Department of Computer Engineering

AY 2022 – 2023

Python Lab Manual

NAME: VIDHI KANSARA

SAP-ID: 60004200087

BRANCH: COMPUTER

BATCH: A4

Lab Experiment 1

AIM: To study and implement different data types and operators in python.

DESCRIPTION:

I] Data Types in Python:

1. **Text Type:**
   a. **String (str):** Strings in python are surrounded by either single quotation marks, or double quotation marks.
      'hello' is the same as "hello".
      You can display a string literal with the print() function.
      Example: a = "Hello World!"

2. **Numeric Type:**
   Variables of numeric types are created when you assign a value to them:
   a. **Integer (int):** Int, or integer, is a whole number, positive or negative, without decimals, of unlimited length.
      Example: a= 12
   b. **Float (float):** Float, or "floating point number" is a number, positive or negative, containing one or more decimals. Float can also be scientific numbers with an "e" to indicate the power of 10.
      Example: a = 12.5
   c. **Complex (complex):** Complex numbers are written with a "j" as the imaginary part.
      Example: a = 12 + 34j

3. **Sequence Type:**
   a. **List (list):** Lists are used to store multiple items in a single variable.
      Lists are one of 4 built-in data types in Python used to store collections of data, the other 3 are Tuple, Set, and Dictionary, all with different qualities and usage.Lists are created using square brackets.List items are ordered, changeable, and allow duplicate values.List items are indexed, the first item has index [0], the second item has index [1]etc.
      Example: a = ["apple", 12, 12.0, 44j]

b. **Range (range ()):** The range() function returns a sequence of numbers, starting from 0 by default, and increments by 1 (by default), and stops before a specified number.

   Example: a = range(12)

c. **Tuple (tuple):** Tuples are used to store multiple items in a single variable. Tuple is one of 4 built-in data types in Python used to store collections of data, theother 3 are List, Set, and Dictionary, all with different qualities and usage. A tuple is a collection which is ordered and unchangeable.Tuples are written with round brackets.

   Example: a= ("apple", 12, 4.12, 7j)

4. **Mapping Type:**
   a. **Dictionary (dict):** A dictionary is a collection which is ordered*, changeable and do not allowduplicates. Dictionary items are ordered, changeable, and does not allow duplicates. Dictionary items are presented in key:value pairs, and can be referred to by usingthe key name. Keys can be accessed using the keys() method and values can be accessed using thevalues() method.

      Example: a = {"fruit": "apple", “vegetable": “potato”}

5. **Set Type:**
   a. **Set (set):** Sets are used to store multiple items in a single variable. Set is one of 4 built-in data types in Python used to store collections of data, theother 3 are List, Tuple, and Dictionary, all with different qualities and usage. A set is a collection which is unordered, unchangeable*, and unindexed. Set items are unordered, unchangeable, and do not allow duplicate values.

      Example: a = {"apple", "banana", "cherry"}

6. **Boolean Type:**
   a. **Boolean (bool):** Booleans represent one of two values: True or False.

      Example: a = True, a = False

7. **Binary Type:**
   a. **Bytes (bytes):** The byte data type is used to manipulate binary data in python. Bytesis supported by buffer protocol, named memory view. The memory view can accessthe memory of other binary object without copying the actual data. The byte literals can be formed by these options.

      Example: x = b"Hi"

8. **None Type:**
   a. **None (None):** The None keyword is used to define a null value, or no value at

all.None is not the same as 0, False, or an empty string. None is a data type of its own(NoneType) and only None can be None.

Example: a = None

II] Operators in Python: Operators are used to perform operations on variables and values. In python, there are a total of 7 types of operators as follows:

1. **Arithmetic Operators:** Arithmetic operators are used with numeric values to perform common mathematicaloperations:
   a. **Addition (+):**
      Example: a + b
   b. **Subtraction (-):**
      Example: a - b
   c. **Multiplication (*):**
      Example: a * b
   d. **Division (/):**
      Example: a / b
   e. **Modulus (%):**
      Example: a % b
   f. **Exponentiation (**):**
      Example: a ** b
   g. **Floor**
      Division (//):

      Example: a//b

2. **Assignment Operators:**
   Assignment operators are used to assign values to variables:
   a. **Assign (=):**
      Assign value of right side of expression to left side operand.
      Example: a = 12
   b. **Add and Assign (+=):**Add right side operand with left side operand and then assign to left operand.
      Example: a += 12
   c. **Subtract and Assign (-=):**Subtract right operand from left operand and then assign to left operand.
      Example: a -= 12

d. **Multiply and Assign (*=):**Multiply right operand with left operand and then

assign to left operand.
Example: a*= 12

e. **Divide and Assign (/=):**Divide left operand with right operand and then assign to left operand.

Example: a /= 12

f. **Modulus and Assign (%=):**
Takes modulus using left and right operands and assign result to left operand.
Example: a %= 12

g. **Exponentiate and Assign (**=):**
Calculate exponent (raise power) value using operands and assign value to leftoperand.
Example: a **= 12

h. **Divide (Floor) and Assign (//=):**
Divide left operand with right operand and then assign the value(floor) to leftoperand.
Example: a//= 12

3. **Comparison Operators:**
Comparison operators are used to compare two values:
   a. **Equal (==):**
   Example: a == b

   b. **Not equal (!=):**
   Example: a!= b

   c. **Greater than (>):**
   Example: a>b

   d. **Less than (<):**
   Example: a<b

   e. **Greater than or equal to (>=):**
   Example: a >= b

   f. **Less than or equal to (<=):**
   Example: a <= b

4. **Logical Operators:**
Logical operators are used to combine conditional statements:
   a. **'and' operator:**
   Returns True if both statements are true.

Example: a <1 and a < 2

b. **'or' operator:**
   Returns True if one of the statements is true.
   Example: a < 12 or a< 2

c. **'not' operator:**
   Reverse the result, returns False if the result is true and vice-versa.
   Example: not(a < 15 and a < 10)

5. **Identity Operators:**
   Identity operators are used to compare the objects, not if they are equal, but if they areactually the same object, with the same memory location:
   a. **'is' operator:**
      Returns True if both variables are the same object.
      Example: a is b

   b. **'is not' operator:**
      Returns True if both variables are not the same object.
      Example: a is not b

6. **Membership Operators:**
   Membership operators are used to test if a sequence is present in an object:
   a. **'in' operator:**
      Returns True if a sequence with the specified value is present in the object.
      Example: a in b

   b. **'not in' operator:**
      Returns True if a sequence with the specified value is not present in the object.
      Example: a not in b

7. **Bitwise Operators:**
   Bitwise operators are used to compare (binary) numbers:
   a. **AND (&):**
      Sets each bit to 1 if both bits are 1.
      Example: a & b

   b. **OR (|):**
      Sets each bit to 1 if one of two bits is 1.
      Example: a | b

   c. **XOR (^):**
      Sets each bit to 1 if only one of two bits is 1.

Example: a ^ b

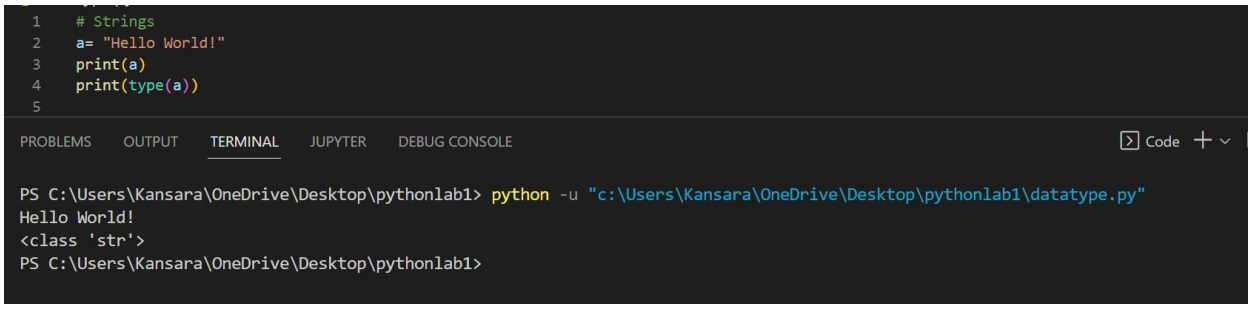
d. **NOT (~):**
Inverts all the bits.
Example: ~a

e. **Zero fill left shift (<<):**
Shift left by pushing zeros in from the right and let the leftmost bits fall off.
Example: a << 2

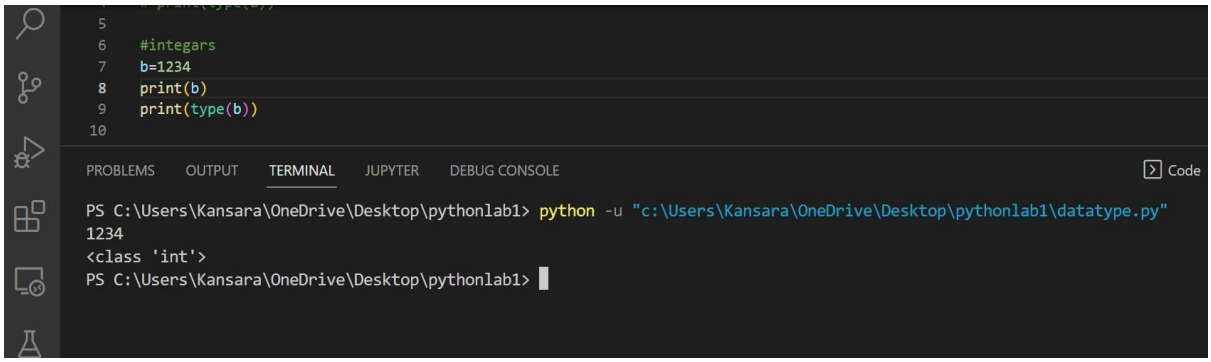
f. **Signed right shift (>>):**
Shift right by pushing copies of the leftmost bit in from the left, and let the rightmostbits fall off.

IMPLEMENTATION:

1) Data Types in Python:
   a. Strings:



   b. Integars:



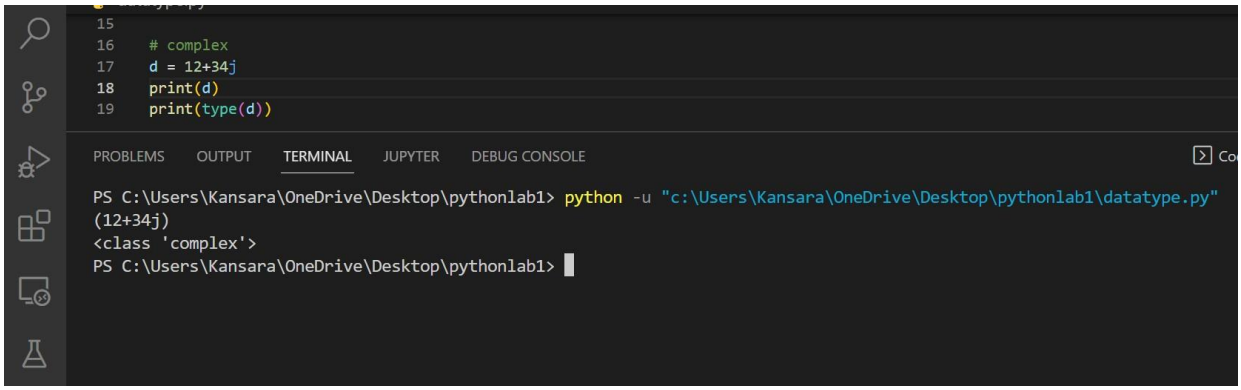
   c. Float:

```
11  # float
12  c=12.34
13  print(c)
14  print(type(c))
15
```

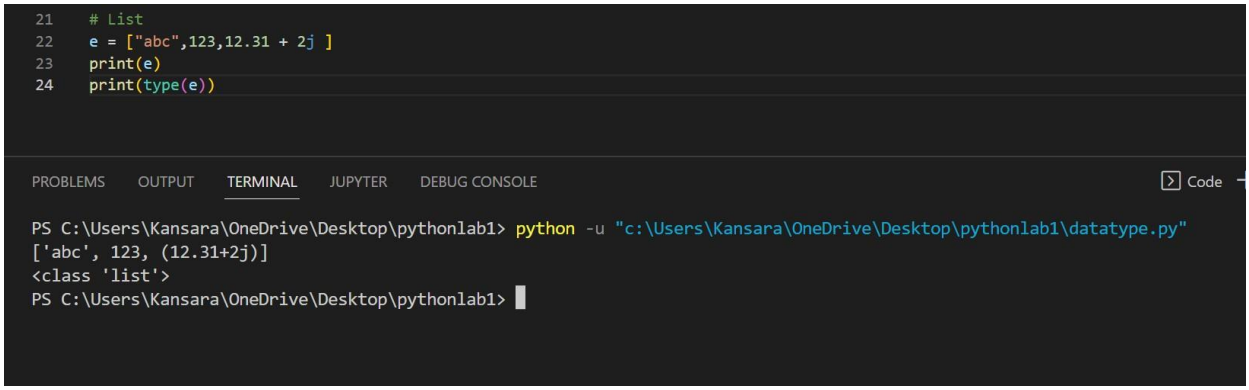
PROBLEMS OUTPUT TERMINAL JUPYTER DEBUG CONSOLE Code

```
PS C:\Users\Kansara\OneDrive\Desktop\pythonlab1> python -u "c:\Users\Kansara\OneDrive\Desktop\pythonlab1\datatype.py"
12.34
<class 'float'>
PS C:\Users\Kansara\OneDrive\Desktop\pythonlab1>
```

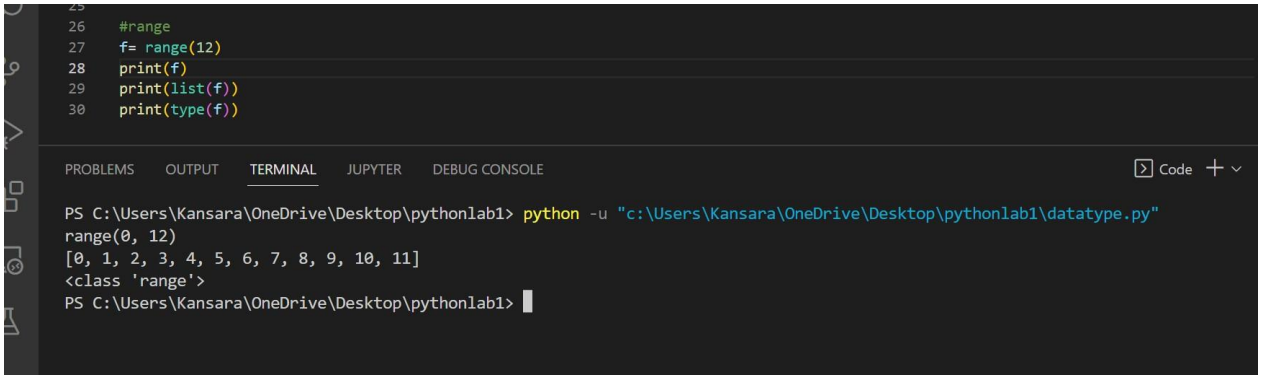
d. Complex:



e. List:



f. Range:



g. Tuple:

```
32  # tupple
33  g= ("abc",123,12.31 + 2j)
34  print(g)
35  print(type(g))
```

```
PROBLEMS  OUTPUT  TERMINAL  JUPYTER  DEBUG CONSOLE

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\Kansara\OneDrive\Desktop\pythonlab1> python -u "c:\Users\Kansara\OneDrive\
('abc', 123, (12.31+2j))
<class 'tuple'>
PS C:\Users\Kansara\OneDrive\Desktop\pythonlab1>
```

h. Dictionary:

```
42  # Dictionary
43  i={'frui (method) keys: () -> dict_keys[str, str]
44  print(i)
45  print(ty D.keys() -> a set-like object providing a view on D's keys
46  print(i.keys())
47  print(i.values())
48
```

```
PROBLEMS  OUTPUT  TERMINAL  JUPYTER  DEBUG CONSOLE                    Code +

PS C:\Users\Kansara\OneDrive\Desktop\pythonlab1> python -u "c:\Users\Kansara\OneDrive\Desktop\pythonlab1\datatype.py"
{'fruit': 'mango', 'number': '123'}
<class 'dict'>
dict_keys(['fruit', 'number'])
dict_values(['mango', '123'])
PS C:\Users\Kansara\OneDrive\Desktop\pythonlab1>
```

i. Set:

```
37  # Set
38  h= {"abc",123,12.31 + 2j}
39  print(h)
40  print(type(h))
41
```

```
PROBLEMS  OUTPUT  TERMINAL  JUPYTER  DEBUG CONSOLE                    Code

PS C:\Users\Kansara\OneDrive\Desktop\pythonlab1> python -u "c:\Users\Kansara\OneDrive\Desktop\pythonlab1\datatype.py"
{'abc', (12.31+2j), 123}
<class 'set'>
PS C:\Users\Kansara\OneDrive\Desktop\pythonlab1>
```

j. Boolean

```
48
49  # Boolean
50  j= True
51  print(j)
52  print(type(j))
```

```
PROBLEMS  OUTPUT  TERMINAL  JUPYTER  DEBUG CONSOLE
PS C:\Users\Kansara\OneDrive\Desktop\pythonlab1> python -u "c:\Users\Kansara\OneDrive\Desktop\pythonlab1\datatype.py"
True
<class 'bool'>
PS C:\Users\Kansara\OneDrive\Desktop\pythonlab1>
```

k. Binary(bytes):

```
54  # Binary-bytes
55  k=b'hi'
56  print(k)
57  print(type(k))
```

```
PROBLEMS  OUTPUT  TERMINAL  JUPYTER  DEBUG CONSOLE
PS C:\Users\Kansara\OneDrive\Desktop\pythonlab1> python -u "c:\Users\Kansara\OneDrive\Desktop\pythonlab1\datatype.py"
b'hi'
<class 'bytes'>
PS C:\Users\Kansara\OneDrive\Desktop\pythonlab1>
```

l. None:

```
58
59  # None
60  l=None
61  print(l)
62  print(type(l))
```

```
PROBLEMS  OUTPUT  TERMINAL  JUPYTER  DEBUG CONSOLE
PS C:\Users\Kansara\OneDrive\Desktop\pythonlab1> python -u "c:\Users\Kansara\OneDrive\Desktop\pythonlab1\datatype.py"
None
<class 'NoneType'>
PS C:\Users\Kansara\OneDrive\Desktop\pythonlab1>
```

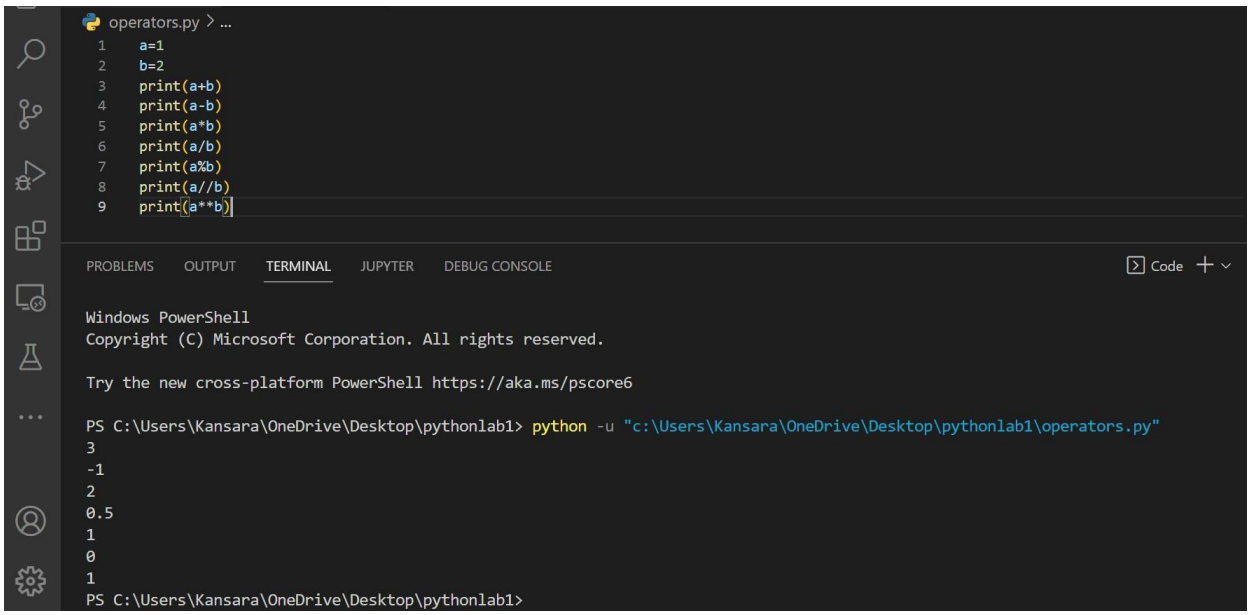
m. Id() method:

```
63
64  # Id()
65  m=12
66  print(id(m))
67  print(m)
```

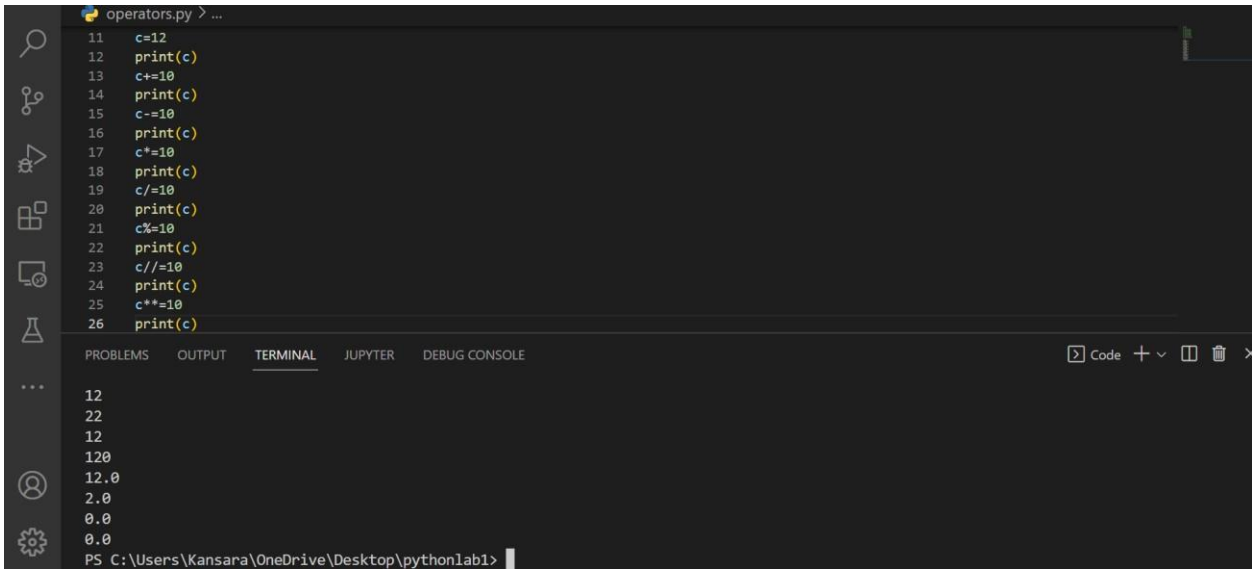
```
PROBLEMS  OUTPUT  TERMINAL  JUPYTER  DEBUG CONSOLE
PS C:\Users\Kansara\OneDrive\Desktop\pythonlab1> python -u "c:\Users\Kansara\OneDrive\Desktop\pythonlab1\datatype.py"
2851043895952
12
PS C:\Users\Kansara\OneDrive\Desktop\pythonlab1>
```

2) Operators in python:

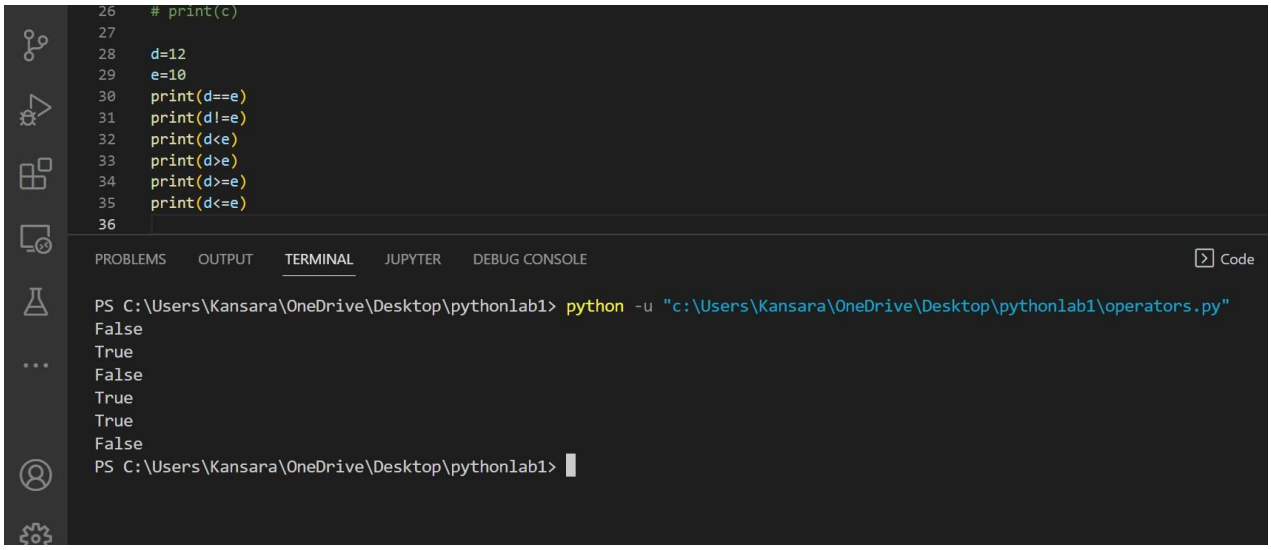
a. Arithmetic operators:



