ESO

PRACTICAL-01

Aim: Demonstrate the use of different file accessing modes, different attributes and Read() method.

Step 1: Create a file object using open method and use the write accessing mode follow up by writing some contents onto the file and then closing the file.

Step 2: Now open the file in read mode and then use read(), readline(), readlines(), and store the output in variable and finally display the contents of variable.

Step 3: Now use the file object for finding the name of file, the file mode in which it is opened whether the file is still open or close and finally, the output of the softspace attribute.

```
File obj = open ("abc.txt", "w") # File open (write mode)
File obj.write ("Computer science subjects" 021 + "\n")
File obj.write ("DBMS\n Python \n DS\n")
File obj.close ()

Fileobj = open ("abc.txt", "r")
str1 = Fileobj.read()
print ("The output of read method :", str1)
File obj.close ()
>>> The output of read method : Computer science
        Subject
        DBMS
        Python
        DS

# readline ()
  File obj = open ("abc.txt", "r")
  str2 = File obj.readline ()
  print ("The output of readline method: Computer
          Science subject.

# readlines
  File obj = open ("abc.txt", "r")
  str3 = Fileobj.readlines ()
  print ("The output of readlines method:", str3)
  File obj.close ()
>>> The output of readlines method: Computer
Science subject
        DBMS
        Python
        DS
```

```
#file attributes
a=fileobj.name
print("Name of file (name attribute):", a)
>>> (Name of file (name attribute', abc.txt)
b= fileobj.closed
print ("(close) attribute:", b)
>>> (close) attribute = True.
c= fileobj.mode
print ("filemode", c)
>>> ('filemode', 'r')
d=fileobj.softspace
print ("softspace", d)
>>> ("softspace:", 0
# w+ mode
Fileobj=open("abc.txt", "w+")
Fileobj.write("sourabh")
Fileobject.close()

# r+ mode
Fileobj = open("abc.txt","r+")
str1 = Fileobj.read(r)
print ("output of r+", str1)
Fileobj.close()
>>> ('output of r+','saurabh')

# write mode
Fileobj = open("abc.txt","w")
File obj.write("DBMS")
Fileobj.close()

# read mode
File obj = open("abc.txt","r")
str2=File obj.read()
print ("output of read mode 1", str2)
>>>('output of read mode', 'saurabh')
```

Step 4: Now open the fileobject in write mode. Write some another content close subsequently, then again open the fileobject in 'w+' mode that is the update mode and write contents.

Step 5: Open fileobject in read mode, display the updated written contents and close, open again in 'a+' mode with parameters passed and display the output subsequently.

Step 6: Now open fileobject in append mode, open write method, write content close the fileobject again. Open the fileobject in read mode and display the append output.

Step 7: Open the file object in read mode, declare a variable and perform file object · tell method and store the output consequently in variable.

Step 8: Use the seek method with the arguments with opening the file object in read mode and closing subsequently.

Step 9: Open file object with read mode also use the readlines method and store the output consequently in and print the same for counting the length. Use the for condition statement and display the length.

```
#append mode
File obj=open("abc.txt", "a")
File obj. write("Data structure")
File close()
Fileobject = open("abc.txt", "r")
Str3 = File obj. read()
print("output of appendmode", str3)
Fileobj. close()
```

>>> output of append mode : "Saurabh", "Data Structure"]

```
#tell()
Fileobj = open("abc.txt", "r")
POS = File obj. tell()
print("Tell():", POS)
File object. close()
```

>>> ("tell():", POS)

```
#seek()
File obj=open("abc.txt", "r")
Str4 = File obj. seek(0,0)
Str8 = Fileobj. read(10)
Print("The beginning of the line is =", str8)
```

#CODE :

```
class odd:
    def __iter__(self):
        self.num=1
        return self
    def next(self):
        if self.num<=10:
            num=self.num
            self.num+=2
            return num
        else:
            raise StopIteration.
```

```
>>> y=count()
>>> z=iter(y)
>>> z.next()
1
>>> z.next()
3
>>> z.next()
5
>>> z.next()
7
>>> z.next()
9
>>> z.next()
11
```

PRACTICAL-02

- **Aim:** Demonstrate the use of iteration and iterators.

- **Theory:** In python, iterator is an object which implements iterator class which has 2 methods namely __iter()__ and __next()__.
  List, tuple, dictionary & the set all represents an iterable object.

Q1. Write a program using iterable objects for displaying the odd numbers in range 1–10.

Algorithm:

S1: Define a iter() with argument and initialize the value and return that value.

S2: Define the next() with an argument and compare the upper limit by using a conditional statement.

