SE', 'IT', 'A']
7. After Poping 5th element in List, List is ['Vishal', 'Rajesh', 'Mahajan', 'SE', 'IT']
8. After Sorting List, List is ['IT', 'Mahajan', 'Rajesh', 'SE', 'Vishal']
9. After reversing List, List is ['Vishal', 'SE', 'Rajesh', 'Mahajan', 'IT']
10. Number of Times SE occured in List is 1
11. After reversing List, List is []
PS F:\College Stuff\Vishal Mahajan SE IT SEM 4\Python Lab\EXP 1>
```

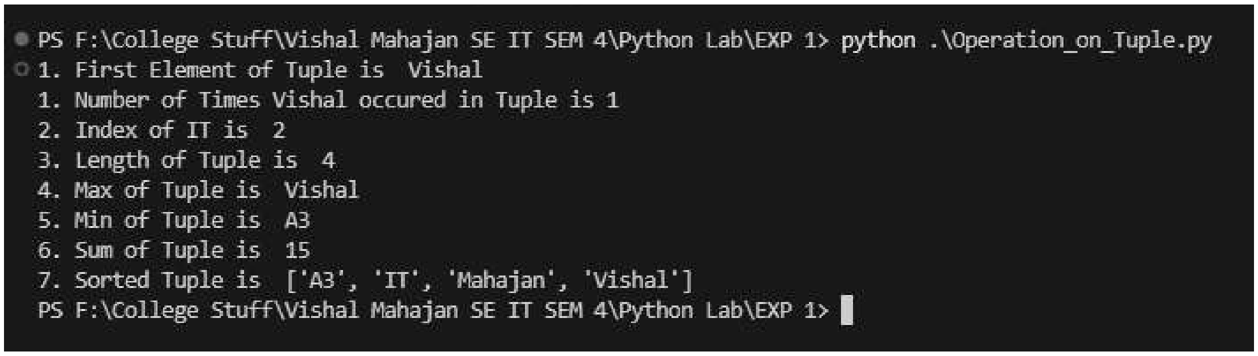
c. Write a Python program to demonstrate working on tuples and any methods of Tuple.

Code:

```
# Create a tuple
EXP1 = ("Vishal","Mahajan","IT","A3")

#Accessing Elements of Tuple
print("1. First Element of Tuple is ",EXP1[0])

#Built-in Functions on Tuple
#1. Count() : Returns the number of times a specified value
occurs in a tuple
count_of_Vishal=EXP1.count("Vishal")
print("1. Number of Times Vishal occured in Tuple
is",count_of_Vishal)

#2. Index() : Searches the tuple for a specified value and
returns the position of where it was found
index_of_IT=EXP1.index("IT")
print("2. Index of IT is ",index_of_IT)

#3. len() : Returns the length of the tuple
length_of_tuple=len(EXP1)
print("3. Length of Tuple is ",length_of_tuple)

#4. max() : Returns the largest value in a tuple
max_of_tuple=max(EXP1)
print("4. Max of Tuple is ",max_of_tuple)

#5. min() : Returns the smallest value in a tuple
min_of_tuple=min(EXP1)
print("5. Min of Tuple is ",min_of_tuple)
```

```
#6. sum() : Sums the items of an iterator
EXP1_int=(1,2,3,4,5)
sum_of_tuple=sum(EXP1_int)
print("6. Sum of Tuple is ",sum_of_tuple)

#7. sorted() : Returns a sorted list of the specified iterable
object
sorted_tuple=sorted(EXP1)
print("7. Sorted Tuple is ",sorted_tuple)
```

Output:



```
PS F:\College Stuff\Vishal Mahajan SE IT SEM 4\Python Lab\EXP 1> python .\Operation_on_Tuple.py
1. First Element of Tuple is  Vishal
1. Number of Times Vishal occured in Tuple is 1
2. Index of IT is  2
3. Length of Tuple is  4
4. Max of Tuple is  Vishal
5. Min of Tuple is  A3
6. Sum of Tuple is  15
7. Sorted Tuple is  ['A3', 'IT', 'Mahajan', 'Vishal']
PS F:\College Stuff\Vishal Mahajan SE IT SEM 4\Python Lab\EXP 1>
```

d. Write a Python program to demonstrate any 10 methods of Dictionary.

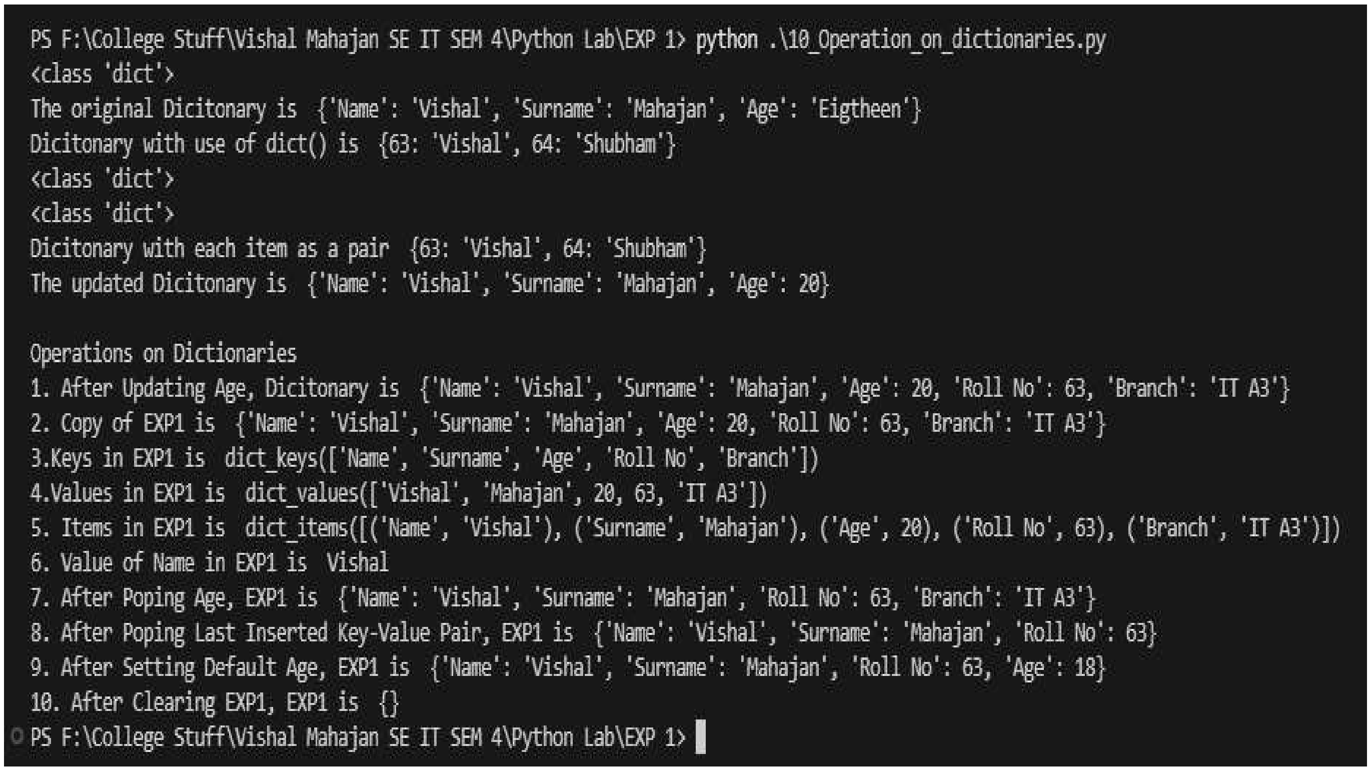
Code:

```python
#Python Dictionaries

#Three ways to create Dictionaries
#Type 1
type1={"Name":"Vishal","Surname":"Mahajan","Age":"Eigtheen"}
print(type(type1))
print("The original Dicitonary is ",type1)

#Type2
type2= dict ({63:"Vishal",64:"Shubham"})
print("Dicitonary with use of dict() is ",type2)
print(type(type2))

#Type3
type3= dict([(63,"Vishal"),(64,"Shubham")])
print(type(type3))
print("Dicitonary with each item as a pair ",type3)

#Change the Values
type1["Age"]=20
print("The updated Dicitonary is ",type1)



#Printing all keys
for keys in type1:
    print(keys)

#Printing all values of keys
for values in type1:
    print(type1[values])
```

```
#Operations on Dictionaries
EXP1={"Name":"Vishal","Surname":"Mahajan","Age":18,"Roll
No":63,"Branch":"IT A3"}