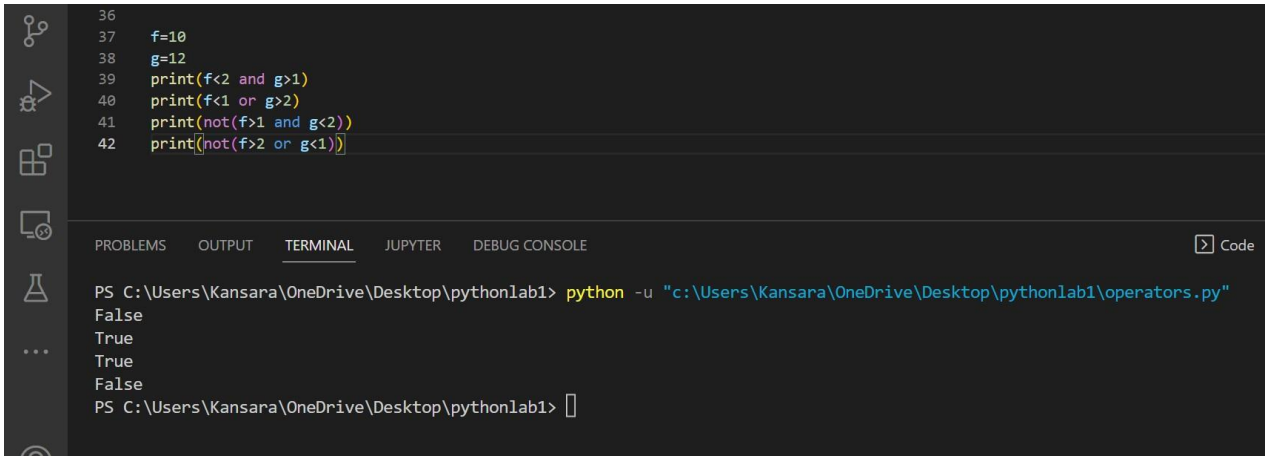
b. Assignment Operators:



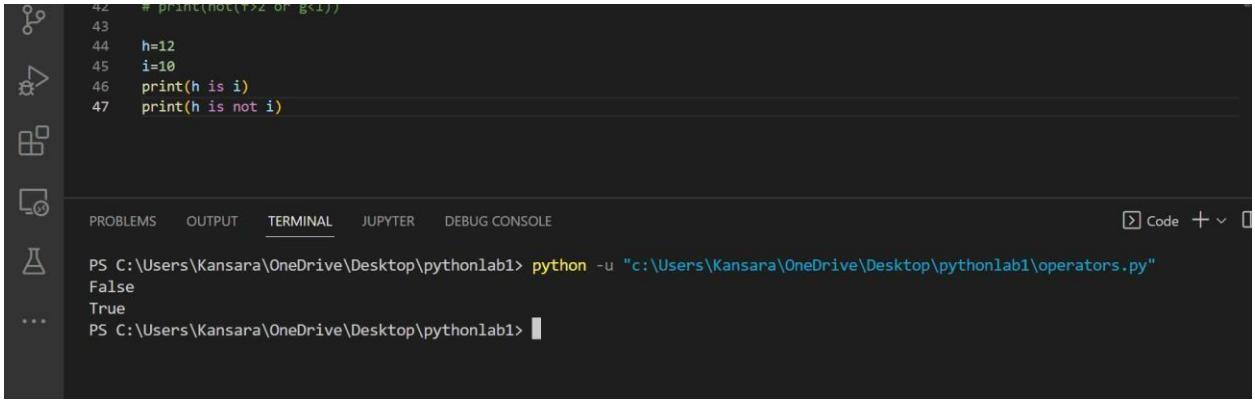
c. Comparison Operators



d. Logical Operators:

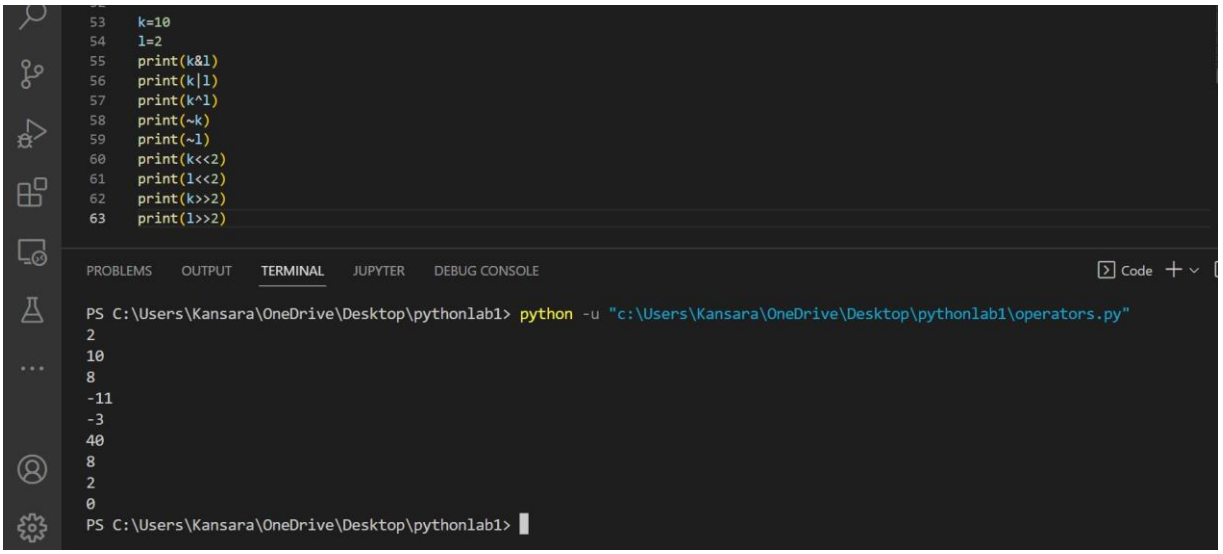


e. Identity Operators:



f. Membership Operators:

g. Bitwise Operators:



CONCLUSION: From this lab we understand about 8 different types of built-in data types forhandling different types of received data. We also understood about how data is being used inpython by understanding about 7 types of operators. Thus, we understood about data types and their operators in python and implemented them using various examples.

Lab Experiment – 02

AIM: To study and implement input – output statements and control and loop statements.

THEORY:

1. Input-Output Statements: There are different ways and syntax statements used to take input from user and display the required output.
   a. print() function: The print() function prints the specified message to the screen, or other standard output device. The message can be a string, or any other object, the object will be converted into a string before written to the screen.
   b. input() function: The input() function allows user input.

      Example:
      ```
      x = input('Enter your name:')
      print('Hello, ' + x)
      ```
   c. append() function: Python's append() function inserts a single element into an existing list. The element will be added to the end of the old list rather than being returned to a new list. Adds its argument as a single element to the end of a list. The length of the list increases by one.

   Example:
   ```
   my_list = ['geeks', 'for']
   my_list.append('geeks')
   print my_list
   ```

   d. add() function: The add() method adds an element to the set. If the element already exists, the add() method does not add the element.
   e. map() function: The **map()** function returns a map object(which is an iterator) of the results after applying the given function to each item of a given iterable(list, tuple etc.)
   f. split() function: The split() method splits a string into a list.You can specify the separator, default separator is any whitespace.

2. Control Statements/Loop Statements: Python programming language provides the following types of loops to handle looping requirements and control statements.
   a. While loop: In python, a while loop is used to execute a block of statements repeatedly until a given condition is satisfied. And when the condition becomes false, the line immediately after the loop in the program is executed.

   Syntax:
   ```
   while expression:
       statement(s)
   ```

   b. For loop: A for loop is used for iterating over a sequence (that is either a list, a tuple, a dictionary, a set, or a string). It is less like the for keyword in other programming languages, and works more like an iterator method as found in other object-orientated programming languages. With the for loop we can execute a set of statements, once for each item in a list, tuple, set etc.

      Syntax:
      ```
      for iterator_var in sequence:
      ```

```
statements(s)
```

c. If..else..: The if statement alone tells us that if a condition is true it will execute a block of statements and if the condition is false it won't. But what if we want to do something else if the condition is false. Here comes the *else* statement. We can use the *else* statement with *if* statement to execute a block of code when the condition is false.

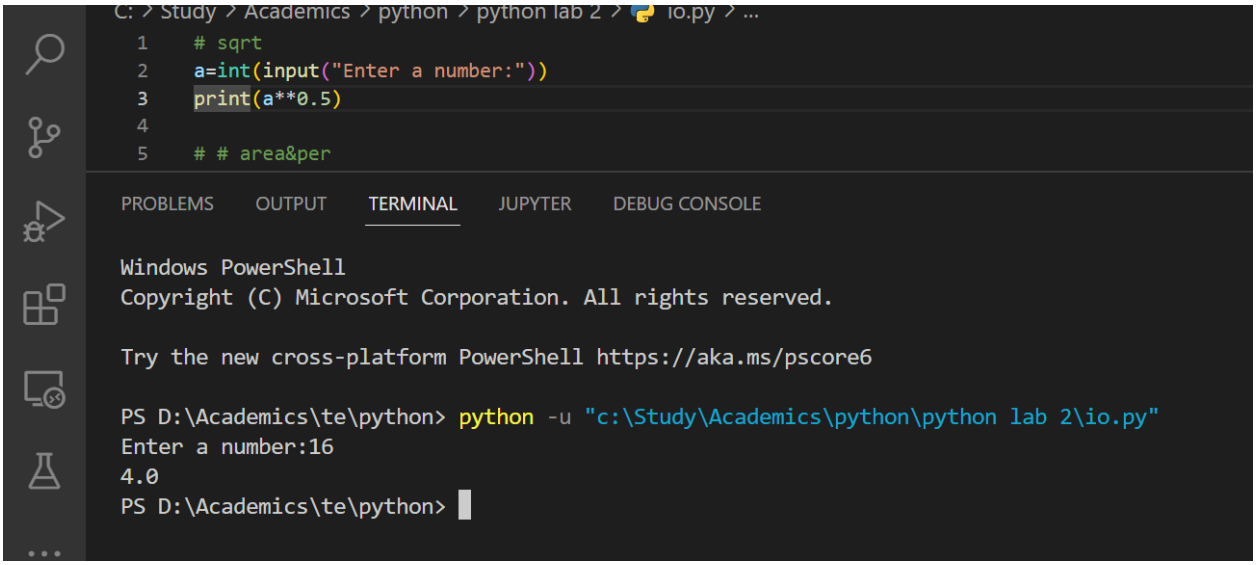
Syntax:
```
if (condition):
        # Executes this block if
        # condition is true
else:
    # Executes this block if
    # condition is false
```

d. If..elif..else statement: If-elif-else statement is used in Python for decision-making i.e the program will evaluate test expression and will execute the remaining statements only if the given test expression turns out to be true. This allows validation for multiple expressions.

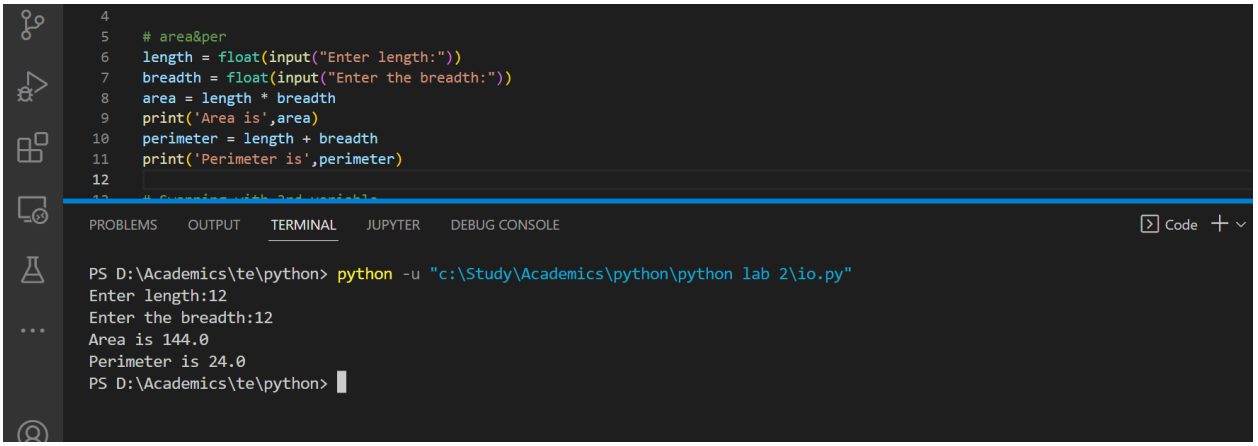
Syntax:
```
if test expression:
    Body of if
elif test expression:
    Body of elif
else:
    Body of else
```

IMPLEMENTATION:

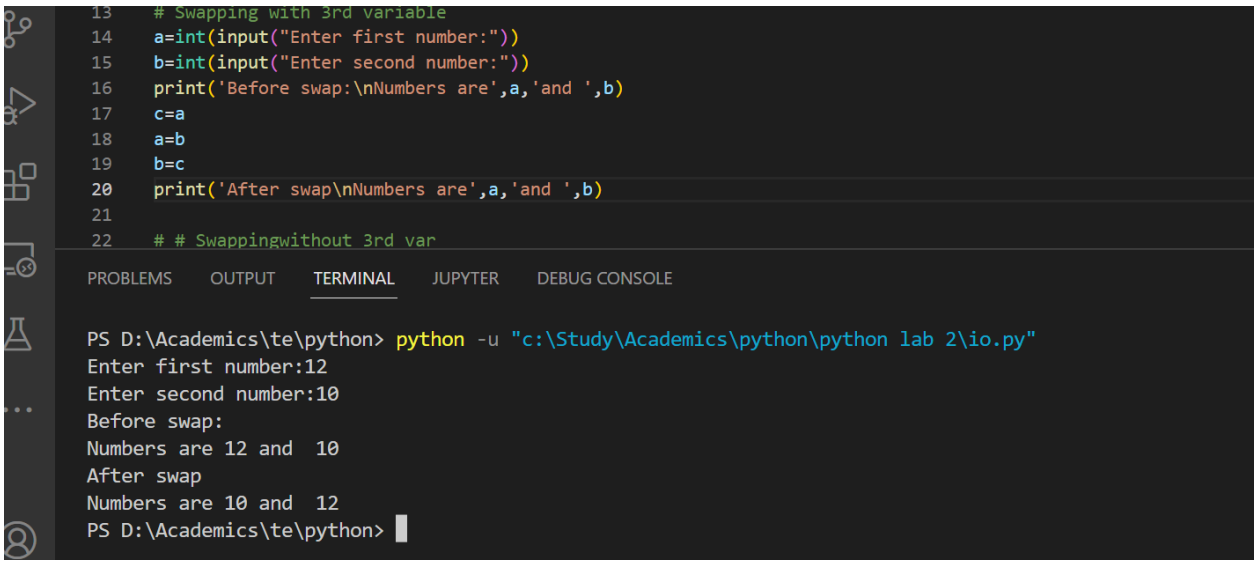
1. Find square root of a number



2. Find area and perimeter of rectangle.



3. Swapping of two numbers:
   a. With third variable



   b. Without third variable

```python
# Swappingwithout 3rd var
a=int(input("Enter first number:"))
b=int(input("Enter second number:"))
print('Before swap:\nNumbers are',a,'and ',b)
a=a+b
b=a-b
a=a-b
print('After swap\nNumbers are',a,'and ',b)

# # Adding elements in list
```

PROBLEMS OUTPUT TERMINAL JUPYTER DEBUG CONSOLE

```
PS D:\Academics\te\python> python -u "c:\Study\Academics\python\python lab 2\io.py"
Enter first number:12
Enter second number:10
Before swap:
Numbers are 12 and  10
After swap
```

4. Adding elements in list, Set and tuple.

```python
# Adding elements in list
List=list()
l=int((input("Size of list:")))
print("Enter list elements:")
for i in range(0,l):
    List.append(int(input()))
print(list)
# Adding elements in set
Set=set()
s=int(input("Size of set:"))
print("Enter set element:")
for i in range(0,s):
    Set.add(int(input()))
print(set)
# Adding elements in tuple
T=(2,3,4,5,6)
print('Tupple before adding elements:')
print(T)
L=list(T)
L.append(int(input("Enter the neww element:")))
T=tuple(L)
print("Tupple after adding new element:")
print(T)
```

PROBLEMS OUTPUT TERMINAL JUPYTER DEBUG CONSOLE

```
PS D:\Academics\te\python> python -u "c:\Study\Academics\python\python lab 2\io.py"
Size of list:2
Enter list elements:
1
2
<class 'list'>
Size of set:2
Enter set element:
1
2
<class 'set'>
Tupple before adding elements:
(2, 3, 4, 5, 6)
Enter the neww element:4
Tupple after adding new element:
(2, 3, 4, 5, 6, 4)
PS D:\Academics\te\python>
```

5. Print the give triangle pattern using for loop.

```python
# pattern
n = int(input("Enter number of rows: "))
for i in range(1,n+1):
    for j in range(1, i+1):
        print(i, end=" ")
    print()
```

PROBLEMS OUTPUT TERMINAL JUPYTER DEBUG CONSOLE

```
PS C:\Study\Academics\python\python lab 2> python -u "c:\Study\Academics\python\python lab 2\controlstatments.py"
Enter number of rows: 4
1
2 2
3 3 3
4 4 4 4
PS C:\Study\Academics\python\python lab 2>
```

6. Print factorial of a number using for loop.

```python
# factorial
n=int(input("Enter the number for factorial:"))
fact=1
if n<0:
    print("Factorial does not exist for negative numbers")
elif n==0:
    print("The factorial of 0 is 1")
else:
    for i in range(1, n+1):
        fact = fact * i
print("Factorial is:",fact)
```

```
PS C:\Study\Academics\python\python lab 2> python -u "c:\Study\Academics\python\python lab 2\tempCodeRunnerFile.py"
Enter the number for factorial:4
Factorial is: 24
PS C:\Study\Academics\python\python lab 2>
```

7. Print the Fibonacci sequence up to given 'n' value using for loop

```python
# fibonacci
n=int(input("Enter the terms:"))
f=0
s=1
if n<=0:
    print('Enter valid input.')
else:
    print(f,s,end=" ")
    for x in range(2,n):
        next=f+s
        print(next,end=" ")
```

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Study\Academics\python\python lab 2> python -u "c:\Study\Academics\python\python lab 2\controlstatments.py"
Enter the terms:4
0 1 1 2
```

8. Demonstrate 'while' loop with one example

```python
10
num = int(input("Enter a number: "))
fac = 1
i = 1
while i <= num:
    fac = fac * i
    i = i + 1
print("factorial of ", num, " is ", fac)
```

```
PS C:\Study\Academics\python\python lab 2> python -u "c:\Study\Academics\python\python lab 2\controlstatments.py"
Enter a number: 4
factorial of  4  is  24
PS C:\Study\Academics\python\python lab 2>
```

9. Check whether the input number is even or odd using if...else loop

```
43
44  # evenodd
45  n=int(input('Enter the number to be checked:'))
46  if (n%2)==0:
47      print(n,'is even.')
48  else:
49      print(n,'is odd.')
50
```

PROBLEMS OUTPUT TERMINAL JUPYTER DEBUG CONSOLE

```
PS C:\Study\Academics\python\python lab 2> python -u "c:\Study\Academics\python\python lab 2\controlstatments.py"
Enter the number to be checked:5
5 is odd.
PS C:\Study\Academics\python\python lab 2>
```

10. Check whether the input year is leap year or not using nested if

```
67
68  # nested if else
69  a=int(input("Enter the year:"))
70  if a%4 ==0:
71      if a%100 ==0:
72          if a%400 ==0:
73              print(a,"is a leap year.")
74          else:
75              print(a,"is not a leap year.")
76      else:
77          print(a,"is a leap year.")
78  else:
79      print(a,"is not a leap year.")
80
```

PROBLEMS OUTPUT TERMINAL JUPYTER DEBUG CONSOLE

```
PS C:\Study\Academics\python\python lab 2> python -u "c:\Study\Academics\python\python lab 2\controlstatments.py"
Enter the year:2020
2020 is a leap year.
PS C:\Study\Academics\python\python lab 2>
```

11. Demonstrate 'if...elif...else' loop with one example

```
50
51  # elif..if..else
52  num = 2+3j
53  if type(num) == int:
54      print("Number is integer.")
55  elif type(num) == list:
56      print("number is a list.")
57  elif type(num) == tuple:
58      print("Number is tuple.")
59  elif type(num) == set:
60      print('Number is a set')
61  elif type(num) == float:
62      print('Number is a float integar data type.')
63  elif type(num) == str:
64      print("It is a string.")
65  else:
66      print("number is complex.")
67
```

PROBLEMS OUTPUT TERMINAL JUPYTER DEBUG CONSOLE

```
PS C:\Study\Academics\python\python lab 2> python -u "c:\Study\Academics\python\python lab 2\controlstatments.py"
number is complex.
PS C:\Study\Academics\python\python lab 2>
```

12. Demonstrate 'continue', 'break' and 'pass' with one example each

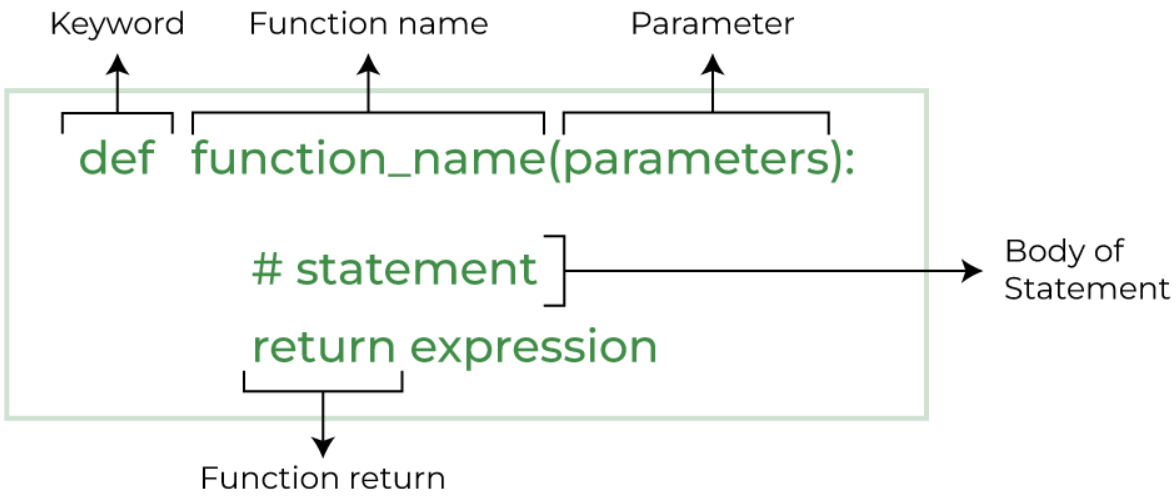
CONCLUSION: Thus from above set of experiments we understood about how input/output statements work and how looping and control statements function and thus implemented using few experiments.

Lab Experiment – 03

AIM: To implement functions in Python.

THEORY: A function is a block of code which only runs when it is called. One can pass data, known as parameters, into a function. A function can return data as a result. The main idea to use function is to put some commonly or repeatedly done tasks together and make a function so that instead of writing the same code again and again for different inputs, we can do the function calls to reuse code contained in it over and over again.

SYNATAX AND STRUCTURE OF A FUNCTION:



There are various types of inbuilt functions mentioned for data structures like lists, sets, tuples, dictionary. Some of them are given below with some descriptions.

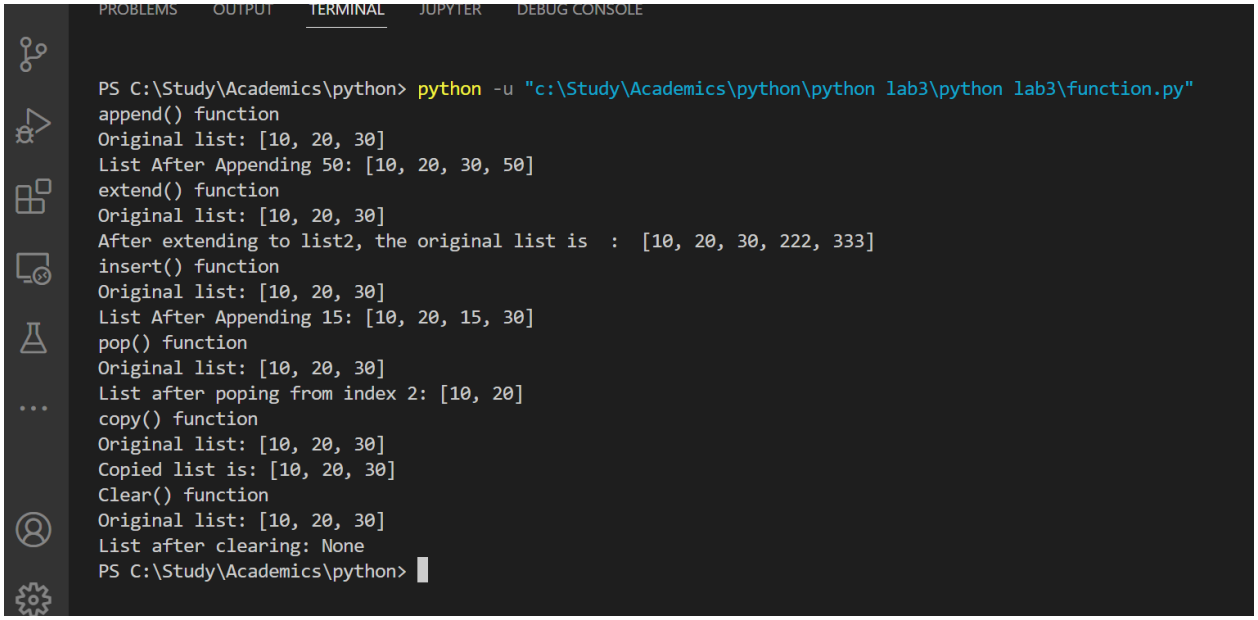
IMPLEMENTATION:

1 Functions of data structures:

a. LISTS

```
# append()-appends element to list

print("append() function")

list1 = [10, 20, 30]

print("Original list:", list1)

list1.append(50)

print("List After Appending 50:", list1)