S3: Now create an object of the given class and pass this object in the iter method.

Q2. Write a program using an iterator for calculating the power of a given number. For instance, number entered is 2 then value calculated should be $1, 2^1, 2^2, 2^3, 2^4$.

Algorithm:

S1: Define iter() with argument and initialize value and return the value.

S2: Now define next() with an argument and compare the upper limit by using conditional statement.

S3: Now create an object of the given class & pass this object in the iter method.

#CODE :

028

```
class power:
    def __iter__(self):
        self.p=0
        return self
    def next(self):
        if self.p<=10:
            num=self.p
            self.p+=1
            p0=2**num
            print("2**",self.p-1,"=",p0)
            return p0
        else:
            raise StopIteration.
```

```
>>> p=power()
>>> x=iter(p)
>>> x.next()
>>> 2**a=21
>>> x.next()
>>> 2**1=4
>>> x.next()
>>> 2**3=8
```

CODE:

```
class fact:
    def __iter__(self):
        self.f = 1
        return self
    def next(self):
        if self.f <= 10:
            num = self.f
            self.f += 1
            fact = 1
            for i in range(1, num+1):
                fact = fact * i
            print(self.f - 1, "!=", fact)
        else:
            raise StopIteration
```

```
>>> f = fact()
>>> x = iter(f)
>>> x.next()
1 != 1
>>> x.next()
2 != 2
>>> x.next()
3 != 6
```

Q3. Write a program using iterable concept to find factorial of number in range 1-10 :

. Algorithm :

S1: Define iter() with argument & initialize the value and return the value.

S2: Define next() with an argument and compare the upper limit by using a conditional statement.

S3: Now create an object of the given class & pass the object in iter method.

Q4. Write a program using iterable concept to display multiples of 2 in range 1 to 10.

. Algorithm :

S1: Define iter() with argument & initialize the value and return the value.

S2: Define the next() with an argument and compare the upper limit by using a conditional statement.

S3: Now create an object of the given class & pass this object in iter method.

CODE:

030

```
class mult :
    def __iter__(self):
        self.m = 1
        return self
    def next(self):
        if self.m <= 10:
            num = self.m
            self.m += 1
            table = 2 * num
            print("2 *", num, "=",
                  table)
        else:
            raise StopIteration
```

```
>>> m = mult()
>>> n = iter(m)
>>> n.next()
2 * 1 = 2
>>> n.next()
2 * 3 = 6
>>> n.next()
2 * 4 = 8
```

080

#CODE:

```
def accept_age():
    age = int(input("enter your age"))
    if age > 30 or age < 16:
        raise value error
    else:
        print("your age is", age)
valid = False
while not valid:
    try:
        age = accept_age()
        valid = True
    except ValueError:
        print("your age is not a range
```

>>> Enter your age : 15
your age is not in range.
>>> enter your age : 32
Your age is not in range.
>>> enter your age : 17
your age is 17.

PRACTICAL-03

Aim: Demonstrate the use of exception Handling.

Theory: An exception is an event which occurs during execution of program which disrupts the normal flow of program. Thus an exception represents object which represents an error.

Q1. Write a program to check the range of the age of the students in the given class & its age does not fall in given range use value error exception otherwise return the valid no.

Algorithm:

S1: Define a function which accepts a variable as age of a student from standard input.

S2: Use if conditional statement to check whether the input age falls in range & to return the age else use value error exception.

S3: Define the while loop to check whether the Boolean expression holds there use of try block to accept the age of student & terminate the looping condition.

S4: Use except method w/ valueerror & print the message 'not a valid range'

Q2. Write a program to check whether the number is given class & if the number is a floating point use value error as exception for the given input.

Algorithm:

S1: Use the try block & accept the input using input() & convert it into integer datatype and subsequently terminate the block.

#CODE:

```
while True:
    try:
        a = int(input("Enter a number:"))
        print("Valid number")
        break
    except ValueError:
        print("Not a Valid number! try again")
```

>>> Enter a number : 17.2
Not a valid number! Try again:
Enter a number : 17
Valid number.

CODE:

```
def divide (a,b):
    ans=a/b
    return ans.
While True:
    try:
        a = int ("input ("Enter first number:")
        b = int (input ("Enter second number:")
        ans = divide (a,b)
        print ("division of ", a, " and ", b, "is", ans
    except ZeroDivisionError:
        print ("Error!")
```

```
>>> Enter first number: 1
>>> Enter second number: 1
    Division of 1 and 1 is 1
>>> Enter first number: 1
>>> Enter second number: 0
    Error!
```

Step 2: Use the except block with exception as valueerror & display appropriate message is suspicious code is part of try block.