#1. Update() : Updates the dictionary with the specified
key-value pairs
EXP1.update({"Age":20})
print("1. After Updating Age, Dicitonary is ",EXP1)

#2. copy() : Returns a copy of the dictionary
CopyEXP1=EXP1.copy()
print("2. Copy of EXP1 is ",CopyEXP1)

#3.Keys() : Returns a list containing the dictionary's keys
print("3.Keys in EXP1 is ",EXP1.keys())

#4.Values() : Returns a list of all the values in the dictionary
print("4.Values in EXP1 is ",EXP1.values())

#5. Items() : Returns a list containing a tuple for each key
value pair
print("5. Items in EXP1 is ",EXP1.items())

#6. Get() : Returns the value of the specified key
print("6. Value of Name in EXP1 is ",EXP1.get("Name"))

#7. Pop() : Removes the element with the specified key
EXP1.pop("Age")
print("7. After Poping Age, EXP1 is ",EXP1)

#8. Popitem() : Removes the last inserted key-value pair
EXP1.popitem()
print("8. After Poping Last Inserted Key-Value Pair, EXP1 is
",EXP1)
```

```
#9. Setdefault() : Returns the value of the specified key. If
the key does not exist: insert the key, with the specified value
EXP1.setdefault("Age",18)
print("9. After Setting Default Age, EXP1 is ",EXP1)

#10. Clear() : Removes all the elements from the dictionary
EXP1.clear()
print("10. After Clearing EXP1, EXP1 is ",EXP1)
```

Output:



```
PS F:\College Stuff\Vishal Mahajan SE IT SEM 4\Python Lab\EXP 1> python .\10_Operation_on_dictionaries.py
<class 'dict'>
The original Dicitonary is  {'Name': 'Vishal', 'Surname': 'Mahajan', 'Age': 'Eigtheen'}
Dicitonary with use of dict() is  {63: 'Vishal', 64: 'Shubham'}
<class 'dict'>
<class 'dict'>
Dicitonary with each item as a pair  {63: 'Vishal', 64: 'Shubham'}
The updated Dicitonary is  {'Name': 'Vishal', 'Surname': 'Mahajan', 'Age': 20}

Operations on Dictionaries
1. After Updating Age, Dicitonary is  {'Name': 'Vishal', 'Surname': 'Mahajan', 'Age': 20, 'Roll No': 63, 'Branch': 'IT A3'}
2. Copy of EXP1 is  {'Name': 'Vishal', 'Surname': 'Mahajan', 'Age': 20, 'Roll No': 63, 'Branch': 'IT A3'}
3.Keys in EXP1 is  dict_keys(['Name', 'Surname', 'Age', 'Roll No', 'Branch'])
4.Values in EXP1 is  dict_values(['Vishal', 'Mahajan', 20, 63, 'IT A3'])
5. Items in EXP1 is  dict_items([('Name', 'Vishal'), ('Surname', 'Mahajan'), ('Age', 20), ('Roll No', 63), ('Branch', 'IT A3')])
6. Value of Name in EXP1 is  Vishal
7. After Poping Age, EXP1 is  {'Name': 'Vishal', 'Surname': 'Mahajan', 'Roll No': 63, 'Branch': 'IT A3'}
8. After Poping Last Inserted Key-Value Pair, EXP1 is  {'Name': 'Vishal', 'Surname': 'Mahajan', 'Roll No': 63}
9. After Setting Default Age, EXP1 is  {'Name': 'Vishal', 'Surname': 'Mahajan', 'Roll No': 63, 'Age': 18}
10. After Clearing EXP1, EXP1 is  {}
PS F:\College Stuff\Vishal Mahajan SE IT SEM 4\Python Lab\EXP 1>
```

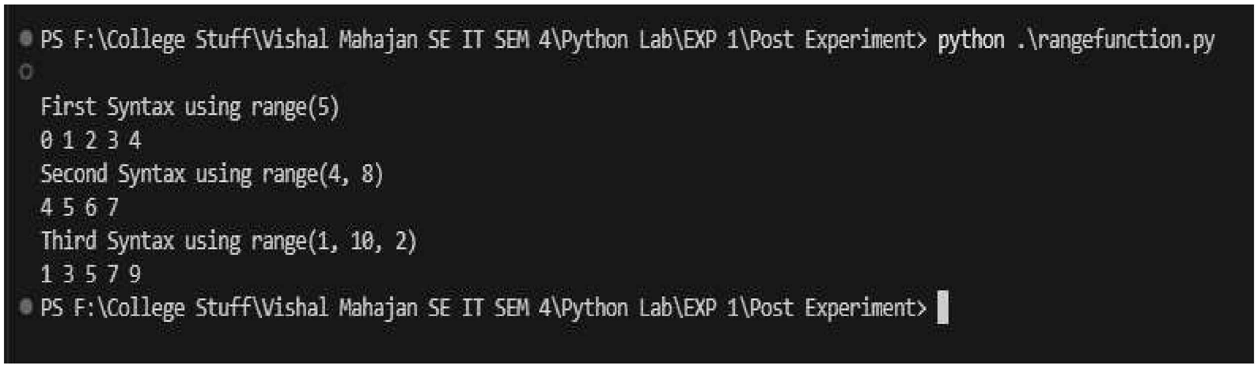
Post-Exp Exercise:

a. Write a Python program to implement three different syntaxes of range function.

Code:

```python
# First Syntax range(last) Return 0 to last-1
print("\nFirst Syntax using range(5)")
for i in range(5):
    print(i, end=' ')

# Second Syntax range(first, last) Return first to last-1
print("\nSecond Syntax using range(4, 8)")
for i in range(4, 8):
    print(i, end=' ')

# Third Syntax range(first, last, gap) Return first to last-1 with gap
print("\nThird Syntax using range(1, 10, 2)")
for i in range(1, 10, 2):
    print(i, end=' ')
```

Output:



```
PS F:\College Stuff\Vishal Mahajan SE IT SEM 4\Python Lab\EXP 1\Post Experiment> python .\rangefunction.py

First Syntax using range(5)
0 1 2 3 4
Second Syntax using range(4, 8)
4 5 6 7
Third Syntax using range(1, 10, 2)
1 3 5 7 9
PS F:\College Stuff\Vishal Mahajan SE IT SEM 4\Python Lab\EXP 1\Post Experiment>
```

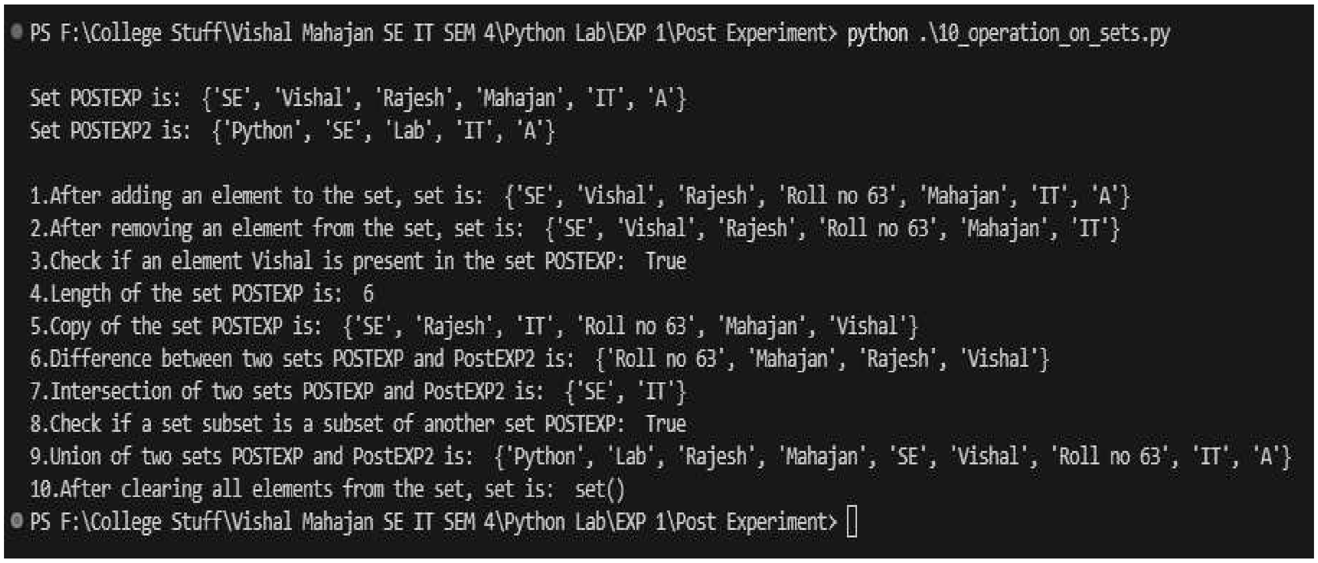
b.Write a Python program to implement any 10 methods of Sets

Code:

```
# Creating a set
PostEXP = {'Vishal', 'Rajesh', 'Mahajan', 'SE', 'IT', 'A'}
print("\nSet POSTEXP is: ", PostEXP)
PostEXP2 = {'Python', 'Lab','SE', 'IT', 'A'}
print("Set POSTEXP2 is: ", PostEXP2)

# 1. set.add(): Adds an element to the set
PostEXP.add('Roll no 63')
print("\n1.After adding an element to the set, set is: ",
PostEXP)

# 2. set.remove(): Removes an element from the set
PostEXP.remove('A')
print("2.After removing an element from the set, set is: ",
PostEXP)

# 3. Element in set: Check if an element is present in the set
print("3.Check if an element Vishal is present in the set
POSTEXP: ", 'Vishal' in PostEXP)

# 4. len(set): Get the length of the set
print("4.Length of the set POSTEXP is: ", len(PostEXP))

# 5. set.copy(): Copy the set
PostEXP_copy = PostEXP.copy()
print("5.Copy of the set POSTEXP is: ", PostEXP_copy)

# 6. set.difference(): Get the difference between two sets
difference = PostEXP.difference(PostEXP2)
print("6.Difference between two sets POSTEXP and PostEXP2 is: ",
difference)
```

```
# 7. set.intersection(): Get the intersection between two sets
intersection = PostEXP.intersection(PostEXP2)
print("7.Intersection of two sets POSTEXP and PostEXP2 is: ",
intersection)

# 8. set.issubset(): Check if a set subset is a subset of
another set
subset = {'Vishal', 'Rajesh'}
print("8.Check if a set subset is a subset of another set
POSTEXP: ", subset.issubset(PostEXP))

# 9. set.union(): Get the union between two sets
union = PostEXP.union(PostEXP2)
print("9.Union of two sets POSTEXP and PostEXP2 is: ", union)

# 10. set.clear(): Clear all elements from the set
PostEXP.clear()
print("10.After clearing all elements from the set, set is: ",
PostEXP)
```

Output:



```
PS F:\College Stuff\Vishal Mahajan SE IT SEM 4\Python Lab\EXP 1\Post Experiment> python .\10_operation_on_sets.py

 Set POSTEXP is:  {'SE', 'Vishal', 'Rajesh', 'Mahajan', 'IT', 'A'}
 Set POSTEXP2 is:  {'Python', 'SE', 'Lab', 'IT', 'A'}

 1.After adding an element to the set, set is:  {'SE', 'Vishal', 'Rajesh', 'Roll no 63', 'Mahajan', 'IT', 'A'}
 2.After removing an element from the set, set is:  {'SE', 'Vishal', 'Rajesh', 'Roll no 63', 'Mahajan', 'IT'}
 3.Check if an element Vishal is present in the set POSTEXP:  True
 4.Length of the set POSTEXP is:  6
 5.Copy of the set POSTEXP is:  {'SE', 'Rajesh', 'IT', 'Roll no 63', 'Mahajan', 'Vishal'}
 6.Difference between two sets POSTEXP and PostEXP2 is:  {'Roll no 63', 'Mahajan', 'Rajesh', 'Vishal'}
 7.Intersection of two sets POSTEXP and PostEXP2 is:  {'SE', 'IT'}
 8.Check if a set subset is a subset of another set POSTEXP:  True
 9.Union of two sets POSTEXP and PostEXP2 is:  {'Python', 'Lab', 'Rajesh', 'Mahajan', 'SE', 'Vishal', 'Roll no 63', 'IT', 'A'}
 10.After clearing all elements from the set, set is:  set()
PS F:\College Stuff\Vishal Mahajan SE IT SEM 4\Python Lab\EXP 1\Post Experiment>
```