# extend()- extends by adding on with new list

print("extend() function")
```

```
list1 = [10, 20, 30]
print("Original list:", list1)
list2 = [222, 333]
list1.extend(list2)
print("After extending to list2, the original list is  : ", list1)

# insert() – inserts element at particular location
print("insert() function")
list1 = [10, 20, 30]
print("Original list:", list1)
list1.insert(2, 15)
print("List After Appending 15:", list1)

# pop()- pops element from list
print("pop() function")
list1 = [10, 20, 30]
print("Original list:", list1)
list1.pop(2)
print("List after poping from index 2:", list1)

# copy()-copies a list
print("copy() function")
list1 = [10, 20, 30]
print("Original list:", list1)
list2 = list1.copy()
print("Copied list is:", list2)

# clear()-clears the list
print("Clear() function")
list1 = [10, 20, 30]
print("Original list:", list1)
list2 = list1.clear()
print("List after clearing:", list2)
```


PROBLEMS
OUTPUT
TERMINAL
JUPYTER
DEBUG CONSOLE
PS C:\Study\Academics\python> python -u "c:\Study\Academics\python\python lab3\python lab3\function.py"
append() function
Original list: [10, 20, 30]
List After Appending 50: [10, 20, 30, 50]
extend() function
Original list: [10, 20, 30]
After extending to list2, the original list is : [10, 20, 30, 222, 333]
insert() function
Original list: [10, 20, 30]
List After Appending 15: [10, 20, 15, 30]
pop() function
Original list: [10, 20, 30]
List after poping from index 2: [10, 20]
copy() function
Original list: [10, 20, 30]
Copied list is: [10, 20, 30]
Clear() function
Original list: [10, 20, 30]
List after clearing: None
PS C:\Study\Academics\python>

b. TUPLES

```
tuple1 = (10, 20, 30, 50, 50, 70, 90, 80, 100)
print("Original tuple:", tuple1)

# len()- length of tuple
print("len() function")
print("The length of tuple is:",len(tuple1))

# count()-repetation of element in tuple
print("count() function")
print("The count of 50 in tuple is:",tuple1.count(50))

# index()- gives index of element in tuple
print("Index() function")
print("The index of 80 in tuple is:", tuple1.index(80))

# sort()-sorts the elements in tuple
print("sort() function")
print("The sorted tuple is:", sorted(tuple1))

# min()- gives minimum element of tuple
print("min() function")
print("The minimum element of tuple is:", min(tuple1))

# max()- gives maximum element of tuple
print("max() function")
print("The maximum element of tuple is:", max(tuple1))

# sum()-gives sum of elements in tuple
print("sum() function")
print("The sum of elements in tuple is:", sum(tuple1))
```

```
PS C:\Study\Academics\python> python -u "c:\Study\Academics\python\python lab3\python lab3\function.py"
Original tuple: (10, 20, 30, 50, 50, 70, 90, 80, 100)
len() function
The length of tuple is: 9
count() function
The count of 50 in tuple is: 2
Index() function
The index of 80 in tuple is: 7
sort() function
The sorted tuple is: [10, 20, 30, 50, 50, 70, 80, 90, 100]
min() function
The minimum element of tuple is: 10
max() function
The maximum element of tuple is: 100
sum() function
The sum of elements in tuple is: 500
PS C:\Study\Academics\python>
```

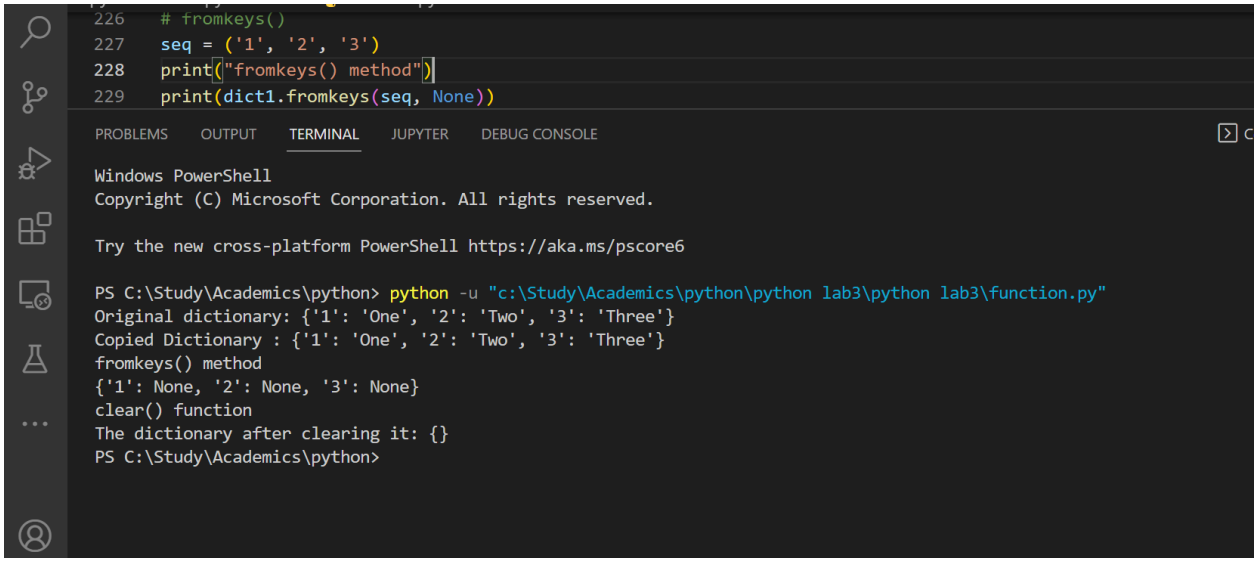
c. SETS

```
set1 = {'a','b', 'c', 'd', 'e'}

print("Original set is:", set1)


# add()- adds element to set

set1.add('f')

print("add() function")

print("Set after adding 'f'", set1)


# discard() – discards element from set

set1.discard('e')

print("discard() function")

print("Set after discarding/Updating 'e': ",set1)


# remove() – removes element from set

set1.remove('a')

print("remove() function")

print("Set after removing 'a':", set1)


# pop()- pops element from the set

set1.pop()

print("pop() function")

print("Set after poping elements:", set1)
```

```
# clear()-clears the set
set1.clear()
print("clear() function")
print("Set after clearing:",set1)
```

d. DICTIONARIES

```
dict1 = {'1': 'One','2':'Two','3':'Three'}
print("Original dictionary:", dict1)

# copy()- copies the dictionary
dict2 = dict1.copy()
print("Copied Dictionary :",dict2)

# fromkeys() – gives details from dictionary
seq = ('1', '2', '3')
print("fromkeys() method")
print(dict1.fromkeys(seq, None))

# clear()- clears the dictionary
dict1.clear()
print("clear() function")
print("The dictionary after clearing it:", dict1)
```


226 # fromkeys()
227 seq = ('1', '2', '3')
228 print("fromkeys() method")
229 print(dict1.fromkeys(seq, None))
PROBLEMS OUTPUT TERMINAL JUPYTER DEBUG CONSOLE
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.
Try the new cross-platform PowerShell https://aka.ms/pscore6
PS C:\Study\Academics\python> python -u "c:\Study\Academics\python\python lab3\python lab3\function.py"
Original dictionary: {'1': 'One', '2': 'Two', '3': 'Three'}
Copied Dictionary : {'1': 'One', '2': 'Two', '3': 'Three'}
fromkeys() method
{'1': None, '2': None, '3': None}
clear() function
The dictionary after clearing it: {}
PS C:\Study\Academics\python>

1. Function histogram(l) that takes as input a list of integers with repetitions and returns a list of pairs:

```
def histogram(l):
    count = 0
    x=[]
    k=[]
    for i in range(len(l)):
        index=i
        count=0
        for j in range(index,len(l)):
            if l[index] == l[j] and l[index] not in k:
                count += 1
            k=k+ [l[index]]
            if(count!=0):
                x=x+[(l[index],count)]
    x.sort()
    x=sorted(x,key=lambda x:x[1])
    return x
print(histogram([13,12,11,13,14,13,7,7,13,14,12]))
print(histogram([7,12,11,13,7,11,13,14,12]))
print(histogram([13,7,12,7,11,13,14,13,7,11,13,14,12,14,14,7]))
```

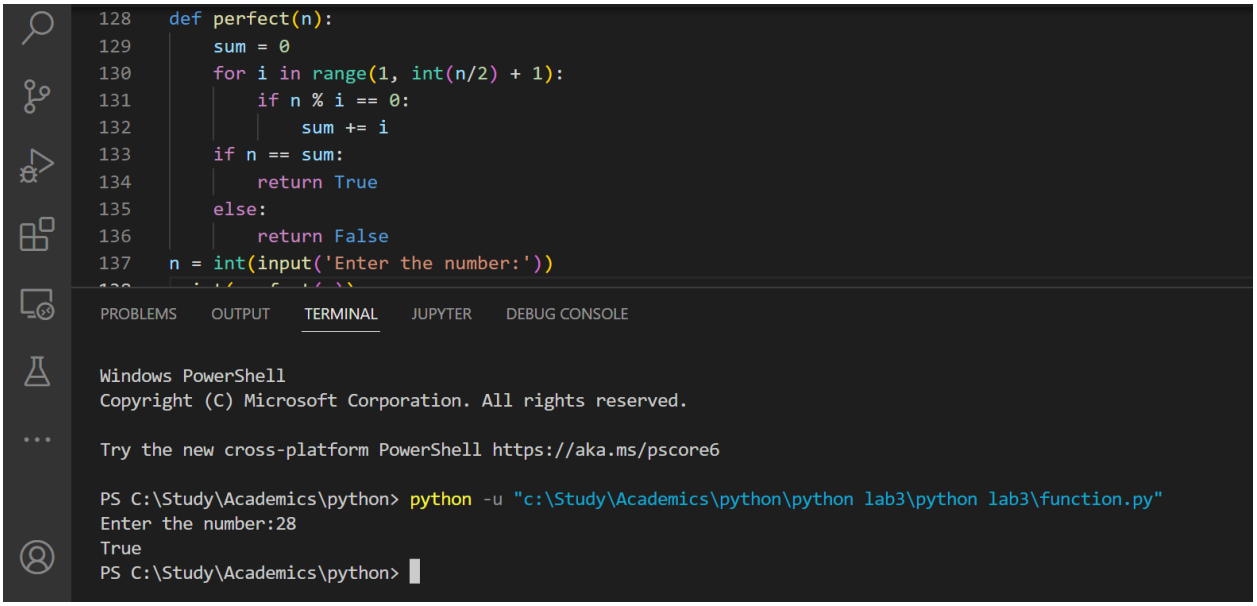
```
104         for j in range(index,len(l)):
105             if l[index] == l[j] and l[index] not in k:
106                 count = count + 1
107         k=k+ [l[index]]
108         if(count!=0):
109             x=x+[(l[index],count)]
110     x.sort()
111     x=sorted(x,key=lambda x:x[1])
112     return x
113 print(histogram([13,12,11,13,14,13,7,7,13,14,12]))
```

PROBLEMS OUTPUT TERMINAL JUPYTER DEBUG CONSOLE

```
PS C:\Study\Academics\python> python -u "c:\Study\Academics\python\python lab3\python lab3\function.py"
[(11, 1), (7, 2), (12, 2), (14, 2), (13, 4)]
[(14, 1), (7, 2), (11, 2), (12, 2), (13, 2)]
[(11, 2), (12, 2), (7, 4), (13, 4), (14, 4)]
PS C:\Study\Academics\python>
```

2. A python function that PERFECT(N) that takes positive integer argument and returns True if integer is perfect, or FALSE otherwise.

```
def perfect(n):
    sum = 0
    for i in range(1, int(n/2) + 1):
        if n % i == 0:
            sum += i
    if n == sum:
        return True
    else:
        return False
n = int(input('Enter the number:'))
print(perfect(n))
```



3. Implement a recursive function to solve tower of Hanoi Problem.

```
def hanoi(disks, source , auxiliary, target):
    if disks ==1:
        print('Move disk 1 from peg{} to peg{}.'.format(source,target))
        return
    hanoi(disks - 1,source,target,auxiliary)
    print('Moves disk{} from peg{} to peg{}.'.format(disks,
   source,target))
    hanoi(disks - 1 , auxiliary , source ,target)
disks = int(input('Enter number of disks:'))
hanoi(disks, 'A','B','C')
a=int(input('Enter a number:'))
b=int(input('Enter a number:'))
maximum = lambda a,b:a if a> b else b
print(f'{maximum(a,b)} is a maximum number')
```

```
56
57  def hanoi(disks, source , auxiliary, target):
58      if disks ==1:
59          print('Move disk 1 from peg{} to peg{}.'.format(source,target))
60          return
61      hanoi(disks - 1,source,target,auxiliary)
62      print('Moves disk{} from peg{} to peg{}.'.format(disks, source,target))
63      hanoi(disks - 1 , auxiliary , source ,target)
64
65  disks = int(input('Enter number of disks:'))
66  hanoi(disks, 'A','B','C')
67
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  JUPYTER

PS C:\Users\djsce.student\Desktop\python lab3> python function.py
Enter number of disks:4
Move disk 1 from pegA to pegB.
Moves disk2 from pegA to pegC.
Move disk 1 from pegB to pegC.
Moves disk3 from pegA to pegB.
Move disk 1 from pegC to pegA.
Moves disk2 from pegC to pegB.
Move disk 1 from pegA to pegB.
Moves disk4 from pegA to pegC.
Move disk 1 from pegB to pegC.
Moves disk2 from pegB to pegA.
Move disk 1 from pegC to pegA.
Moves disk3 from pegB to pegC.
Move disk 1 from pegA to pegB.
Moves disk2 from pegA to pegC.
Move disk 1 from pegB to pegC.
PS C:\Users\djsce.student\Desktop\python lab3>
```

4. Implement lambda function to find greater of two numbers.

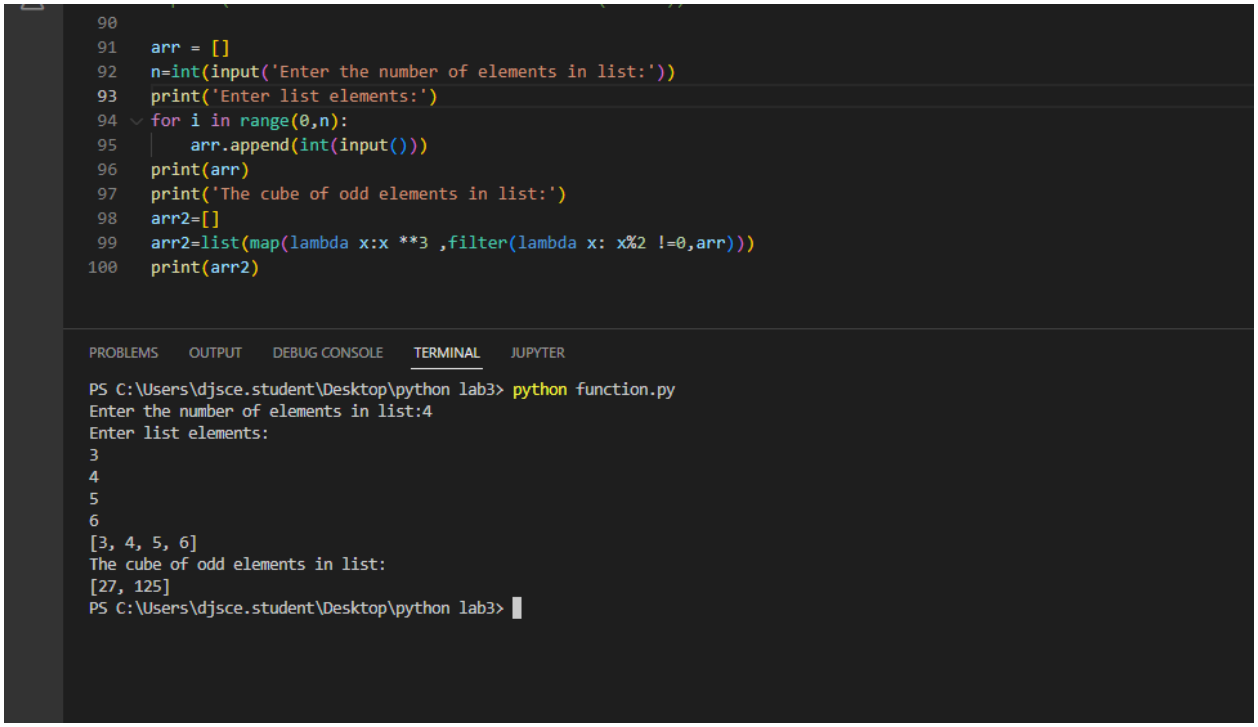
```
a=int(input('Enter a number:'))
b=int(input('Enter a number:'))
maximum = lambda a,b:a if a> b else b
print(f'{maximum(a,b)} is a maximum number')
```

5. Using map function perform element wise addition of elements of two lists.

```
list1=[]
list2=[]
def add(a,b):
    return a+b
n1=int(input('Enter number of elements in list 1:'))
for i in range(0,n1):
    element1=int(input())
    list1.append(element1)
print(list1)
n2=int(input('Enter number of elements in list 2:'))
for i in range(0,n2):
    element2=int(input())
    list2.append(element2)
print(list2)
result=list(map(add,list1,list2))
print('The sum of elements in list :' + str(result))
```

6. Using MAP and FILTER find the cube of all odd numbers from given input list.

```
arr = []
n=int(input('Enter the number of elements in list:'))
print('Enter list elements:')
for i in range(0,n):
    arr.append(int(input()))
print(arr)
print('The cube of odd elements in list:')
arr2=[]
arr2=list(map(lambda x:x **3 ,filter(lambda x: x%2 !=0,arr)))
print(arr2)
```



CONCLUSION: In this experiment we understood about functions and their different uses to solve a question in a simpler and easier manner and understood more about properties of functions that it is block of code that only runs when you call them, they accept data with arguments and can return data and many more.

Lab Experiment – 04

AIM: To implement Classes, Objects and Inheritance

THEORY:

Classes: A class is a user-defined blueprint or prototype from which objects are created. Classes provide a means of bundling data and functionality together. Creating a new class creates a new type of object, allowing new instances of that type to be made. Each class instance can have attributes attached to it for maintaining its state. Class instances can also have methods (defined by their class) for modifying their state.

Objects: An Object is an instance of a Class. A class is like a blueprint while an instance is a copy of the class with actual values. It's not an idea anymore, it's an actual dog, like a dog of breed pug who's seven years old. You can have many dogs to create many different instances, but without the class as a guide, you would be lost, not knowing what information is required.

Inheritance: Inheritance is the capability of one class to derive or inherit the properties from another class. It represents real-world relationships well.It provides the **reusability** of a code. We don't have to write the same code again and again. Also, it allows us to add more features to a class without modifying it.It is transitive in nature, which means that if class B inherits from another class A, then all the subclasses of B would automatically inherit from class A.

IMPLEMENTATION:

1. Explain Classes and Objects with suitable examples.

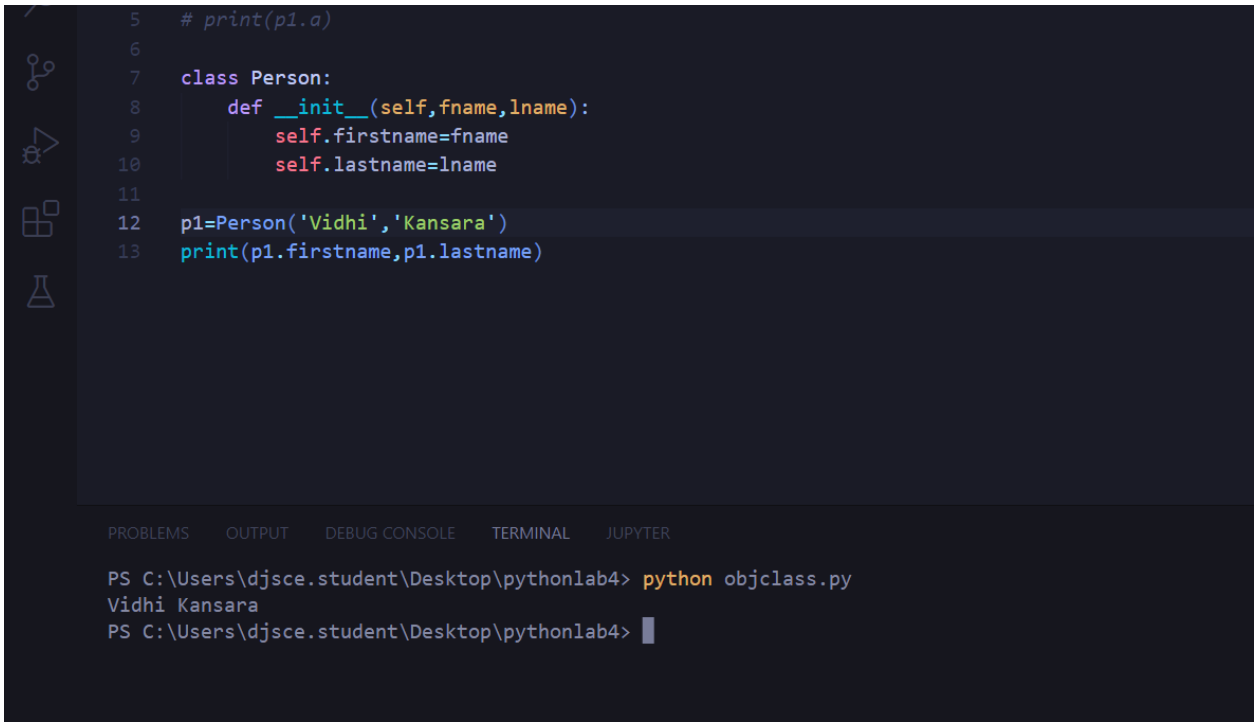
```
class Person:
    def __init__(self,fname,lname):
        self.firstname=fname
        self.lastname=lname
class Person:
    def __init__(self,fname,lname):
        self.firstname=fname
        self.lastname=lname
    def printname(self):
        print(self.firstname,self.lastname)
p1=Person('Vidhi','Kansara')
p1.printname()
```

```
# init constructor
class Person:
    def __init__(self,fname,lname):
        self.firstname=fname
        self.lastname=lname
class Person:
    def __init__(self,fname,lname):
        self.firstname=fname
        self.lastname=lname
```

```
PROBLEMS  OUTPUT  TERMINAL  JUPYTER  DEBUG CONSOLE

PS C:\Study\Academics\python\lab4> python -u "c:\Study\Academics\python\lab4\objclass.py"
Vidhi Kansara
PS C:\Study\Academics\python\lab4>
```

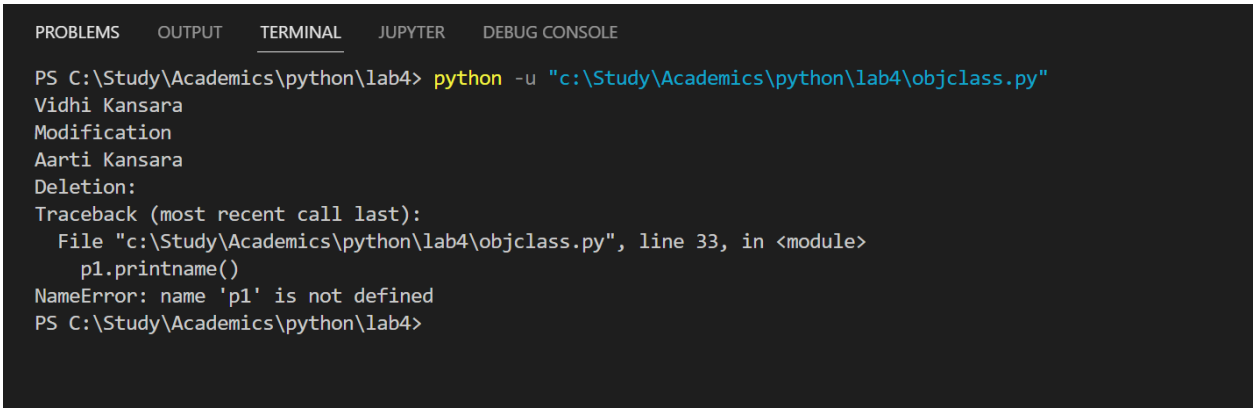
2. Explain use of __init__() function in class with suitable example.

__init__(): It is known as a constructor. The __init__ method can be called when an object is created from the class, and access is required to initialize the attributes of the class.



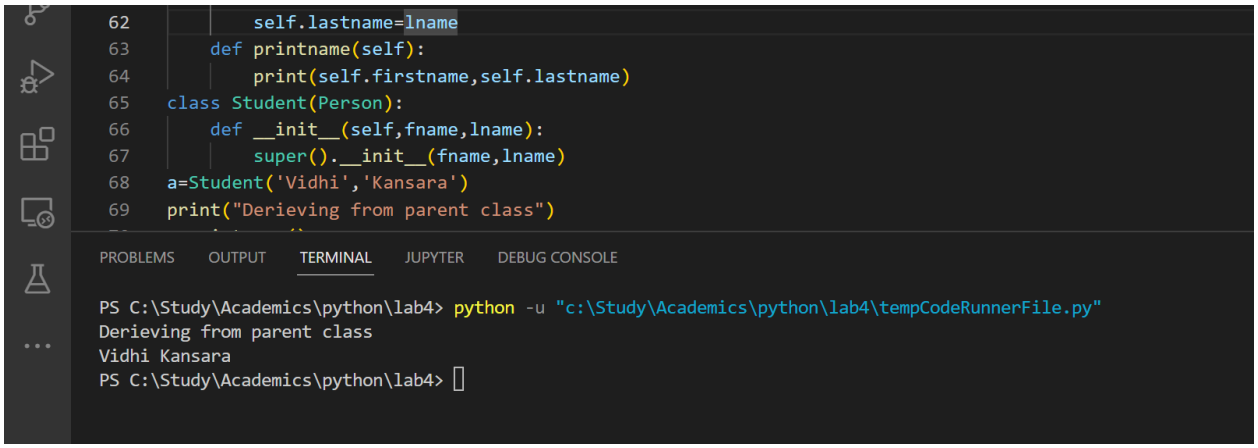
3. Code for Object Methods, Modifying Object Properties and Deleting Objects.

```
class Person:
    def __init__(self,fname,lname):
        self.firstname=fname
        self.lastname=lname
class Person:
    def __init__(self,fname,lname):
        self.firstname=fname
        self.lastname=lname
    def printname(self):
        print(self.firstname,self.lastname)
p1=Person('Vidhi','Kansara')
p1.printname()
class Person:
    def __init__(self,fname,lname):
        self.firstname=fname
        self.lastname=lname
    def printname(self):
        print(self.firstname,self.lastname)
p1=Person('Vidhi','Kansara')
print("Modification")
p1.firstname=('Aarti')
p1.printname()
print("Deletion:")
del(p1)
p1.printname()
```



4. Python Inheritance with suitable examples.

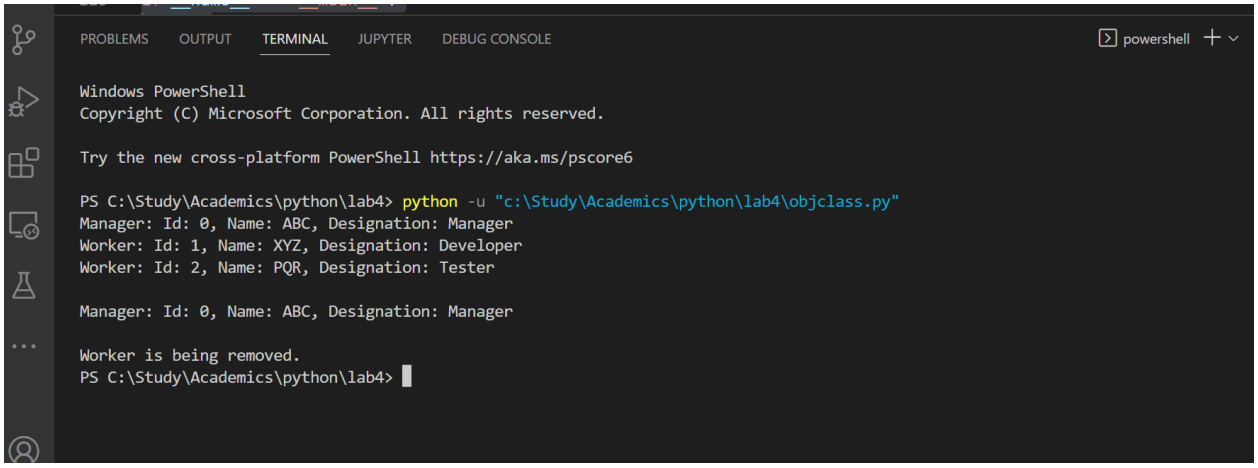
```
class Person:
    def __init__(self,fname,lname):
        self.firstname=fname
        self.lastname=lname
    def printname(self):
        print(self.firstname,self.lastname)
class Student(Person):
    def __init__(self,fname,lname):
        super().__init__(fname,lname)
a=Student('Vidhi','Kansara')
print("Derieving from parent class")
a.printname()
```



5. Write python code to implement the following inheritance example:Classes: Employee, Developer, Tester, ManagerDeveloper, tester, Manager inherit EmployeeManager handles Developer, testerManager class : implement functions to add Developer/Tester and Remove Developer/ TesterDisplay .. to see the list of employees he manages

```
from typing import Union, Sequence

class Employee:

    _id: int = 0

    name: str = ""

    designation: str = ""

    def __init__(self, **kwargs):

        self._id = kwargs["_id"]

        self.name = kwargs["name"]

        self.designation = kwargs["designation"]

    def __str__(self):

        return f"Id: {self._id}, Name: {self.name}, Designation:
{self.designation}\n"


class Developer(Employee):

    def __init__(self, **kwargs):

        super().__init__(**kwargs)


class Tester(Employee):

    def __init__(self, **kwargs):

        super().__init__(**kwargs)


Worker = Union[Developer, Tester]


class Manager(Employee):

    _developers: Sequence[Worker] = []

    def __init__(self, **kwargs):

        super().__init__(**kwargs)
```

```
    def add_worker(self, worker: Worker):
        self._developers.append(worker)
    def remove_worker(self, worker_id: int) -> bool:
        for worker in self._developers:
            if worker._id == worker_id:
                self._developers.remove(worker)
                return True
        return False
    def __str__(self):
        details = f"Manager: {super().__str__()}"
        for worker in self._developers:
            details += f"Worker: {worker.__str__()}"
        return details

def main():
    manager = Manager(_id=0, name="ABC", designation="Manager")
    worker1 = Developer(_id=1, name="XYZ", designation="Developer")
    worker2 = Tester(_id=2, name="PQR", designation="Tester")
    manager.add_worker(worker1)
    manager.add_worker(worker2)
    print(manager)
    manager.remove_worker(worker1._id)
    manager.remove_worker(worker2._id)
    print(manager)
    print("Worker is being removed.")

if __name__ == "__main__":
 main()
```



CONCLUSION: In this experiment we learnt about classes, object and inheritance in python, the interrelation of classes and objects that class is a collection of objects and understood inheritance and thus, implemented it using an example.

Lab Experiment – 05

AIM: To understand and implement exception handling.

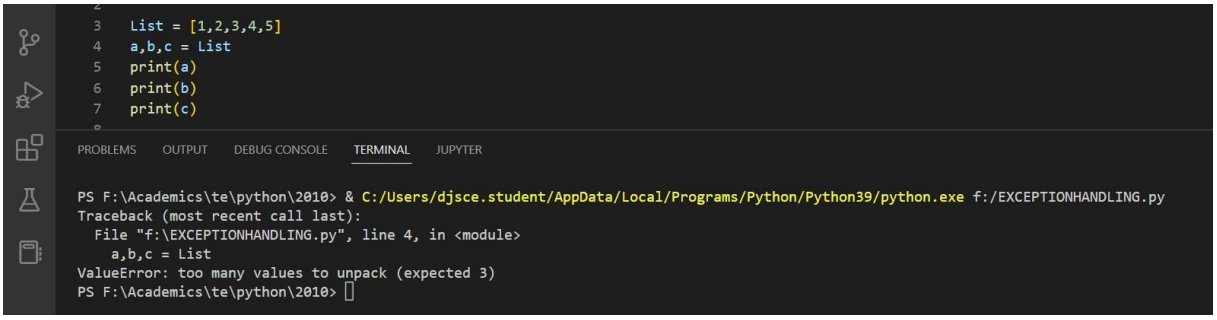
THEORY: Errors are the problems in a program due to which the program will stop the execution. On the other hand, exceptions are raised when some internal events occur which changes the normal flow of the program. Two types of error occur in python.

1) Syntax errors - When the proper syntax of the language is not followed then a syntax error is thrown.
2) Logical errors - When in the runtime an error that occurs after passing the syntax test is called exception or logical type. They are also known as Exceptions. Exceptions are the unusual event that occurs during the execution of the program that interrupts the normal flow of the program. Generally, exceptions occur when the code written encounters a situation it cannot cope with. Whenever an exception is raised, the program stops the execution, and thus the further code is not executed. Therefore, an exception is a python object that represents a run-time error. An exception is a Python object that represents an error. There are different types of exception:
   a) ValueError: It gets raised when the built-in function for a data type has the valid type of arguments, but the arguments have invalid values specified.
   b) ArithmeticError: It is the base class for all errors related to the numericcalculation.
   c) ImportError: It gets raised when an import statement fails.
   d) LookupError: It is the base class for all lookup errors.
   e) KeyboardInterrupt: This gets raised when the user interrupts the programexecution, usually by pressing Ctrl+C.

IMPLEMENTATION:

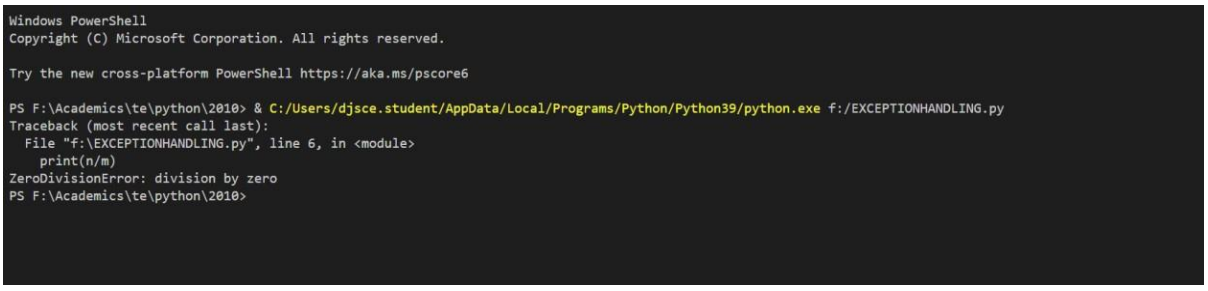
1) Display and handle in-built exceptions:
   a) ValueError

```
List = [1,2,3,4,5]
a,b,c = List
print(a)
print(b)
print(c)
```



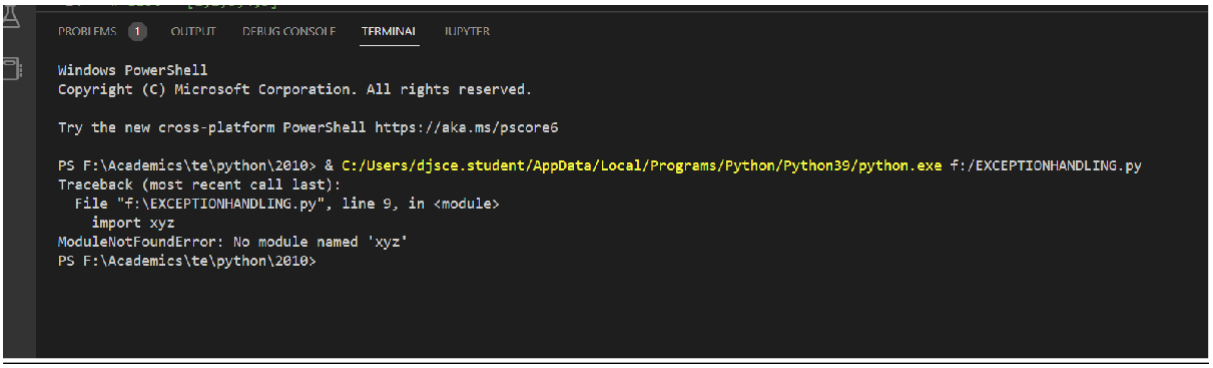
   b) ArithmeticError
      i) ZeroDivisionError

```
n = 9
m = 0
print(n/m)
```



   c) ImportError

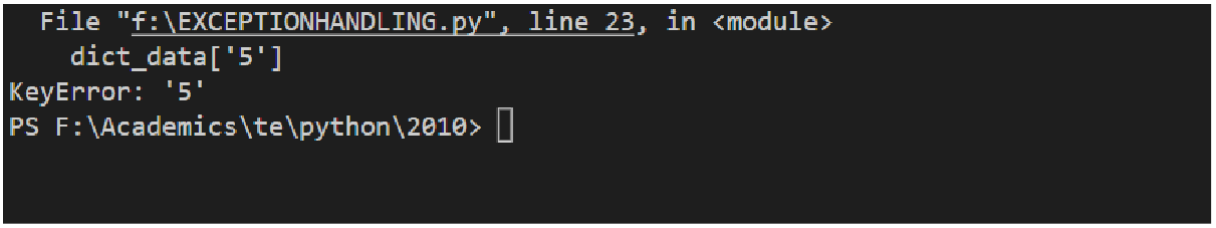
```
import xyz
list_data = [1, 2, 3]
list_data[4]
```



d) LookupError

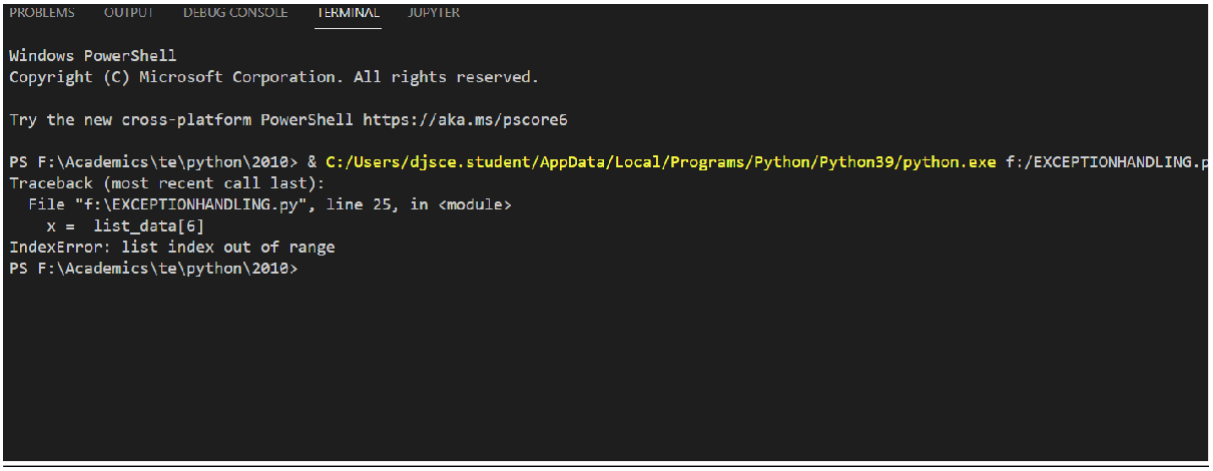
i) KeyError

```
dict_data={'2' : 'two', '4' : 'four', '6' : 'six'}
dict_data['5']
```



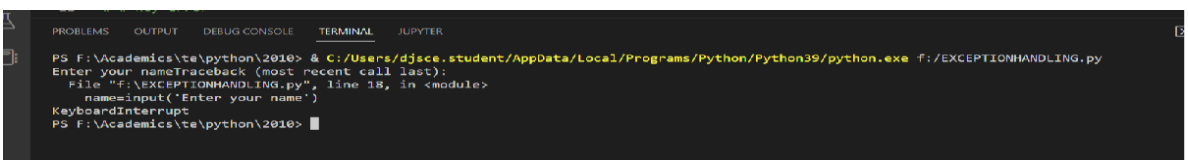
ii) IndexError

```
list_data = [1, 2, 3, 4, 5]
x = list_data[6]
```



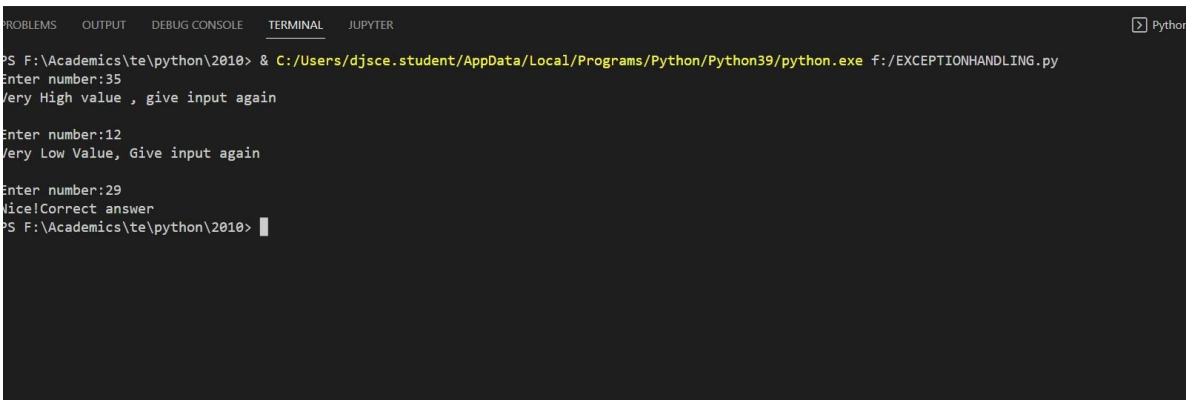
e) KeyboardInterrupt

```
name=input('Enter your name')
```



2) Create a user-defined exception handling

```
class BaseError(Exception):pass
class HighValueError(Exception):pass
class LowValueError(Exception):pass
value = 29
while(1):
     try:
          n=int(input("Enter number:"))if n>
          value:
               raise HighValueErrorelif
          n < value:
               raise LowValueError
     except LowValueError:
          print("Very Low Value, Give input again")print()
     except HighValueError:
          print("Very High value , give input again")print()
     else:
          print("Nice!Correct answer")break
```



CONCLUSION: Thus, from this experiment we learnt about errors and exceptions and how they are implemented in python using various keywords try, except and finally.Exception handling has various advantages like making code more readable, preventing crashes and many more. Thus, we can conclude that exception is an important part of python.

Lab Experiment – 06

AIM: To understand and implement file handling.

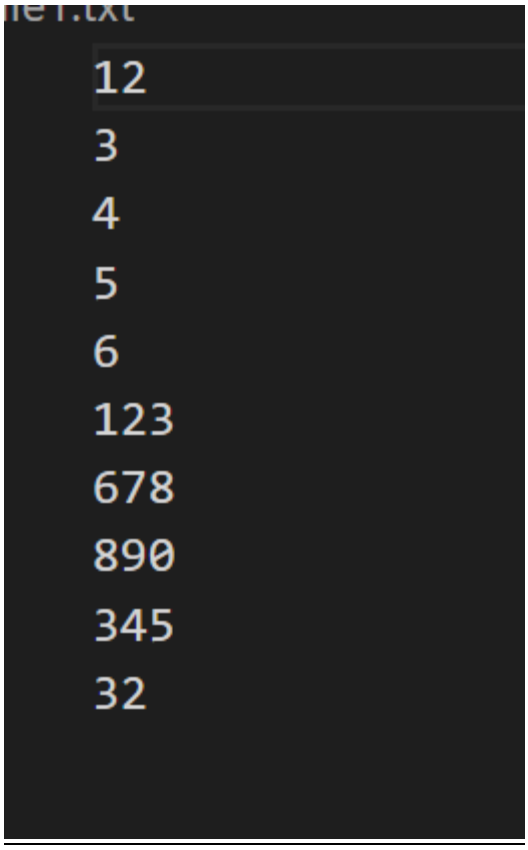
THEORY: Python supports file handling and allows users to handle files i.e., to read and write files, along with many other file handling options, to operate on files. The concept of file handling has stretched over various other languages, but the implementation is either complicated or lengthy, but like other concepts of Python, this concept here is also easy and short. Python treats files differently as text or binary and this is important. Each line of code includes a sequence of characters and they form a text file. Each line of a file is terminated with a special character, called the EOL or End of Line characters like comma {,} or newline character. It ends the current line and tells the interpreter a new one has begun. Let's start with the reading and writing files. There are various modes supported in python file handling features.

1. **r:** open an existing file for a read operation.
2. **w:** open an existing file for a write operation. If the file already contains some data, then it will be overridden but if the file is not present then it creates the file as well.
3. **a:** open an existing file for append operation. It won't override existing data.
4. **r+:** To read and write data into the file. The previous data in the file will be overridden.
5. **w+:** To write and read data. It will override existing data.
6. **a+:** To append and read data from the file. It won't override existing data.

IMPLEMENTATION:

1) Read contents of file and sort the data and put it in another file.

Input file:

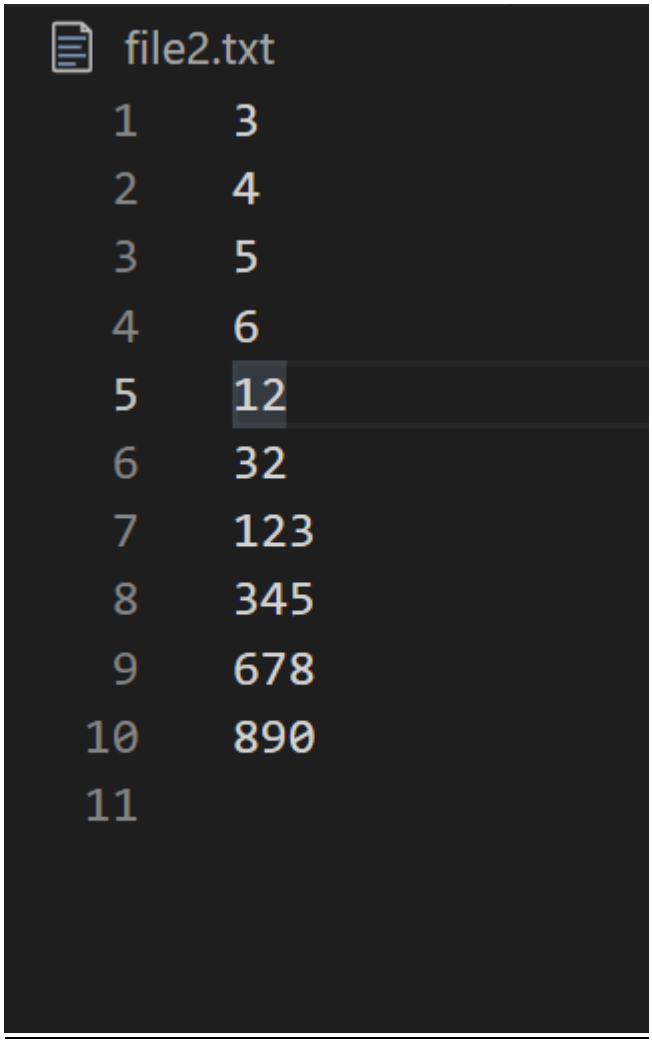


Code:

```
print("Enter 10 numbers")
List1 = list()
fileWrite = open("file1.txt", "w")
for i in range(10):
    List1.append(int(input("Enter number : ")))
    fileWrite.write(str(List1[i]))
    fileWrite.write("\n")
fileWrite.close()
#read from the file
fileRead = open("file1.txt", "r")
List = list()
print(fileRead)
for line in fileRead:
    List.append(int(line))
```

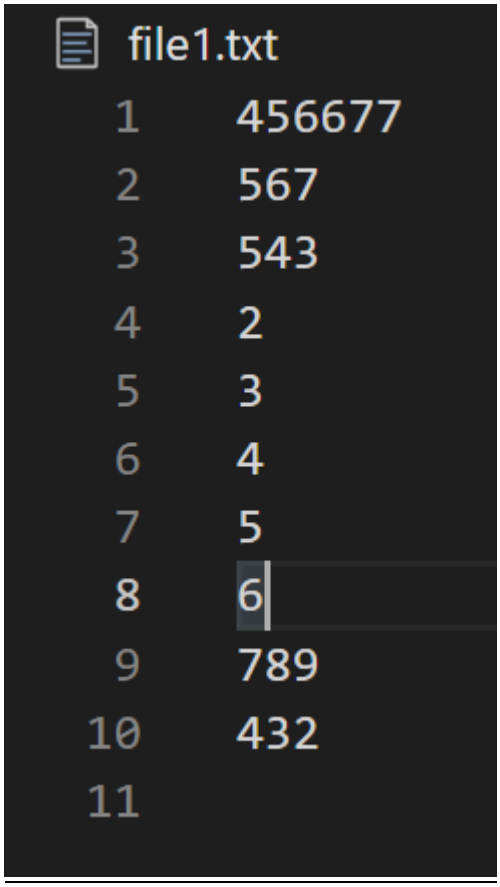
```
List.sort()
print(List)
fileWrite1 = open("file2.txt", "w")
for i in range(len(List)):
    fileWrite1.write(str(List[i]))
    fileWrite1.write("\n")
```

Output file:



2) Read contents of file and sort the data lexicographically and put it in another file.

Input file:

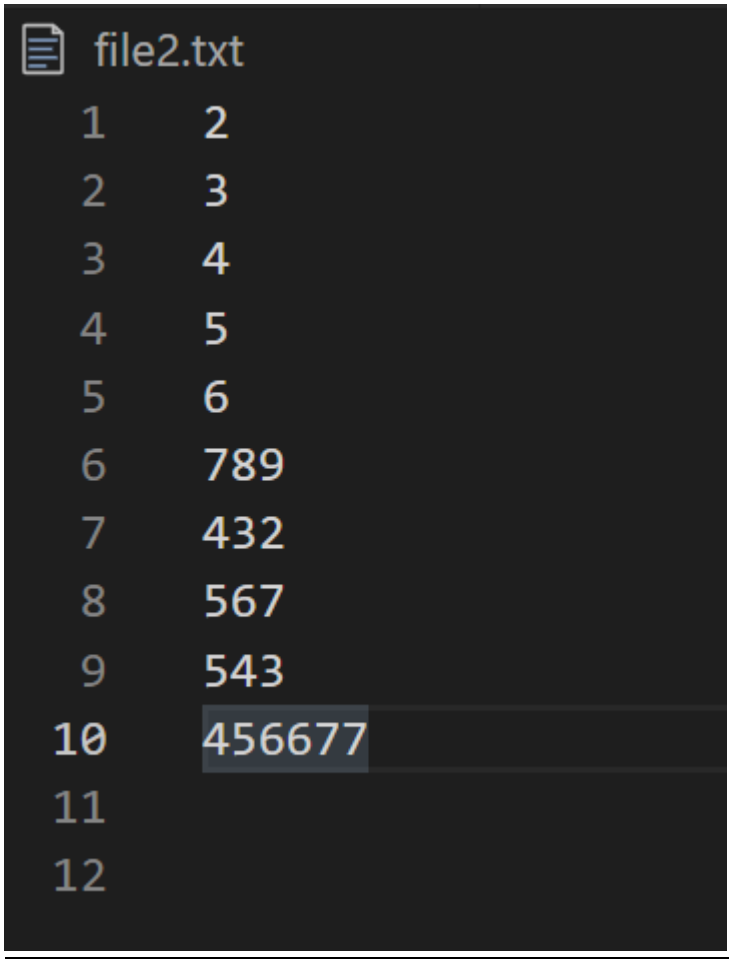


Code:

```
print("Enter 10 numbers")
List1 = list()
fileWrite = open("file1.txt", "w")
for i in range(10):
    List1.append(int(input("Enter number : ")))
    fileWrite.write(str(List1[i]))
    fileWrite.write("\n")
fileWrite.close()
fileRead = open("file1.txt","r")
List = list()
```

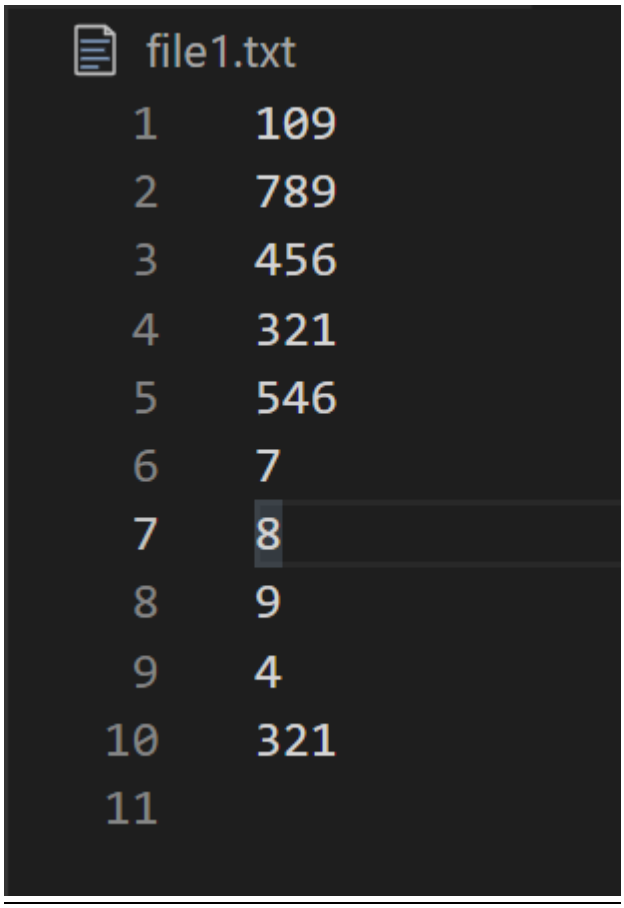
```
f=fileRead.read()
f = f.split(" ")
for line in f:
    List.append(str(line))
    List.sort(key = len)
print(List)
List.sort()
fileWrite = open("file2.txt","w")
for i in range(len(List)):
    fileWrite.write(str(List[i]))
    fileWrite.write("\n")
```

Output file:



3) Read contents of file and sort the data in reverse and put it in another file.

Input file:

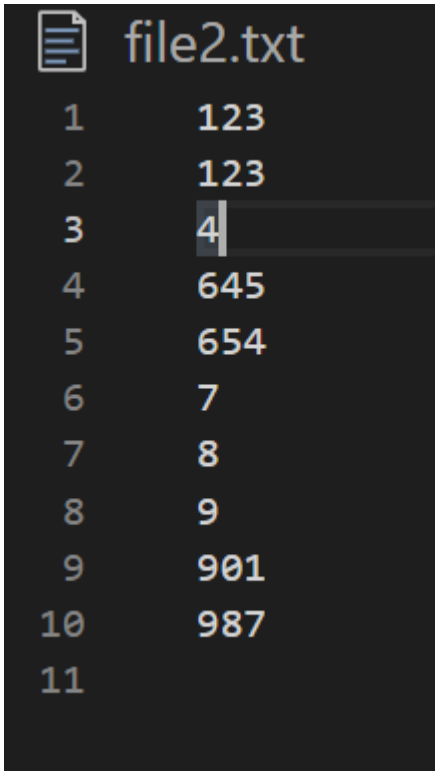


Code:

```
print("Enter 10 numbers")
List1 = list()
fileWrite = open("file1.txt", "w")
for i in range(10):
    List1.append(int(input("Enter number : ")))
    fileWrite.write(str(List1[i]))
    fileWrite.write("\n")
fileWrite.close()
fileRead = open("file1.txt","r")
List = list()
f=fileRead.read()
```

```
f=f.split()
for line in f:
    List.append(str(line)[::-1]
List.sort()
fileWrite = open("file2.txt","w")
for i in range(len(List)):
    fileWrite.write(str(List[i]))
    fileWrite.write("\n")
```

Output file:



CONCLUSION: Thus, from this experiment we learnt about how files are handled and manipulated using python language. It is a useful feature present in python. Here, python treats **binary and text files differently** and this should be kept in mind while working. Also, each file ends with an **EOF indicator** to tell the interpreter that the file has ended.

Lab Experiment – 07

AIM: To understand and implement regular expressions in python.

THEORY: A **Regular Expressions** is a special sequence of characters that uses a search pattern to find a string or set of strings. It can detect the presence or absence of a text by matching it with a particular pattern, and also can split a pattern into one or more sub-patterns

Python has various functions and modules in-built to display and manipulate regular expression.

1. "re" module : supports the use of regex in Python. Its primary function is to offer a search, where it takes a regular expression and a string.
2. re.findall() : Return all non-overlapping matches of pattern in string, as a list of strings. The string is scanned left-to-right, and matches are returned in the order found.
3. re.compile() : Regular expressions are compiled into pattern objects, which have methods for various operations such as searching for pattern matches or performing string substitutions.
4. re.split() : Split string by the occurrences of a character or a pattern, upon finding that pattern, the remaining characters from the string are returned as part of the resulting list.

There are various special sequences used in regular expression.

Special Sequence	Description	
\A	Matches if the string begins with the given character	\Afor
\b	Matches if the word begins or ends with the given character. \b(string) will check for the beginning of the word and (string)\b will check for the ending of the word.	\bge
\B	It is the opposite of the \b i.e. the string should not start or end with the given regex.	\Bge
\d	Matches any decimal digit, this is equivalent to the set class [0-9]	\d
\D	Matches any non-digit character, this is equivalent to the set class [^0-9]	\D
\s	Matches any whitespace character.	\s

Special Sequence	Description	
\S	Matches any non-whitespace character	\S
\w	Matches any alphanumeric character, this is equivalent to the class [a-zA-Z0-9_].	\w
\W	Matches any non-alphanumeric character.	\W
\Z	Matches if the string ends with the given regex	ab\Z

IMPLEMENTATION:

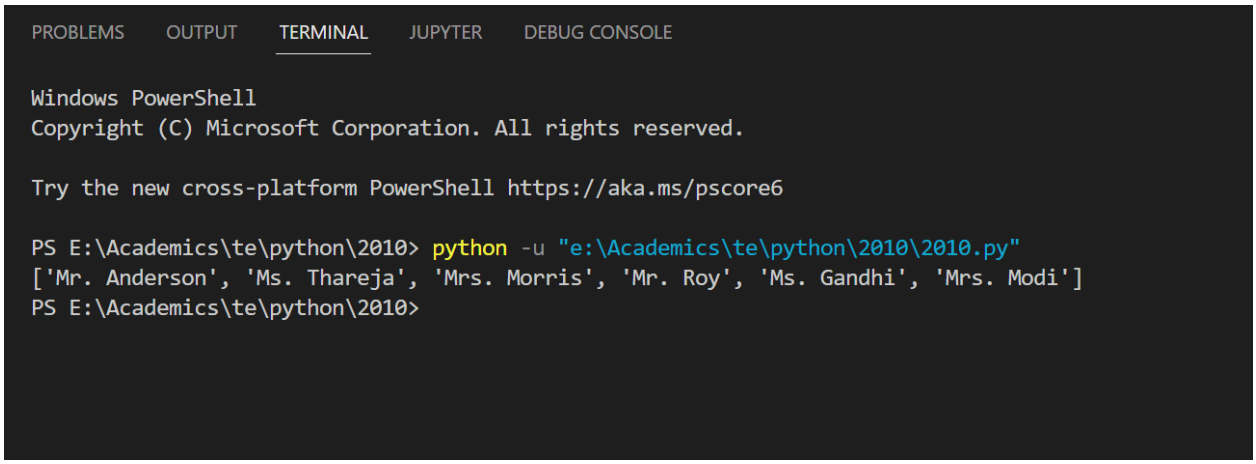
Use regular expression for the text in “Sample.txt” and find:

“Sample.txt” :

```
Mr. Anderson
Ms. Thareja
Mrs. Morris
Mr. Roy
Ms. Gandhi
Mrs. Modi
https://www.google.com
http://www.udemy.com
www.udacity.com
https://www.stackoverflow.com
http://www.djsce.ac.in
https://plus.google.com
rishit.grover@gmail.com
kapeesh.grover@yahoo.co.in
abhishek.shah@gmail.com
shahp98@gmail.com
demo_user@gmail.com
rolflmoa@yahoo.co.in
27777647
233*333*88
455-78-888
022-240-93836
02642*221*381
```

1. Names of the User.

```
import re

file = open('Sample.txt','r')

text = file.read()

pattern = r'M(?:r\.|rs\.|s\.) [a-zA-Z]+'

names = re.findall(pattern,text)

print(names)
```



2. Website name excluding http/s

```
import re

file = open('Sample.txt','r')

text = file.read()

pattern = r"(?i)\b((?:https?://|www\d{0,3}[.]|[a-z0-9.\-]+[.][a-
z]{2,4}/)(?:[^\s()<>]+|\(([^\s()<>]+|(\([^\s()<>]+\)))\))+(?:\(([^\s()<>]
+|(\([^\s()<>]+\)))\)|[^\s`!()\[\]{};:'\".,<>?«»“”‘’]))"

website = re.findall(pattern,text)

print(website)
```

```
Try the new cross-platform PowerShell https://aka.ms/pscore6

PS E:\Academics\te\python\2010> python -u "e:\Academics\te\python\2010\2010.py"
[('https://www.google.com', '', '', '', ''), ('http://www.udemy.com', '', '', '', ''), ('www.udacity.com', '', '', '', ''), ('https://w
ww.stackoverflow.com', '', '', '', ''), ('http://www.djsce.ac.in', '', '', '', ''), ('https://plus.google.com', '', '', '', '')]
PS E:\Academics\te\python\2010>
```

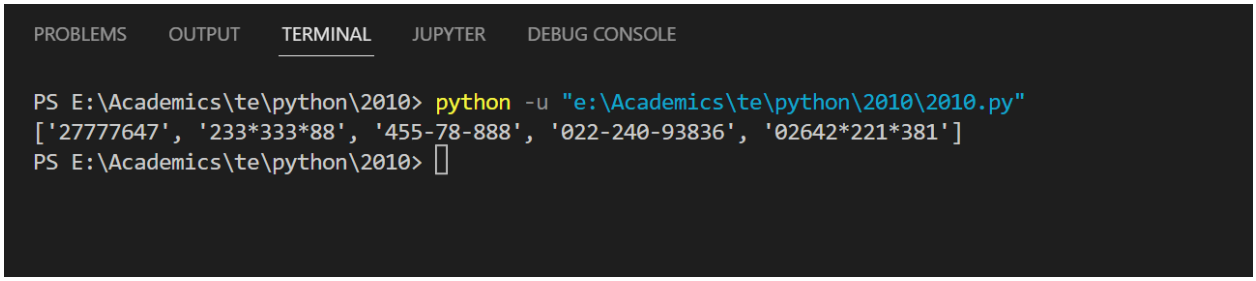
3. Identify email ids

```
import re

file = open('Sample.txt','r')

text = file.read()

pattern = r'[a-zA-z0-9\.\-+_]+@[a-zA-Z0-9\.\-+_]+\.[a-zA-Z]+'

email = re.findall(pattern,text)

print(email)
```

```
PROBLEMS  OUTPUT  TERMINAL  JUPYTER  DEBUG CONSOLE                    Code

PS E:\Academics\te\python\2010> python -u "e:\Academics\te\python\2010\2010.py"
['rishit.grover@gmail.com', 'kapeesh.grover@yahoo.co.in', 'abhishek.shah@gmail.com', 'shahp98@gmail.com', 'demo_user@gmail.com', 'rolfl
moa@yahoo.co.in']
PS E:\Academics\te\python\2010>
```

4. Identify Phone numbers

```
import re

file = open('Sample.txt','r')

text = file.read()

pattern = r'\d{1,5}.\d{1,3}.\d{1,5}'

phone = re.findall(pattern,text)

print(phone)
```



CONCLUSION: Thus, in this experiment we learnt about regular expression and implemented them using python language. Regular expression has multiple benefits like Searching and replacing text in files, validating text input, such as password and email address, ability to rename a hundred files at a time and many more. Thus, regular expression is a useful feature that simplify our day-to-day problems.

Lab Experiment – 08

AIM: To implement database connectivity in python

THEORY: MySQL Connector/Python enables Python programs to access MySQL databases, using an API that is compliant with the Python Database API Specification v2.0 (PEP 249). It is written in pure Python and does not have any dependencies except for the Python Standard Library.**Steps to create connection of python application to database**:

1. Import mysql.connector module
2. Create the connection object.
3. Create the cursor object
4. Execute the query

Creating cursor objects: The cursor object can be defined as an abstraction specified in the Python DB-API 2.0. It facilitates us to have multiple separate working environments through the same connection to the database. We can create the cursor object by calling the 'cursor' function of the connection object. The cursor object is an important aspect of executing queries to the databases.

IMPLEMENTATION:

1. Create Database

```
import mysql.connector
 mydb = mysql.connector.connect(
  host="localhost",
  user="root",
  password="root",
  )
# create db
mycursor = mydb.cursor()
mycursor.execute("CREATE DATABASE electronics")
```

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Study\Academics\python\python_211> python -u "c:\Study\Academics\python\python_211\databaseconnector.py"
<mysql.connector.connection_cext.CMySQLConnection object at 0x000001CADF9B0E20>
PS C:\Study\Academics\python\python_211>
```

2. Create Table

```
import mysql.connector
mydb = mysql.connector.connect(
  host="localhost",
  user="root",
  password="root",
  database="electronics"
)
mycursor = mydb.cursor()
mycursor.execute("""CREATE TABLE Laptop (
Id int(11) NOT NULL,
Name varchar(250) NOT NULL,
Price float NOT NULL,
Purchase_date Date NOT NULL,
```

```
        PRIMARY KEY (Id))""")
```

```
PS C:\Study\Academics\python\python_211> python -u "c:\Study\Academics\python\python_211\databaseconnector.py"
('laptop',)
PS C:\Study\Academics\python\python_211>
```

3. Insert values (Show examples for Single Row and Multiple Rows)
   a. Single Rows:

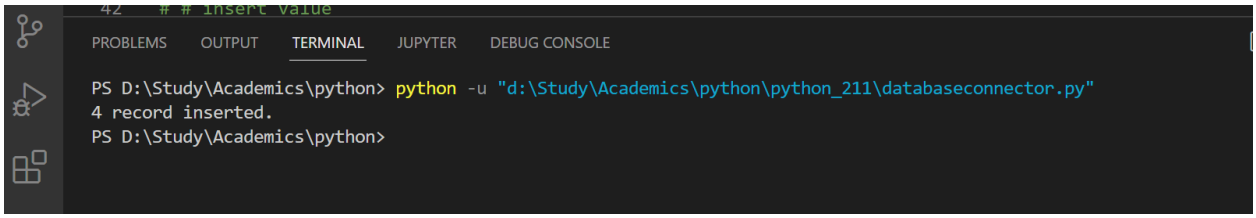
```
import mysql.connector
mydb = mysql.connector.connect(
  host="localhost",
  user="root",
  password="root",
  database="electronics"
)
mycursor = mydb.cursor()
mycursor.execute("""INSERT INTO Laptop (Id, Name, Price,
   Purchase_date)
    VALUES
    (15, 'Lenovo ThinkPad P71', 6459, '2019-08-14') """)
mydb.commit()
print(mycursor.rowcount, "record inserted.")
```

```
PROBLEMS  OUTPUT  TERMINAL  JUPYTER  DEBUG CONSOLE

PS D:\Study\Academics\python> python -u "d:\Study\Academics\python\python_211\databaseconnector.py"
1 record inserted.
PS D:\Study\Academics\python>
```

   b. Multiple Rows:

```
import mysql.connector
mydb = mysql.connector.connect(
  host="localhost",
```

```
  user="root",
  password="root",
  database="electronics"
)
mycursor = mydb.cursor()
sql = """INSERT INTO Laptop (Id, Name, Price, Purchase_date)
                            VALUES
                            (%s,%s,%s,%s) """
val = [(2, 'Area 51M', 6999, '2019-04-14'),(3, 'MacBook Pro', 2499,
'2019-06-20'),
          (4, 'Asus', 8000, '2020-04-14'),(5, 'HP', 15000, '2019-05-
10')]
mycursor.executemany(sql, val)
mydb.commit()
print(mycursor.rowcount, "record inserted.")
```



4. Delete a row based on values

```
import mysql.connector

mydb = mysql.connector.connect(
  host = "localhost",
  user = "root",
  password = "root",
  database = "electronics"
  )
mycursor = mydb.cursor()
```

```
mycursor.execute("DELETE FROM Laptop WHERE NAME = 'Asus' ")
mydb.commit()
print(mycursor.rowcount, "record(s) deleted")
```

```
Try the new cross-platform PowerShell https://aka.ms/pscore6

PS D:\Study\Academics\python> python -u "d:\Study\Academics\python\python_211\databaseconnector.py"
1 record(s) deleted
PS D:\Study\Academics\python>
```

5. Display the rows of the table

```
import mysql.connector
mydb = mysql.connector.connect(
  host = "localhost",
  user = "root",
  password = "root",
  database = "electronics"
  )
mycursor = mydb.cursor()
mycursor.execute("SELECT * FROM Laptop")
myresult = mycursor.fetchall()
for x in myresult:
  print(x)
```

```
PS D:\Study\Academics\python> python -u "d:\Study\Academics\python\python_211\databaseconnector.py"
(2, 'Area 51M', 6999.0, datetime.date(2019, 4, 14))
(3, 'MacBook Pro', 2499.0, datetime.date(2019, 6, 20))
(5, 'HP', 15000.0, datetime.date(2019, 5, 10))
(15, 'Lenovo ThinkPad P71', 6459.0, datetime.date(2019, 8, 14))
PS D:\Study\Academics\python>
```

6. Update the values of a specific row

```
# update table
import mysql.connector
mydb = mysql.connector.connect(
```

```
  host="localhost",
  user="root",
  password="root",
  database="electronics"
)
mycursor = mydb.cursor()
mycursor.execute("UPDATE Laptop SET Price = 25000 WHERE NAME =
   'HP' ")
mydb.commit()
print(mycursor.rowcount, "record(s) deleted")
mydb.commit()
mycursor = mydb.cursor()
mycursor.execute("SELECT * FROM Laptop")
myresult = mycursor.fetchall()
for x in myresult:
  print(x)
```

```
PS C:\Users\Kansara> python -u "d:\Study\Academics\python\python_211\databaseconnector.py"
0 record(s) deleted
(2, 'Area 51M', 6999.0, datetime.date(2019, 4, 14), None)
(3, 'MacBook Pro', 2499.0, datetime.date(2019, 6, 20), None)
(5, 'HP', 25000.0, datetime.date(2019, 5, 10), None)
(15, 'Lenovo ThinkPad P71', 6459.0, datetime.date(2019, 8, 14), None)
PS C:\Users\Kansara>
```

7. Search whether a particular record is present in the table or not

```
import mysql.connector
mydb = mysql.connector.connect(
  host="localhost",
  user="root",
  password="root",
  database="electronics"
)
mycursor = mydb.cursor()
```

```
mycursor.execute("SELECT * FROM Laptop WHERE Name LIKE 'A%' ")
myresult = mycursor.fetchall()
for x in myresult:
  print(x)
```

```
PS C:\Users\Kansara> python -u "d:\Study\Academics\python\python_211\tempCodeRunnerFile.py"
(2, 'Area 51M', 6999.0, datetime.date(2019, 4, 14), None)
PS C:\Users\Kansara>
```

8. Alter the table by adding new column

```
import mysql.connector
mydb = mysql.connector.connect(
  host = "localhost",
  user = "root",
  password = "root",
  database = "electronics"
  )
mycursor = mydb.cursor()
mycursor.execute("ALTER TABLE Laptop ADD Year int(10) ")
mydb.commit()
mycursor = mydb.cursor()
mycursor.execute("SELECT * FROM Laptop")
myresult = mycursor.fetchall()
for x in myresult:
  print(x)
```

```
PS C:\Users\Kansara> python -u "d:\Study\Academics\python\python_211\tempCodeRunnerFile.py"
(2, 'Area 51M', 6999.0, datetime.date(2019, 4, 14), None)
(3, 'MacBook Pro', 2499.0, datetime.date(2019, 6, 20), None)
(5, 'HP', 25000.0, datetime.date(2019, 5, 10), None)
(15, 'Lenovo ThinkPad P71', 6459.0, datetime.date(2019, 8, 14), None)
PS C:\Users\Kansara>
```

9. Delete the table

```
import mysql.connector
mydb = mysql.connector.connect(
  host="localhost",
  user="root",
  password="root",
  database="electronics"
)
mycursor = mydb.cursor()
mycursor.execute("DROP TABLE Laptop")
print('Table Dropped')
mydb.commit()
```

```
PS C:\Users\Kansara> python -u "d:\Study\Academics\python\python_211\databaseconnector.py"
Table Dropped
PS C:\Users\Kansara>
```

10. Delete the database

```
import mysql.connector
mydb = mysql.connector.connect(
  host="localhost",
  user="root",
  password="root",
)
mycursor = mydb.cursor()
mycursor.execute("DROP DATABASE electronics")
print('Database dropped')
```

```
mydb.close()
```

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\Kansara> python -u "d:\Study\Academics\python\python_211\databaseconnector.py"
Database dropped
PS C:\Users\Kansara>
```

CONCLUSION: Thus, from this experiment we learnt about managing MYSQL queries in python and implemented it. Also, we understood the interrelation and similarities between MYSQL queries and python cursor objects.

Lab Experiment – 09

AIM: Implement a client server communication application based on socket Programming.

THEORY: Socket programming is a way of connecting two nodes on a network to communicate with each other. One socket(node) listens on a particular port at an IP, while the other socket reaches out to the other to form a connection. The server forms the listener socket while the client reaches out to the server.
They are the real backbones behind web browsing. In simpler terms, there is a server and a client.

Server: A server has a bind () method which binds it to a specific IP and port so that it can listen to incoming requests on that IP and port. A server has a listen () method which puts the server into listening mode. This allows the server to listen to incoming connections. And last a server has an accept () and close () method. The accept method initiates a connection with the client and the close method closes the connection with the client.

Client: Client is used to interact with server. We could tenet to the server like this just to know that our server is working

CODE:

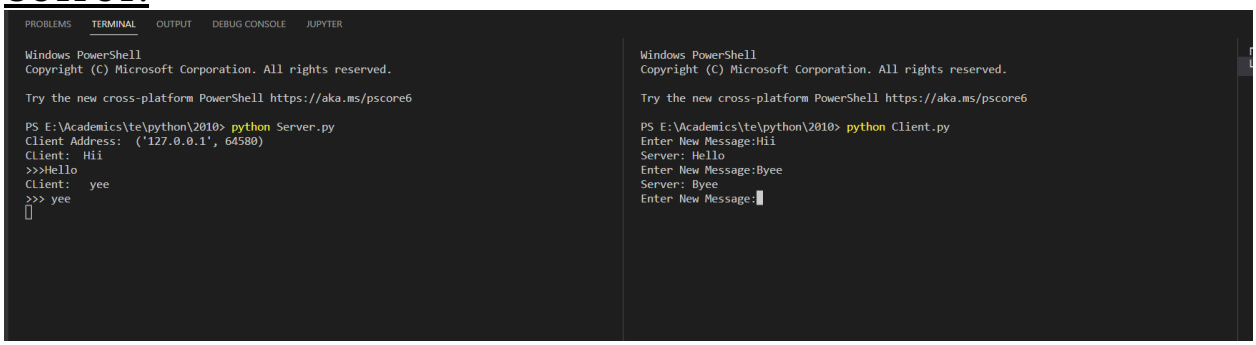
- **Client:**

```
import socket


def mpm():
    host = '127.0.0.1'
    port = 6000
    s = socket.socket() # socket pakage from socket method
    s.connect((host,port))
    while True:
        x = input('Enter New Message:')
        y = x.encode('ascii')
        s.send(y)
        data = s.recv(1024)
        d = data.decode('ascii')
        print('Server:',d)
mpm()
```

- **Server:**

```
import socket

def mpm():
    host = '127.0.0.1'
    port = 6000
    s = socket.socket()
    s.bind((host,port))
    s.listen(1)
    c, addr = s.accept()
    print("Client Address: ", addr)
    while True:
        data = c.recv(1024)
        d = data.decode('ascii')
        print('CLient: ', d)
        x = input(">>>")
        y = x.encode('ascii')
        c.send(y)

mpm()
```

OUTPUT:



CONCLUSION: Thus, from this experiment we learnt about how client and server application function using socket programming in python. We also learnt about different functions like bind(), accept(), etc in socket programming and implemented it in this program.

Lab Experiment -10

AIM: Design a GUI application to show input and output operations using Tkinter.

THEORY: Tkinter is the most commonly used library for developing GUI (Graphical User Interface) in Python. It is a standard Python interface to the Tk GUI toolkit shipped with Python. As Tk and Tkinter are available on most of the Unix platforms as well as on the Windows system, developing GUI applications with Tkinter becomes the fastest and easiest.

Steps to create a GUI application in Tkinter module:

- Import the Tkinter module.
- Create the GUI application main window.
- Add one or more of the above-mentioned widgets to the GUI application.
- Enter the main event loop to take action against each event triggered by the user.

Tkinter Widgets:

Tkinter provides various controls, such as buttons, labels and text boxes used in a GUI application. These controls are commonly called widgets.

- **Button:**

The Button widget is used to display buttons in your application.

- **Label:**

The Label widget is used to provide a single-line caption for other widgets. It can alsocontain images.

- **Text:**

The Text widget is used to display text in multiple lines.

- **Frame:**

The Frame widget is used as a container widget to organize other widgets.

Geometry Management:

All Tkinter widgets have access to specific geometry management methods, which have the purpose of organizing widgets throughout the parent widget area. Tkinter exposes the following geometry manager classes: pack, grid, and place.

- The **pack()** Method − This geometry manager organizes widgets in blocks before placing them in the parent widget.
- The **grid()** Method − This geometry manager organizes widgets in a table-like structure in the parent widget.
- The **place()** Method − This geometry manager organizes widgets by placing them in a specific position in the parent widget.

CODE:

```
import math
from tkinter import *


class MyWindow:
    def__init__(self, win): self.lbl1=Label(win, text='Number
        1:')
        self.lbl2=Label(win, text='Number 2:')
        self.lbl3=Label(win, text='Result')
        self.lbl4=Label(win, text="") self.lbl6=Label(win,
        text="") self.t1=Entry(bd=3) self.t2=Entry(bd=3)
        self.t3=Entry(bd=3)
        self.lbl6.grid(row=0, column=0,columnspan=3, ipadx=20)
        self.lbl1.grid(row=1, column=0,columnspan=1, ipadx=20)
        self.t1.grid(row=1, column=1,columnspan=1, ipadx=20)
        self.lbl2.grid(row=2, column=0,columnspan=1, ipadx=20)
        self.t2.grid(row=2, column=1,columnspan=1, ipadx=20)
        self.lbl4.grid(row=3, column=0, columnspan=3, ipadx=20)
        self.b1=Button(win, text='Add', width=10) self.b1.bind('<Button-1>',
        self.add) self.b2=Button(win, text='Subtract', width=10)
        self.b2.bind('<Button-1>', self.sub) self.b3=Button(win, text='Multiply',
        width=10) self.b3.bind('<Button-1>', self.mul) self.b4=Button(win,
        text='Divide', width=10) self.b4.bind('<Button-1>', self.div)
```

```
        self.b5=Button(win, text='Modulus', width=10) self.b5.bind('<Button-
        1>', self.mod) self.b6=Button(win, text='Sqrt', width=10)
        self.b6.bind('<Button-1>', self.sqroot) self.b7=Button(win, text='Sin',
        width=10) self.b7.bind('<Button-1>', self.sine) self.b8=Button(win,
        text='Cos', width=10) self.b8.bind('<Button-1>', self.cosine)
        self.b9=Button(win, text='Tan', width=10) self.b9.bind('<Button-1>',
        self.tangent) self.b10=Button(win, text='Power', width=10)
        self.b10.bind('<Button-1>', self.power) self.b1.grid(row=4,
        column=0,columnspan=1, ipadx=20) self.b2.grid(row=4,
        column=1,columnspan=1, ipadx=20) self.b3.grid(row=4,
        column=2,columnspan=1, ipadx=20) self.b4.grid(row=5,
        column=0,columnspan=1, ipadx=20) self.b5.grid(row=5,
        column=1,columnspan=1, ipadx=20) self.b6.grid(row=5,
        column=2,columnspan=1, ipadx=20) self.b7.grid(row=6,
        column=0,columnspan=1, ipadx=20) self.b8.grid(row=6,
        column=1,columnspan=1, ipadx=20) self.b9.grid(row=6,
        column=2,columnspan=1, ipadx=20) self.b10.grid(row=7,
        column=1,columnspan=1, ipadx=20)self.lbl5=Label(win, text="")
        self.lbl5.grid(row=8,          column=0,columnspan=1,          ipadx=20)
        self.lbl3.grid(row=9,          column=0,columnspan=1,          ipadx=20)
        self.t3.grid(row=9, column=1,columnspan=1, ipadx=20)
    def                                add(self,event):
        self.t2.config(state='normal')
```

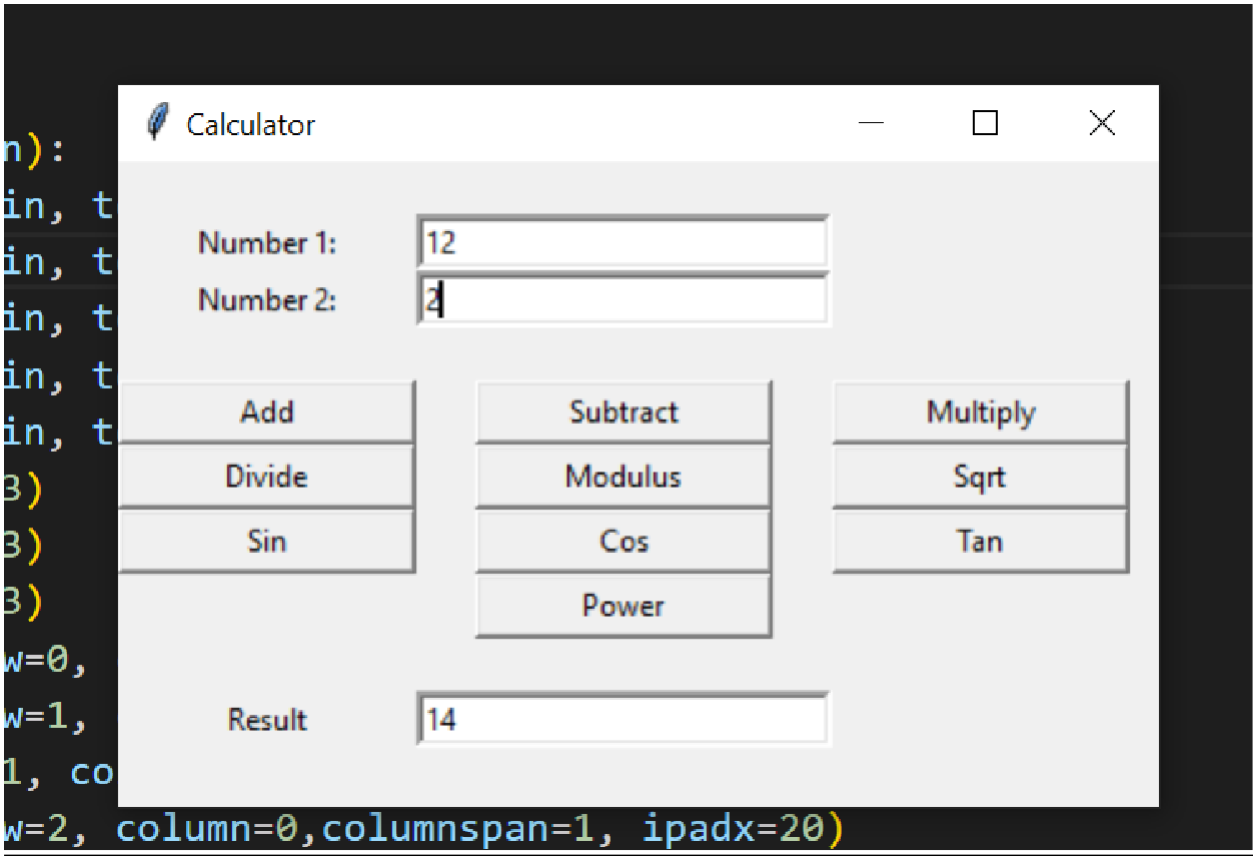
```
        self.t3.delete(0, 'end')
        num1=int(self.t1.get())
        num2=int(self.t2.get()) result=num1+num2
        self.t3.insert(END, str(result))
    def sub(self, event): self.t2.config(state='normal')
        self.t3.delete(0, 'end')
        num1=int(self.t1.get())
        num2=int(self.t2.get()) result=num1-num2
        self.t3.insert(END, str(result))
    def mul(self, event):
        self.t2.config(state='normal')
        self.t3.delete(0, 'end')
        num1=int(self.t1.get())
        num2=int(self.t2.get()) result=num1*num2
        self.t3.insert(END, str(result))
    def div(self, event): self.t2.config(state='normal')
        self.t3.delete(0, 'end')
        num1=int(self.t1.get())
        num2=int(self.t2.get()) result=num1/num2
        self.t3.insert(END, str(result))
    def mod(self, event):
        self.t2.config(state='normal')
```

```
        self.t3.delete(0, 'end')
        num1=int(self.t1.get())
        num2=int(self.t2.get())
        result=num1%num2 self.t3.insert(END,
        str(result))sqroot(self, event):
    def  self.t2.config(state='disabled')
        self.t3.delete(0, 'end')
        num1=int(self.t1.get())
        result=math.sqrt(num1)
        self.t3.insert(END, str(round(result,
                                                    4)))
        self.t3.insert(END, str()) sine(self, event):
    def  self.t2.config(state='disabled') self.t3.delete(0,
        'end') num1=int(self.t1.get())
        result=math.sin(num1*(math.pi/180))
        self.t3.insert(END, str(round(result,


                                                    4)))
    def  cosine(self, event):
        self.t2.config(state='disabled')
        self.t3.delete(0, 'end') num1=int(self.t1.get())
        result=math.cos(num1*(math.pi/180))
        self.t3.insert(END, str(round(result,
                                                    4)))
    def  tangent(self, event):
        self.t2.config(state='disabled')
        self.t3.delete(0, 'end')
        num1=int(self.t1.get())
```

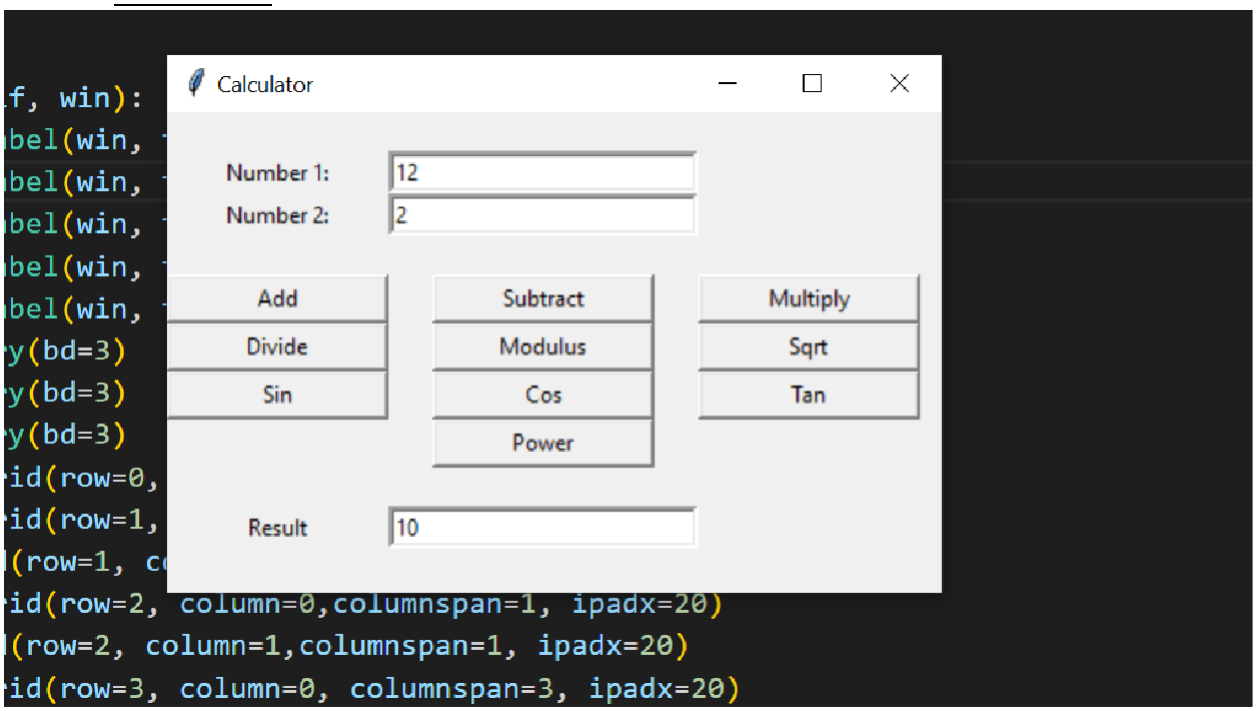
```
        result=math.tan(num1*(math.pi/180)) self.t3.insert(END,
        str(round(result, 4)))
    def power(self, event):
        self.t2.config(state='normal')
        self.t3.delete(0, 'end')
        num1=int(self.t1.get())
        num2=int(self.t2.get()) result=pow(num1,
        num2) self.t3.insert(END, str(result))

window=Tk()
mywin=MyWindow(window)
window.title('Calculator')
window.geometry("420x260+10+10")
window.mainloop()
```

OUPTUT:

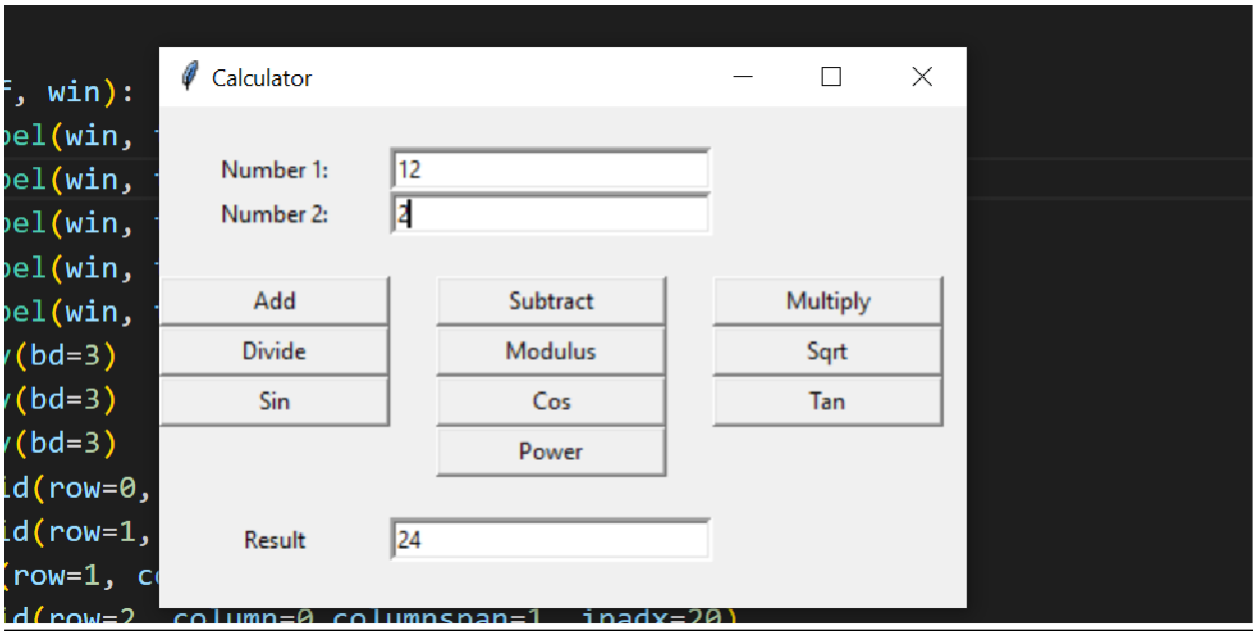
1. Addition:



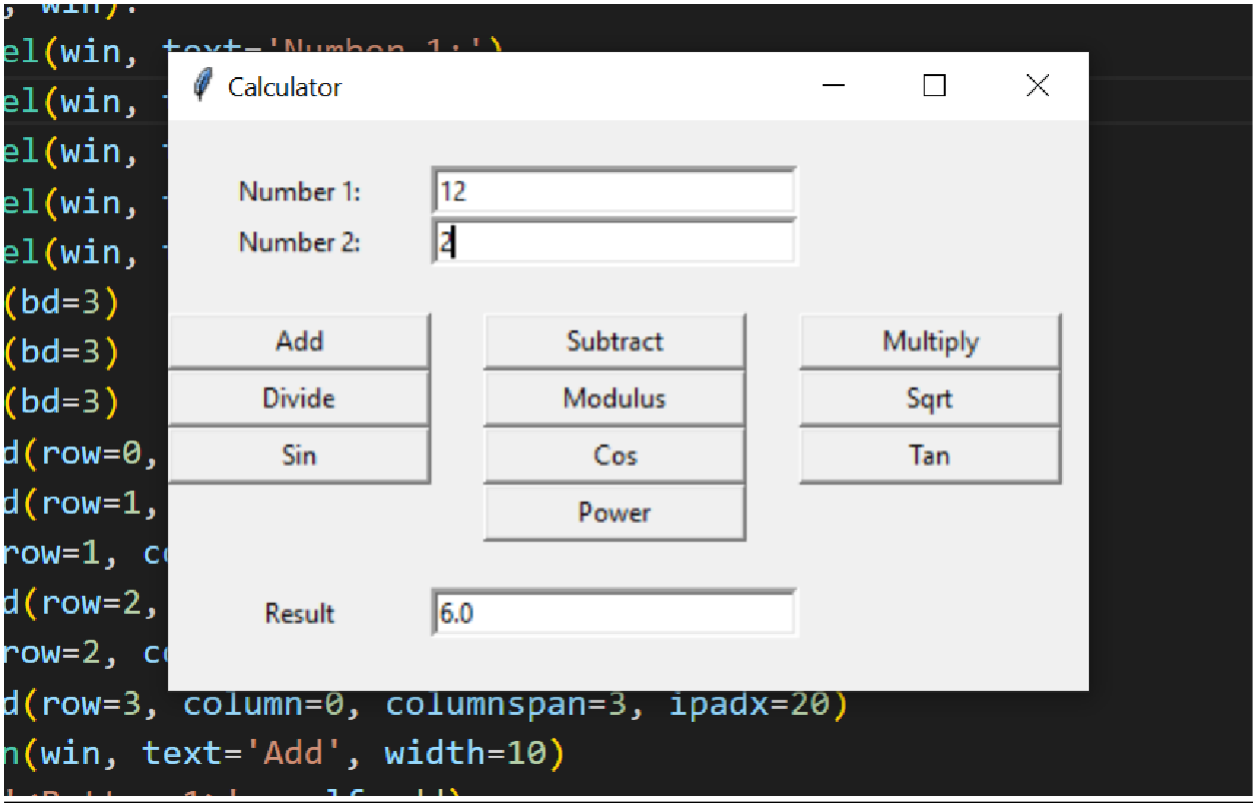
2. Subtraction:



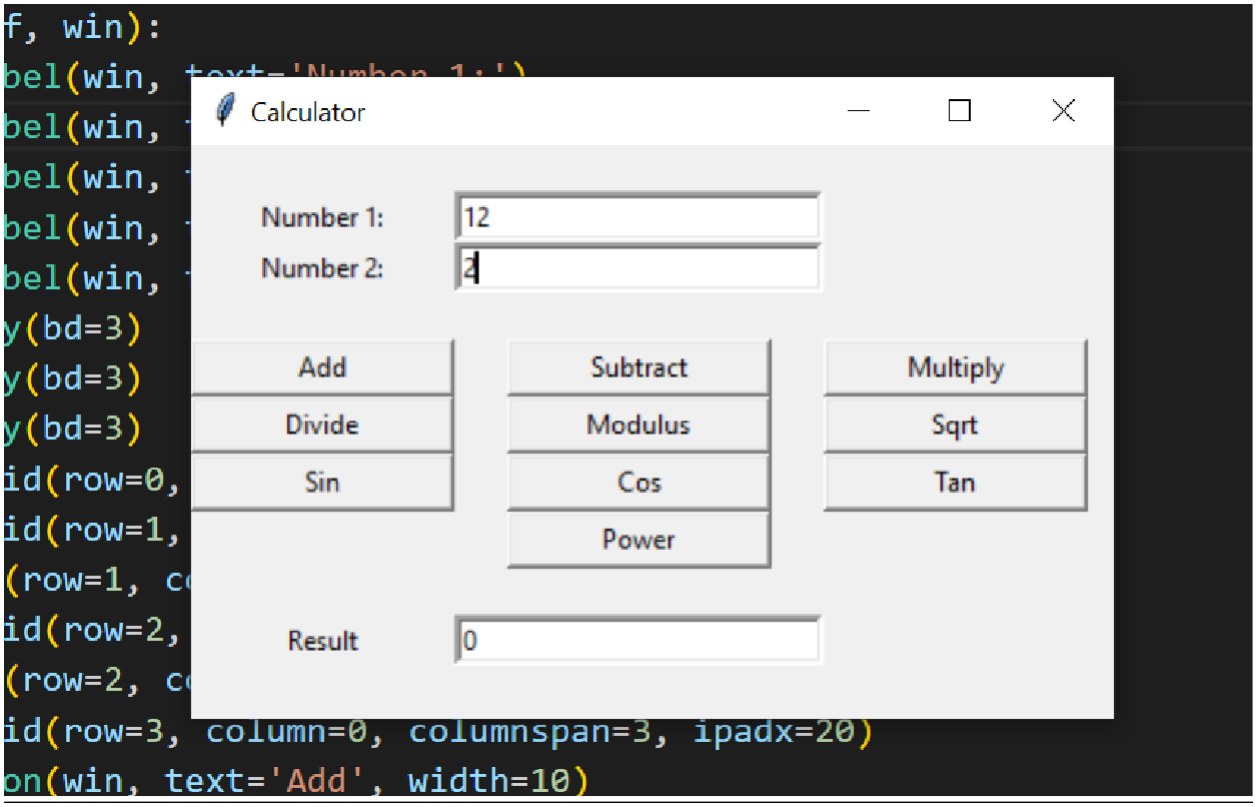
3. Multiplication:



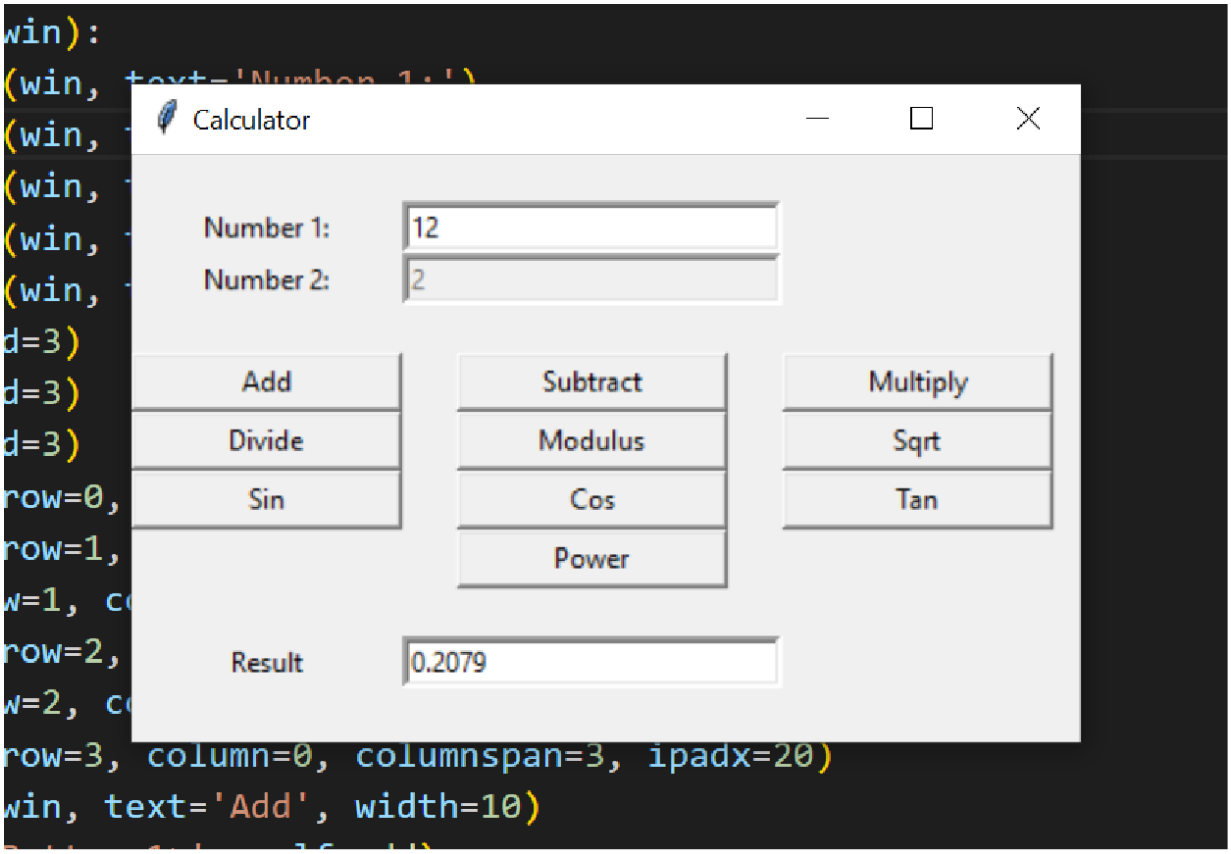
4. Division:



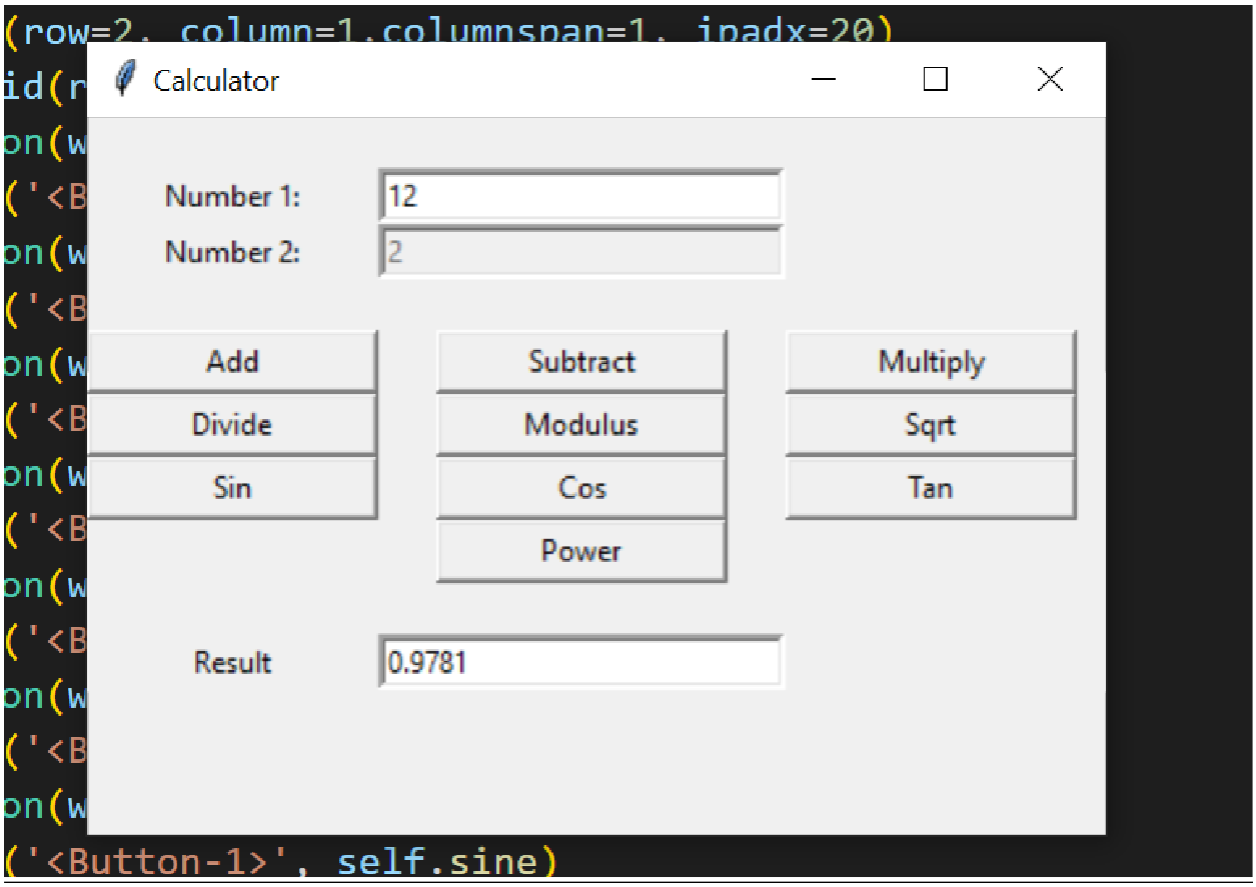
5. Modulus:


Calculator
Number 1:
12
Number 2:
2
Add
Subtract
Multiply
Divide
Modulus
Sqrt
Sin
Cos
Tan
Power
Result
0

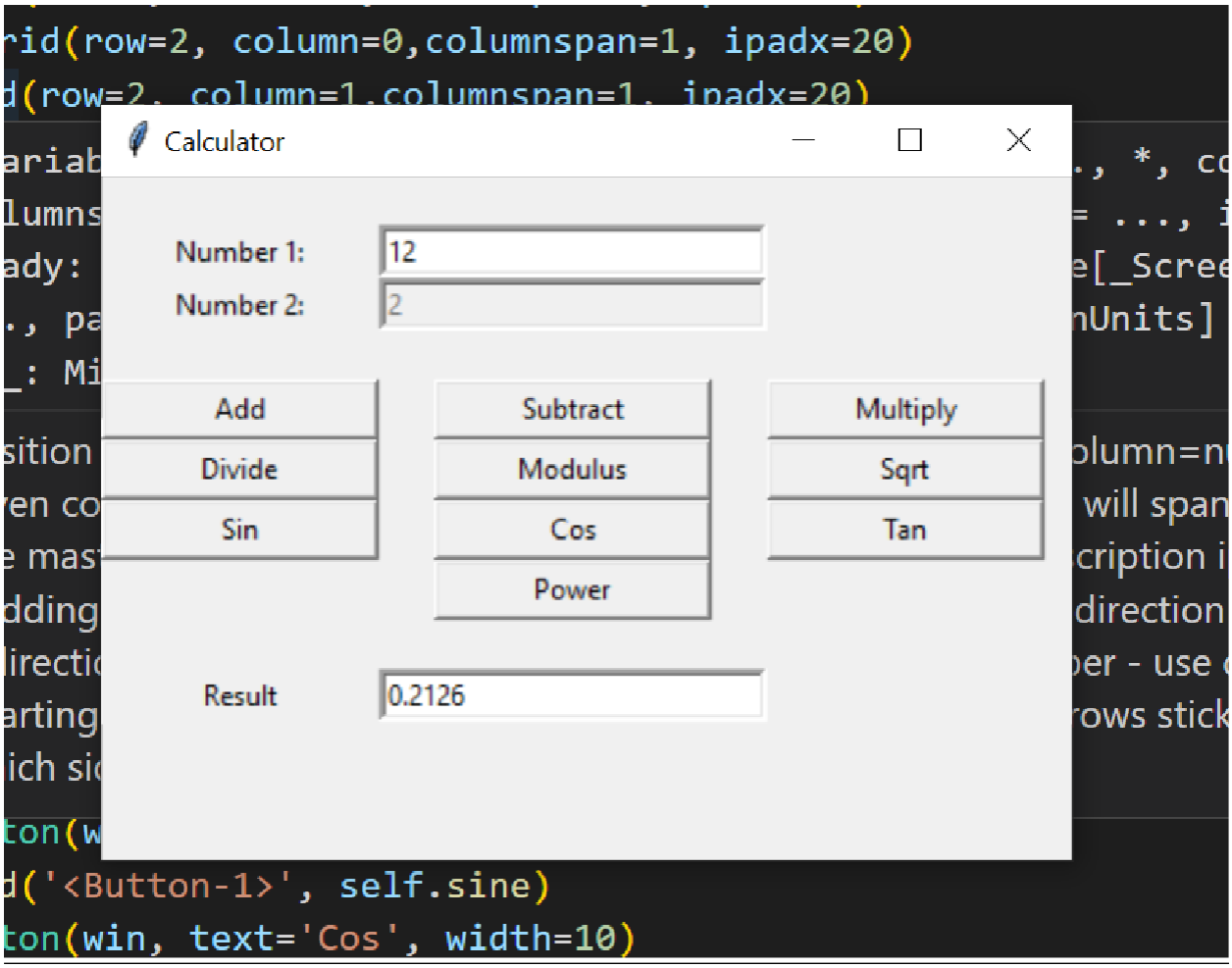
6. Sine:


Calculator
Number 1:
12
Number 2:
2
Add
Subtract
Multiply
Divide
Modulus
Sqrt
Sin
Cos
Tan
Power
Result
0.2079

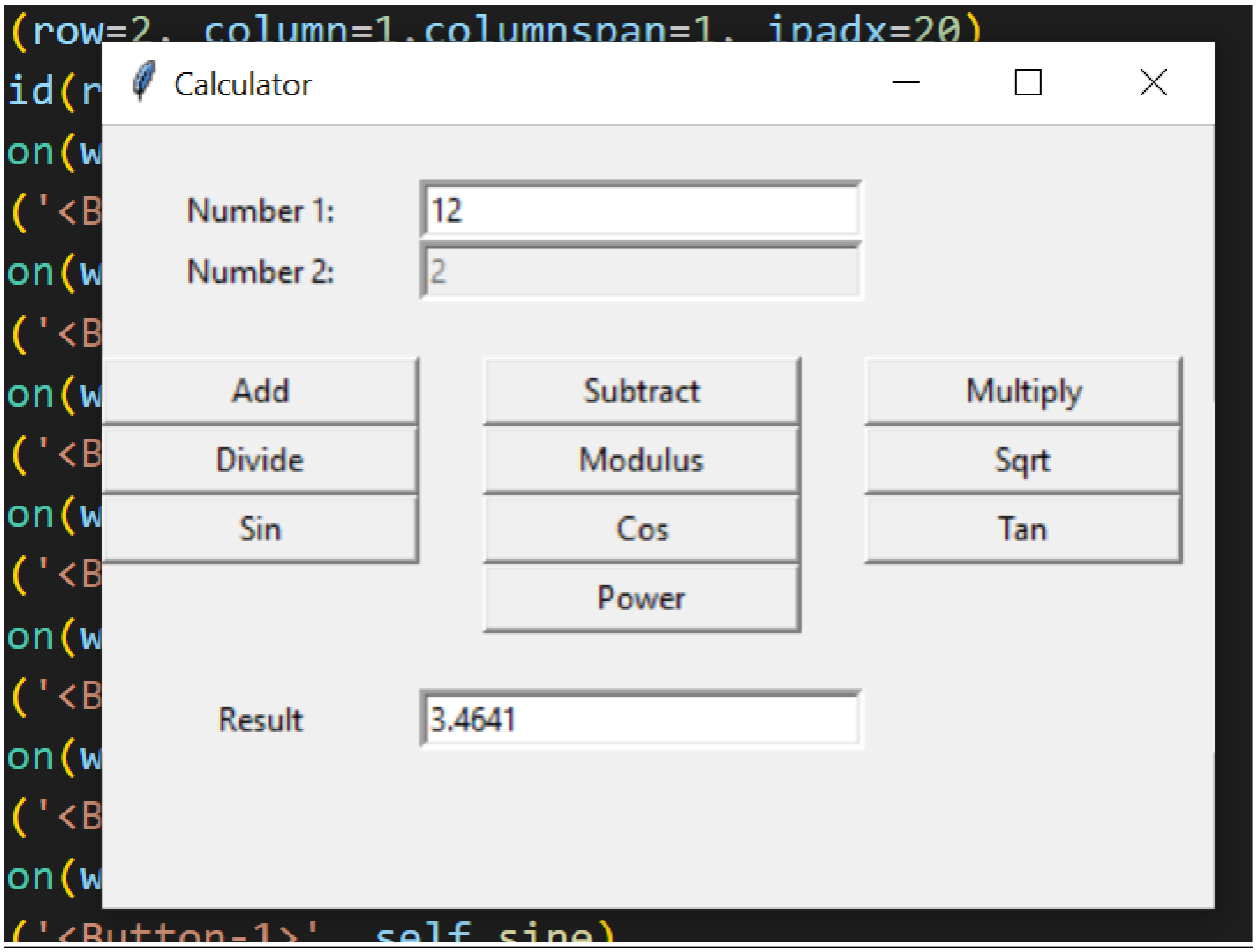
7. Cosine:


(row=2, column=1,columnspan=1, ipadx=20)
('<Button-1>', self.sine)
Calculator
Number 1:
12
Number 2:
2
Add
Subtract
Multiply
Divide
Modulus
Sqrt
Sin
Cos
Tan
Power
Result
0.9781

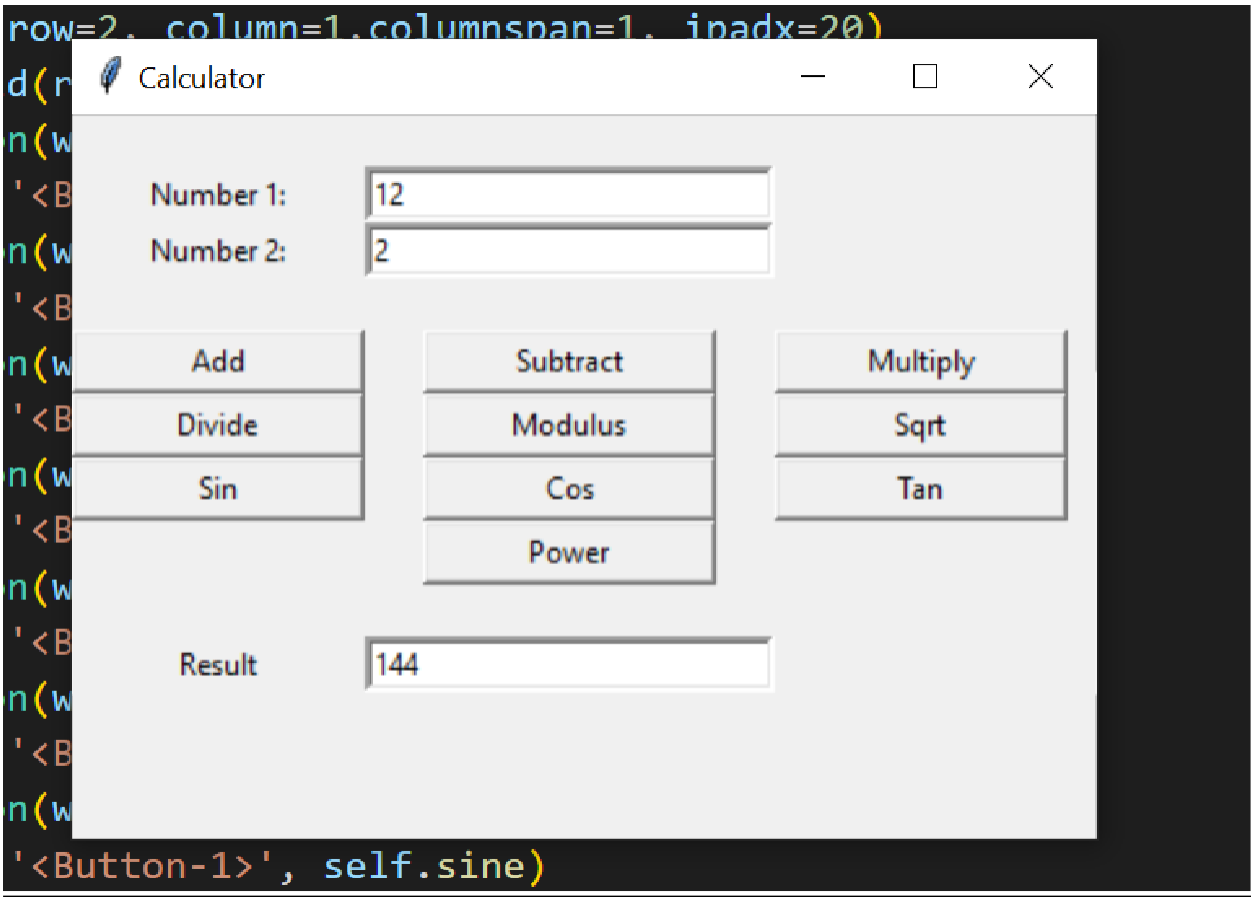
8. Tangent:


Calculator
Number 1:
12
Number 2:
2
Add
Subtract
Multiply
Divide
Modulus
Sqrt
Sin
Cos
Tan
Power
Result
0.2126

9. Square root:



10. Power:



CONCLUSION: In this experiment we learnt about GUI's in python. We also understood about Tkinter in python. We studied about various classes and methods used in GUIs in detailabout the input, output and basic functionalities for our calculator app created.

Lab Experiment -11

AIM: To implement various operations on dataset.

THEORY: Pandas is a Python library used for working with data sets. It has functions for analyzing, cleaning, exploring, and manipulating data. The name "Pandas" has a reference to both "Panel Data", and "Python Data Analysis" and was created by Wes McKinney in 2008.Pandas allows us to analyze big data and make conclusions based on statistical theories. Pandas can clean messy data sets, and make them readable and relevant. Relevant data is very important in data science. Pandas gives you answers about the data, like is there a correlation between two or more columns, what is average value, max value, min value? Pandas are also able to delete rows that are not relevant, or contains wrong values, like empty or NULL values.

CODE:

```
import pandas as pd


dataSeries = {"Name":"Vidhi","Age":19,"Sap id":60004200087,"Branch":"C
omputer Engineering"}


s = pd.Series(data=dataSeries, name="Vidhi details")


linearSeries = [1,'Vidhi','Ketan',True]
s2 = pd.Series(data=linearSeries, name='One Dimensional data')


df = pd.DataFrame({"Vidhi":['DJ','Comps'],"Dhanashri":['Ruia','BMM'],
                   "Amal":['Rajiv Gandhi','Mechanical'],"Muskan":['DTL
','B.Com']}
                  ,index=['College','Dept'])
df


data = pd.read_csv("melb_data.csv")
data.head(10)
```

Rooms	Type	Price	Method	SellerG	Date	Distance	Postcode	...	Bathroom	Car	Landsize	BuildingArea	YearBuilt	CouncilArea	Lattitude	Longtitude	Regionname	Propertycount
2	h	1480000.0	S	Biggin	3/12/2016	2.5	3067.0	...	1.0	1.0	202.0	NaN	NaN	Yarra	-37.7996	144.9984	Northern Metropolitan	4019.0
2	h	1035000.0	S	Biggin	4/02/2016	2.5	3067.0	...	1.0	0.0	156.0	79.0	1900.0	Yarra	-37.8079	144.9934	Northern Metropolitan	4019.0
3	h	1465000.0	SP	Biggin	4/03/2017	2.5	3067.0	...	2.0	0.0	134.0	150.0	1900.0	Yarra	-37.8093	144.9944	Northern Metropolitan	4019.0
3	h	850000.0	PI	Biggin	4/03/2017	2.5	3067.0	...	2.0	1.0	94.0	NaN	NaN	Yarra	-37.7969	144.9969	Northern Metropolitan	4019.0
4	h	1600000.0	VB	Nelson	4/06/2016	2.5	3067.0	...	1.0	2.0	120.0	142.0	2014.0	Yarra	-37.8072	144.9941	Northern Metropolitan	4019.0
2	h	941000.0	S	Jellis	7/05/2016	2.5	3067.0	...	1.0	0.0	181.0	NaN	NaN	Yarra	-37.8041	144.9953	Northern Metropolitan	4019.0
3	h	1876000.0	S	Nelson	7/05/2016	2.5	3067.0	...	2.0	0.0	245.0	210.0	1910.0	Yarra	-37.8024	144.9993	Northern Metropolitan	4019.0
2	h	1636000.0	S	Nelson	8/10/2016	2.5	3067.0	...	1.0	2.0	256.0	107.0	1890.0	Yarra	-37.8060	144.9954	Northern Metropolitan	4019.0
1	u	300000.0	S	Biggin	8/10/2016	2.5	3067.0	...	1.0	1.0	0.0	NaN	NaN	Yarra	-37.8008	144.9973	Northern Metropolitan	4019.0
2	h	1097000.0	S	Biggin	8/10/2016	2.5	3067.0	...	1.0	2.0	220.0	75.0	1900.0	Yarra	-37.8010	144.9989	Northern Metropolitan	4019.0

```
data.isnull().sum()
```

```
Suburb            0
Address           0
Rooms             0
Type              0
Price             0
Method            0
SellerG           0
Date              0
Distance          0
Postcode          0
Bedroom2          0
Bathroom          0
Car              62
Landsize          0
BuildingArea   6450
YearBuilt      5375
CouncilArea    1369
Lattitude         0
Longtitude        0
Regionname        0
Propertycount     0
dtype: int64
```

```
subset_of_data = data.iloc[:,4:]
subset_of_data
```

	Price	Method	SellerG	Date	Distance	Postcode	Bedroom2	Bathroom	Car	Landsize	BuildingArea	YearBuilt	CouncilArea	Lattitude	Longtitude	Regionname	Propertycoun
0	1480000.0	S	Biggin	3/12/2016	2.5	3067.0	2.0	1.0	1.0	202.0	NaN	NaN	Yarra	-37.79960	144.99840	Northern Metropolitan	4019.
1	1035000.0	S	Biggin	4/02/2016	2.5	3067.0	2.0	1.0	0.0	156.0	79.0	1900.0	Yarra	-37.80790	144.99340	Northern Metropolitan	4019.
2	1465000.0	SP	Biggin	4/03/2017	2.5	3067.0	3.0	2.0	0.0	134.0	150.0	1900.0	Yarra	-37.80930	144.99440	Northern Metropolitan	4019.
3	850000.0	PI	Biggin	4/03/2017	2.5	3067.0	3.0	2.0	1.0	94.0	NaN	NaN	Yarra	-37.79690	144.99690	Northern Metropolitan	4019.
4	1600000.0	VB	Nelson	4/06/2016	2.5	3067.0	3.0	1.0	2.0	120.0	142.0	2014.0	Yarra	-37.80720	144.99410	Northern Metropolitan	4019.
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	.
13575	1245000.0	S	Barry	26/08/2017	16.7	3150.0	4.0	2.0	2.0	652.0	NaN	1981.0	NaN	-37.90562	145.16761	South-Eastern Metropolitan	7392.
13576	1031000.0	SP	Williams	26/08/2017	6.8	3016.0	3.0	2.0	2.0	333.0	133.0	1995.0	NaN	-37.85927	144.87904	Western Metropolitan	6380.
13577	1170000.0	S	Raine	26/08/2017	6.8	3016.0	3.0	2.0	4.0	436.0	NaN	1997.0	NaN	-37.85274	144.88738	Western Metropolitan	6380.
13578	2500000.0	PI	Sweeney	26/08/2017	6.8	3016.0	4.0	1.0	5.0	866.0	157.0	1920.0	NaN	-37.85908	144.89299	Western Metropolitan	6380.
13579	1285000.0	SP	Village	26/08/2017	6.3	3013.0	4.0	1.0	1.0	362.0	112.0	1920.0	NaN	-37.81188	144.88449	Western	6543

```
subset_of_data.columns
```

```
subset_of_data.isnull().sum()
```

```
for column in subset_of_data.columns:
  print(f"{column} has {((subset_of_data[column].isnull().sum())/len(subset_of_data))*100:.2f} % values")
```

```
Price has 0.00 % values
Method has 0.00 % values
SellerG has 0.00 % values
Date has 0.00 % values
Distance has 0.00 % values
Postcode has 0.00 % values
Bedroom2 has 0.00 % values
Bathroom has 0.00 % values
Car has 0.46 % values
Landsize has 0.00 % values
BuildingArea has 47.50 % values
YearBuilt has 39.58 % values
CouncilArea has 10.08 % values
Lattitude has 0.00 % values
Longtitude has 0.00 % values
Regionname has 0.00 % values
Propertycount has 0.00 % values
```

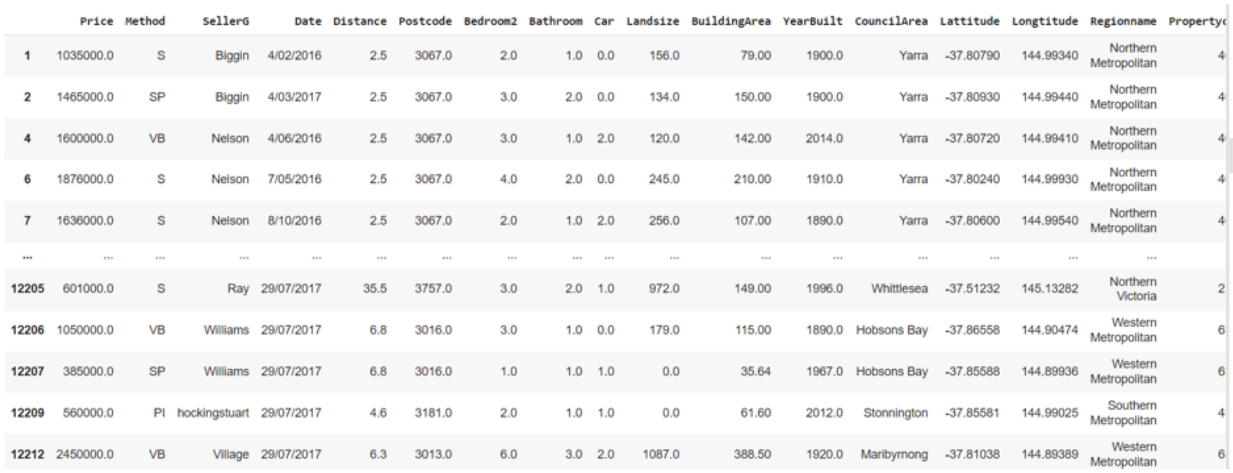
```
subset_of_data.shape
```

```
subset_of_data.shape
```

```
(13580, 17)
```

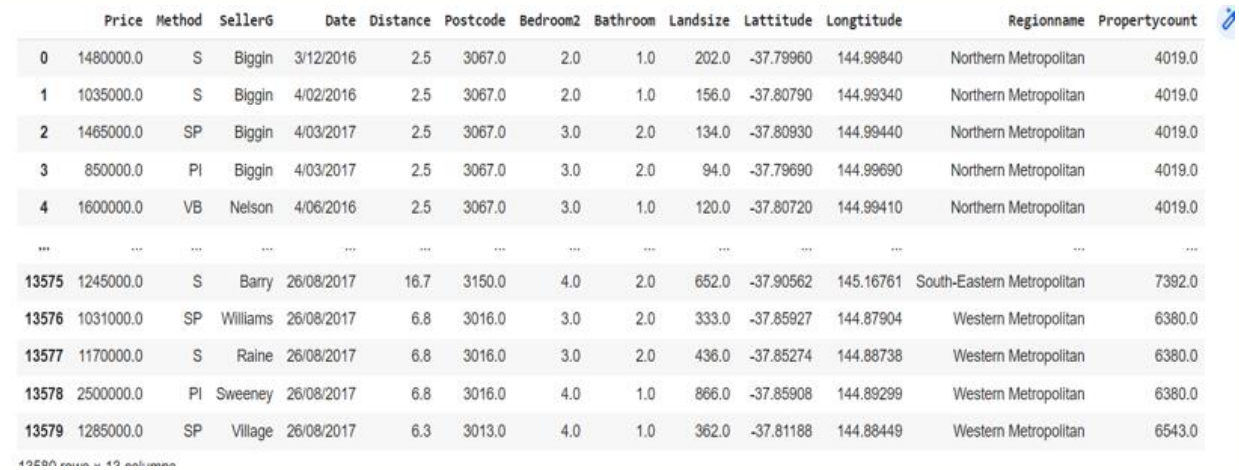
```
missing_rows = subset_of_data.dropna()
```

```
missing_rows
```



	Price	Method	SellerG	Date	Distance	Postcode	Bedroom2	Bathroom	Car	Landsize	BuildingArea	YearBuilt	CouncilArea	Lattitude	Longtitude	Regionname	Propertyc
1	1035000.0	S	Biggin	4/02/2016	2.5	3067.0	2.0	1.0	0.0	156.0	79.00	1900.0	Yarra	-37.80790	144.99340	Northern Metropolitan	4
2	1465000.0	SP	Biggin	4/03/2017	2.5	3067.0	3.0	2.0	0.0	134.0	150.00	1900.0	Yarra	-37.80930	144.99440	Northern Metropolitan	4
4	1600000.0	VB	Nelson	4/06/2016	2.5	3067.0	3.0	1.0	2.0	120.0	142.00	2014.0	Yarra	-37.80720	144.99410	Northern Metropolitan	4
6	1876000.0	S	Nelson	7/05/2016	2.5	3067.0	4.0	2.0	0.0	245.0	210.00	1910.0	Yarra	-37.80240	144.99930	Northern Metropolitan	4
7	1636000.0	S	Nelson	8/10/2016	2.5	3067.0	2.0	1.0	2.0	256.0	107.00	1890.0	Yarra	-37.80600	144.99540	Northern Metropolitan	4
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
12205	601000.0	S	Ray	29/07/2017	35.5	3757.0	3.0	2.0	1.0	972.0	149.00	1996.0	Whittlesea	-37.51232	145.13282	Northern Victoria	2
12206	1050000.0	VB	Williams	29/07/2017	6.8	3016.0	3.0	1.0	0.0	179.0	115.00	1890.0	Hobsons Bay	-37.86558	144.90474	Western Metropolitan	6
12207	385000.0	SP	Williams	29/07/2017	6.8	3016.0	1.0	1.0	1.0	0.0	35.64	1967.0	Hobsons Bay	-37.85588	144.89936	Western Metropolitan	6
12209	560000.0	PI	hockingstuart	29/07/2017	4.6	3181.0	2.0	1.0	1.0	0.0	61.60	2012.0	Stonnington	-37.85581	144.99025	Southern Metropolitan	4
12212	2450000.0	VB	Village	29/07/2017	6.3	3013.0	6.0	3.0	2.0	1087.0	388.50	1920.0	Maribyrnong	-37.81038	144.89389	Western Metropolitan	6

```
missing_columns = subset_of_data.dropna(axis=1)
missing_columns
```



	Price	Method	SellerG	Date	Distance	Postcode	Bedroom2	Bathroom	Landsize	Lattitude	Longtitude	Regionname	Propertycount
0	1480000.0	S	Biggin	3/12/2016	2.5	3067.0	2.0	1.0	202.0	-37.79960	144.99840	Northern Metropolitan	4019.0
1	1035000.0	S	Biggin	4/02/2016	2.5	3067.0	2.0	1.0	156.0	-37.80790	144.99340	Northern Metropolitan	4019.0
2	1465000.0	SP	Biggin	4/03/2017	2.5	3067.0	3.0	2.0	134.0	-37.80930	144.99440	Northern Metropolitan	4019.0
3	850000.0	PI	Biggin	4/03/2017	2.5	3067.0	3.0	2.0	94.0	-37.79690	144.99690	Northern Metropolitan	4019.0
4	1600000.0	VB	Nelson	4/06/2016	2.5	3067.0	3.0	1.0	120.0	-37.80720	144.99410	Northern Metropolitan	4019.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...
13575	1245000.0	S	Barry	26/08/2017	16.7	3150.0	4.0	2.0	652.0	-37.90562	145.16761	South-Eastern Metropolitan	7392.0
13576	1031000.0	SP	Williams	26/08/2017	6.8	3016.0	3.0	2.0	333.0	-37.85927	144.87904	Western Metropolitan	6380.0
13577	1170000.0	S	Raine	26/08/2017	6.8	3016.0	3.0	2.0	436.0	-37.85274	144.88738	Western Metropolitan	6380.0
13578	2500000.0	PI	Sweeney	26/08/2017	6.8	3016.0	4.0	1.0	866.0	-37.85908	144.89299	Western Metropolitan	6380.0
13579	1285000.0	SP	Village	26/08/2017	6.3	3013.0	4.0	1.0	362.0	-37.81188	144.88449	Western Metropolitan	6543.0

13580 rows × 13 columns

```
filling_with_zero = subset_of_data.fillna(0)
```

filling_with_zero

	Price	Method	SellerG	Date	Distance	Postcode	Bedroom2	Bathroom	Car	Landsize	BuildingArea	YearBuilt	CouncilArea	Lattitude	Longtitude	Regionname	Propertycoun
0	1480000.0	S	Biggin	3/12/2016	2.5	3067.0	2.0	1.0	1.0	202.0	0.0	0.0	Yarra	-37.79960	144.99840	Northern Metropolitan	4019.
1	1035000.0	S	Biggin	4/02/2016	2.5	3067.0	2.0	1.0	0.0	156.0	79.0	1900.0	Yarra	-37.80790	144.99340	Northern Metropolitan	4019.
2	1465000.0	SP	Biggin	4/03/2017	2.5	3067.0	3.0	2.0	0.0	134.0	150.0	1900.0	Yarra	-37.80930	144.99440	Northern Metropolitan	4019.
3	850000.0	PI	Biggin	4/03/2017	2.5	3067.0	3.0	2.0	1.0	94.0	0.0	0.0	Yarra	-37.79690	144.99690	Northern Metropolitan	4019.
4	1600000.0	VB	Nelson	4/06/2016	2.5	3067.0	3.0	1.0	2.0	120.0	142.0	2014.0	Yarra	-37.80720	144.99410	Northern Metropolitan	4019.
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	.
13575	1245000.0	S	Barry	26/08/2017	16.7	3150.0	4.0	2.0	2.0	652.0	0.0	1981.0	0	-37.90562	145.16761	South-Eastern Metropolitan	7392.
13576	1031000.0	SP	Williams	26/08/2017	6.8	3016.0	3.0	2.0	2.0	333.0	133.0	1995.0	0	-37.85927	144.87904	Western Metropolitan	6380.
13577	1170000.0	S	Raine	26/08/2017	6.8	3016.0	3.0	2.0	4.0	436.0	0.0	1997.0	0	-37.85274	144.88738	Western Metropolitan	6380.
13578	2500000.0	PI	Sweeney	26/08/2017	6.8	3016.0	4.0	1.0	5.0	866.0	157.0	1920.0	0	-37.85908	144.89299	Western Metropolitan	6380.

filling_with_mean = subset_of_data['BuildingArea'].fillna(subset_of_data['BuildingArea'].mean())

filling_with_mean

```
<ipython-input-17-2705aeac5e44>:1: FutureWarning: Dropping of nuisance columns in DataFrame reductions (with 'numeric_only=None') is deprecated; in a future version this will raise Typ
  filling_with_median = subset_of_data.fillna(subset_of_data.median())
```

	Price	Method	SellerG	Date	Distance	Postcode	Bedroom2	Bathroom	Car	Landsize	BuildingArea	YearBuilt	CouncilArea	Lattitude	Longtitude	Regionname	Propertycoun
0	1480000.0	S	Biggin	3/12/2016	2.5	3067.0	2.0	1.0	1.0	202.0	126.0	1970.0	Yarra	-37.79960	144.99840	Northern Metropolitan	4019.
1	1035000.0	S	Biggin	4/02/2016	2.5	3067.0	2.0	1.0	0.0	156.0	79.0	1900.0	Yarra	-37.80790	144.99340	Northern Metropolitan	4019.
2	1465000.0	SP	Biggin	4/03/2017	2.5	3067.0	3.0	2.0	0.0	134.0	150.0	1900.0	Yarra	-37.80930	144.99440	Northern Metropolitan	4019.
3	850000.0	PI	Biggin	4/03/2017	2.5	3067.0	3.0	2.0	1.0	94.0	126.0	1970.0	Yarra	-37.79690	144.99690	Northern Metropolitan	4019.

filling_with_median = subset_of_data.fillna(subset_of_data.median())

filling_with_median

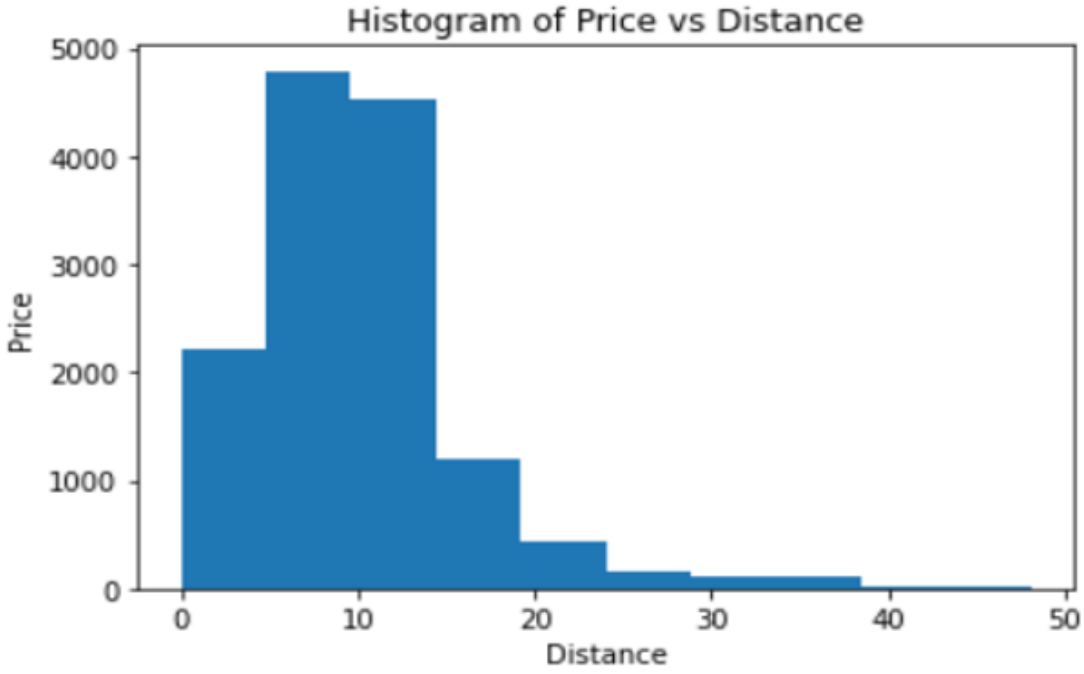
subset_of_data.describe()

```
subset_of_data.describe()
```

	Price	Distance	Postcode	Bedroom2	Bathroom	Car	Landsize	BuildingArea	YearBuilt	Lattitude	Longtitude	Propertycount
count	1.358000e+04	13580.000000	13580.000000	13580.000000	13580.000000	13518.000000	13580.000000	7130.000000	8205.000000	13580.000000	13580.000000	13580.000000
mean	1.075684e+06	10.137776	3105.301915	2.914728	1.534242	1.610075	558.416127	151.967650	1964.684217	-37.809203	144.995216	7454.417378
std	6.393107e+05	5.868725	90.676964	0.965921	0.691712	0.962634	3990.669241	541.014538	37.273762	0.079260	0.103916	4378.581772
min	8.500000e+04	0.000000	3000.000000	0.000000	0.000000	0.000000	0.000000	0.000000	1196.000000	-38.182550	144.431810	249.000000
25%	6.500000e+05	6.100000	3044.000000	2.000000	1.000000	1.000000	177.000000	93.000000	1940.000000	-37.856822	144.929600	4380.000000
50%	9.030000e+05	9.200000	3084.000000	3.000000	1.000000	2.000000	440.000000	126.000000	1970.000000	-37.802355	145.000100	6555.000000
75%	1.330000e+06	13.000000	3148.000000	3.000000	2.000000	2.000000	651.000000	174.000000	1999.000000	-37.756400	145.058305	10331.000000
max	9.000000e+06	48.100000	3977.000000	20.000000	8.000000	10.000000	433014.000000	44515.000000	2018.000000	-37.408530	145.526350	21650.000000

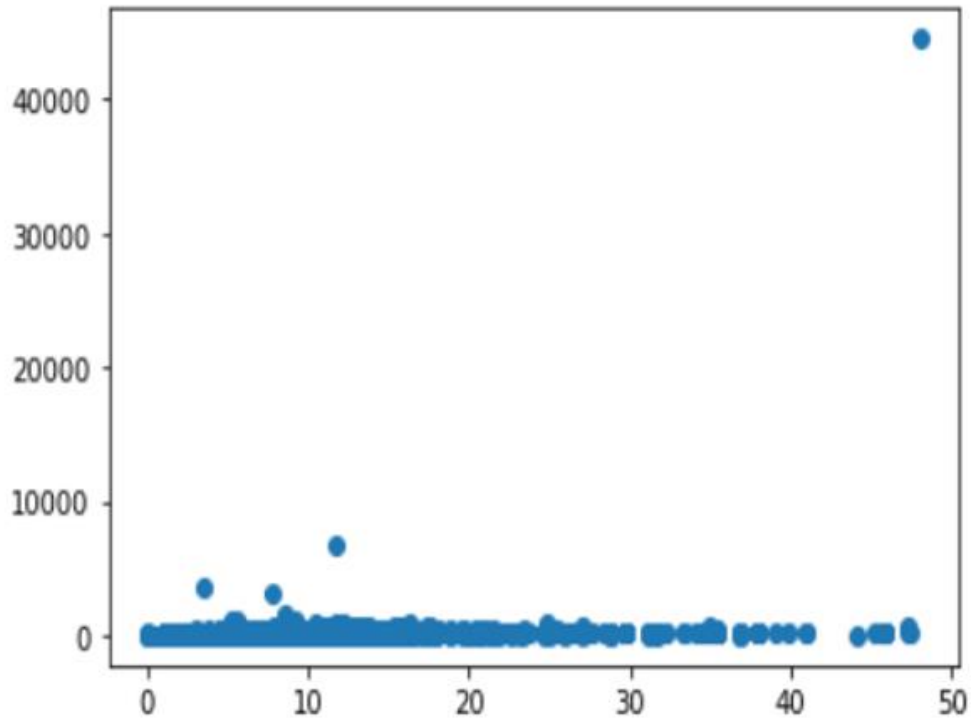
```
import matplotlib.pylot as plt
%matplotlib inline
plt.hist(data['Distance'])
plt.xlabel('Distance')
plt.ylabel('Price')
plt.title('Histogram of Price VS Distance')
```

```
Text(0.5, 1.0, 'Histogram of Price vs Distance')
```

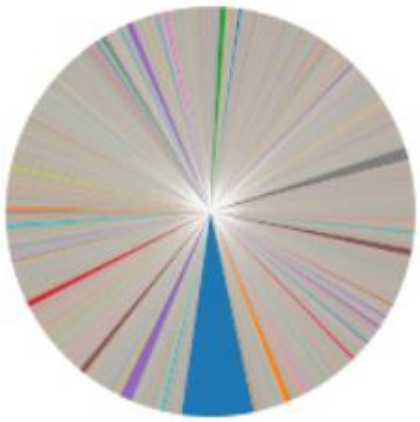



```
plt.bar(data['Distance'],data['BuildingArea'])
```

<matplotlib.collections.PathCollection at 0x7f96a180f220>




```
plt.scatter(data['DIstance'],data['BuildingArea'])
```



CONCLUSION: Thus, from this experiment we learnt about how datasets are managed and it's various operations using Pandas.

Lab Experiment – 12

AIM: Implement a web application using Django Framework and understand CRUD functionality.

THEORY: Django is an open-source web framework written in the Python programming language. Named after the jazz guitarist Django Reinhardt, it is used by some of the largest websites in the world including Instagram, Mozilla, and NASA, but also lightweight enough to be a popular choice for weekend side projects and startups. Its "batteries-included" approach means a powerful website can be generated quickly in the hands of a skilled developer.

A "web framework" is software that standardizes and abstracts away common difficulties and redundancies involved in making a website. For example, most websites need to connect to a database, deploy to a server, handle URL routing, security, user registration, generate templates, and so on. In the early days, programmers had to do all of this from scratch but they quickly recognized the commonalities and started creating web frameworks.

CODES:

- urls.py (project)

```
from django.contrib import admin
from django.urls import path
from django.urls import include
urlpatterns = [
   path('admin/', admin.site.urls),
   path('', include('base.urls')),
]
```

- views.py (app)

```
from django.shortcuts import render
def home(request):
      return render(request, "home.html")
```

```
def projects(request):
     return render(request, "todo.html")
def timetable(request):
     return render(request,"timetable.html")
def aboutus(request):
     return render(request, "aboutus.html")
```

- home.html

```
{% include 'navbar.html' %}
{% load static %}
<link rel="stylesheet" href="{% static 'styles/style.css' %}">
<title>Home</title>
<section class="tt-area">
    <center>
        <h2>Home Page</h2>
        <div class="box-align">
            <p>Lorem ipsum dolor sit amet consectetur adipisicing elit. Molestiae excepturi iusto animi architecto eligendi aliquam quia, corrupti earum reprehenderit fuga magni laboriosam dignissimos non dolorem recusandae exercitationem, ipsam, ullam voluptatibus.</p>
        </div>
    </center>
</section>
```

- navbar.html

```
{% load static %}
<link rel="stylesheet" href="{% static 'styles/style.css'
  %}">
<navbar>
  <ul>
  <li><a href="/">Home</a></li>
  <li><a href="/todo">TO-Do Tasks</a></li>
  <li><a href="/timetable">Time-Table</a></li>
   <li><a href= "/aboutus">About Us</a></li>
  </ul>
</navbar>
<hr>
```

- timetable.html

```
{% include 'navbar.html' %}
{% load static %}
<link rel="stylesheet" href="{% static 'styles/style.css'
  %}">
<title>Timetable</title>
    <section class="tt-area">
        <center>
            <h2>TimeTable</h2>
          <div class="box-align">
              <p>7:00 - 14:00 - College</p>
              <p>16:00 - 20:00 - Study</p>
              <p>20:00 - 21:00 - Walk</p>
              <p>22:00 - Sleep</p>
          </div>
```

```
            </center>
        </section>
```

- aboutus.html

```
{% include 'navbar.html' %}

{% load static %}

<link rel="stylesheet" href="{% static 'styles/style.css' %}">

<title>About Us</title>

<section class="tt-area">

    <center>

        <h2>Task Manager</h2>

        <div class="box-align">

            <p>Lorem ipsum dolor sit amet consectetur adipisicing elit. Molestiae excepturi iusto animi architecto eligendi aliquam quia, corrupti earum reprehenderit fuga magni laboriosam dignissimos non dolorem recusandae exercitationem, ipsam, ullam voluptatibus.</p>

        </div>

    </center>

</section>
```

- todo.html

```
{% include 'navbar.html' %}

{% load static %}

<link rel="stylesheet" href="{% static 'styles/style.css' %}">

<title>To-Do Tasks</title>
```

```
<section class="todo-area">
      <center>
            <h2>List of Tasks</h2>
           <div class="todo-item">
                <h2>DBMS</h2>
                <p>ESE Preparation</p>
                <p>Assignment 2</p>
           </div>
            <div class="todo-item">
                <h2>Python</h2>
                <p>Mini project</p>
            </div>
      </div>
      </center>
</section>
```

- style.css

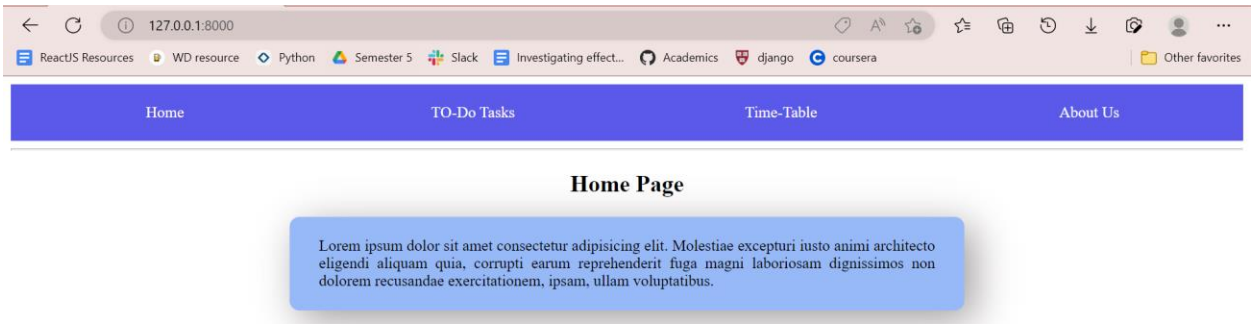
```
navbar {
      width: 100%;
      margin:0px 0;
      }
            navbar ul{
      width: 100%;
      list-style: none;
      margin: 0;
      padding: 0;
      overflow: hidden;
      }
```

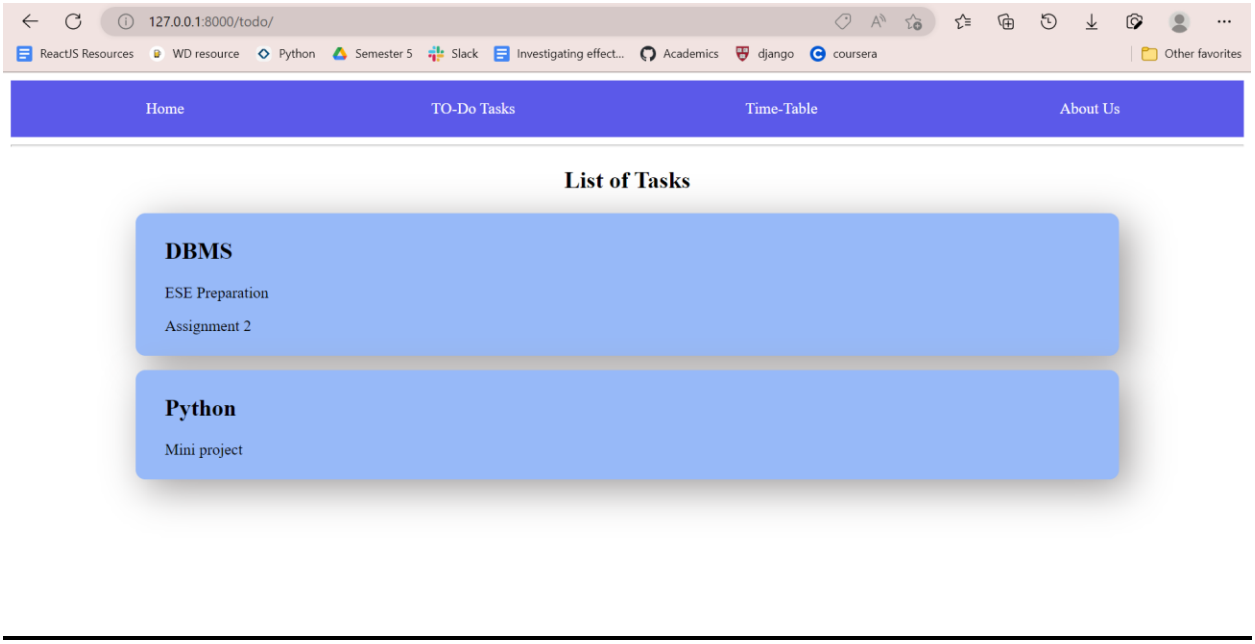
```
      navbar ul li{
      width: 25%;
      float: left;
      }
            navbar ul li a{
      text-decoration: none;
      color: rgb(255, 255, 255);
      text-align: center;
      padding: 20px 0;
      display: block;
      width: 100%;
      background-color: rgb(91, 89, 233);
      }

.home{
      margin-right: 5%;
      margin-top: 15px;
      margin-left: 15px;
      margin-bottom: 15px;
      padding-top: 15px;
      padding-left: 15px;
      padding-right: 5px;
      padding-bottom: 15px;
      border-radius: 10px;
      box-shadow: 15px 15px 15px rgb(157, 156, 234);
      text-align: justify;
      color: rgb(0, 0, 0);
```

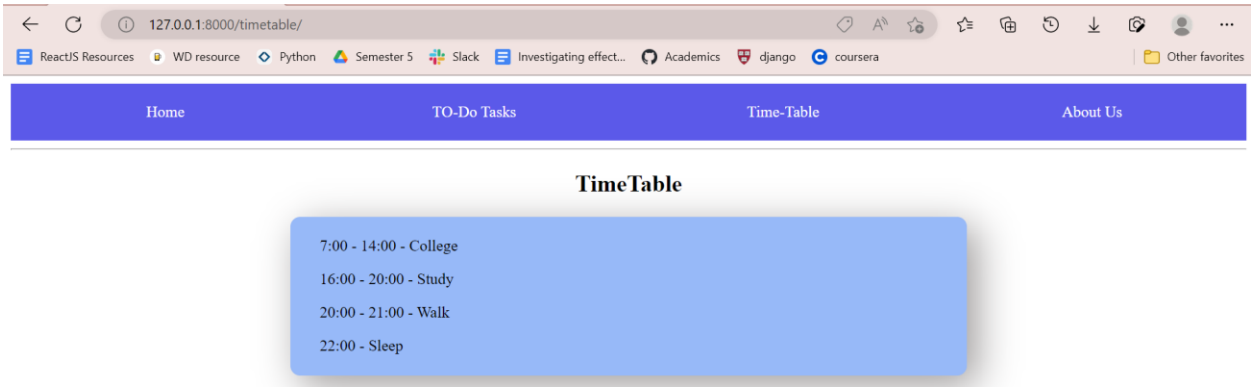
```
}

.todo-area {

      background-repeat: no-repeat;
      background-position: left;
      box-sizing: border-box;
}

.todo-item {
      width: 75%;
      margin-top:5px;
      margin-bottom:15px;
      margin-left: 5%;
      margin-right: 5%;
      padding-top:5px;
      padding-bottom:5px;
      padding-left: 30px;
      padding-right: 30px;
      border-radius: 10px;
      box-shadow: 10px 10px 40px rgb(158, 155, 155);
      text-align: justify;
      color: rgb(0, 0, 0);
      background-color:rgb(151, 185, 248);
}

.tt-area {
```

```
        background-repeat: no-repeat;
        background-position: left;
        box-sizing: border-box;
    }
    .box-align {
        width: 50%;
        margin-top:5px;
        margin-bottom:15px;
        margin-left: 5%;
        margin-right: 5%;
        padding-top:5px;
        padding-bottom:5px;
        padding-left: 30px;
        padding-right: 30px;
        border-radius: 10px;
        box-shadow: 10px 10px 40px rgb(158, 155, 155);
        text-align: justify;
        color: rgb(0, 0, 0);
        background-color:rgb(151, 185, 248);
    }
```

OUTPUT:


127.0.0.1:8000
ReactJS Resources
WD resource
Python
Semester 5
Slack
Investigating effect...
Academics
django
coursera
Other favorites
Home
TO-Do Tasks
Time-Table
About Us
Home Page
Lorem ipsum dolor sit amet consectetur adipisicing elit. Molestiae excepturi iusto animi architecto eligendi aliquam quia, corrupti earum reprehenderit fuga magni laboriosam dignissimos non dolorem recusandae exercitationem, ipsam, ullam voluptatibus.


127.0.0.1:8000/todo/
ReactJS Resources
WD resource
Python
Semester 5
Slack
Investigating effect...
Academics
django
coursera
Other favorites
Home
TO-Do Tasks
Time-Table
About Us
List of Tasks
DBMS
ESE Preparation
Assignment 2
Python
Mini project



CONCLUSION: Thus, from this experiment we learnt about how web pages function using Django and understood CRUD functionalities of Django.