Q3. Write a program to demonstrate use of zero division error.

Algorithm:

S1: Use the try block & accept the input using input() & then convert it into integer datatype.

S2: Define a function with 2 parameter to divide the nos. given by user.

S3: Define while loop to check whether the boolean expression holds true.

S4: Use except with zero division error & print the message.

mm
19/12/19

880

Practical-04

Aim: Demonstrate the use of regular expression.

Theory: Regular expression represents the segment of characters which is mainly used for finding & replacing the given pattern in a string, for this we import re-module and common usage of regular expression involves following functionalities:

- searching a given string.
- Finding it.
- Breaking it into smaller substrings.
- Replacing part of it.

Q1. Write a regular expression ~~regress~~ seggregat numeric and alphabetic values from given string.

Algorithm:

1: Now apply string & pattern in findall() and display the output.

#1 code:

```
import re
string = "roll no 1234 abc4567"
result = re.findall ("\d +", string)
result = re.findall ("\D +", string)
print (result)
print (result)
```

#output

```
>>> ['1234', '4567']
>>>['hello', 'abc']
```

#1 2 code :

```
import re
string = "Python is an important language"
result = re.search ("^Python", string)
print (result)
if result:
    print ("Match found")
else:
    print ("Match not found")
```

```
>>> <re.match object : span = (0, 6)
                match = "Python">
>>> match found.
```

S2: Id is used for matching all decimal digits whereas D is used to match non-decimal digits.

Q2. Write a regular expression for finding the match string at beginning of given sequence.

Algorithm:

S1: Import re-module & apply a string.

S2: Use search() with "\APython" and string as two parameters.

S3: Now display the output ~~stmt if~~.

S4: Use if stmt for user to know whether the match is found or not.

Q3. Write a regular expression to ~~change~~ check whether the given mobile number starts with 8 or 9 & the total length of digit should be atmost 10.

Algorithm:

S1: Import re-module and apply a string of module no. 8.

S2: Now use for conditional stmt. to find if the number starts with 8 or 9 and total no. should length of 10. Use match() inside for stmt to find match in given string.

S3: Use if conditional stmt to know whether we have a match or not. If we have use group() to display the output and if we don't display incorrect mobile no.

#3 code:

```
import re
li=[ "9876543210", "8765432109",
     "7654340987", "6543210987"]

for element in li:
    result = re.match("[8-9][0-9]{9}", element)

    if results :
        print ("correct mobile no")
        print ((result.group(1));

    else :
        print ("Incorrect mobile no.")
```

#output:

```
>>> correct mobile no:
    9876543210
    correct mobile no.
    8765432109
    Incorrect mobile no.
    Incorrect mobile no.
```

A code:

```
import re
string = "Python is Important".
result1 = re.findall ("\W+", string)
result 2 = re.findall ("\w+", string)
print (result1)
print (result2)
```

output:

```
>>> ["'Python', ' ', 'is', 'important,"]
['Python', 'is', 'important'].
```

Q4. Write a regular expression for extracting a word from given string along with space character in between the word and subsequently extracts the word without space character.

Algorithm:

S1: Import re-module and apply a string.

S2: Use findall() to extract a word from given string.

S3: Use "\w *" to extract word along with space & use "\w+" to extract word without space.

S4: Now display the output.

Q5. Write a regular expression for extracting first and last word for a string.

Algorithm:

S1: Import re module and apply a string.

S2: Use findall() in use "\w+" as onto parameter to find first word of string.

S3: Now display the result.

Q6. Write a regular expression for extracting the date in format dd-mm-yyyy by using the findall() format Amit 201 24-12-2019

Algorithm:

S1: Import re-module and apply string.

S2: Use findall method and use '\d{2}-\d{4}' as an parameter.

S3: Now display the output

5 code:


```
import re
string = "Python is important"
result = re.findall("^\w+", string)
result = re.findall("\w+$", string)
print(result)
print(result1)
```

#output:

```
>>> ['Python']
>>> ['Importance']
```

6 code:

```
import re
string = "Amit 201 24-12-2019"
result = regindall("\d{2}-\d{2}-\d{4}", string)
print(result)
```

#output:

```
>>> ['24-12-2019']
```

#code 7:

```
import re
string = "abc@tcsc.edu"
result1 = re.findall("^\w+", string)
result2 = re.findall("t\w+.\w+$",
                     string)
result3 = re.findall("[\w\.-]+",
                     string)
print(result1)
print(result2)
print(result3)
```

#output:

```
>>> ['abc']
>>> ['tcsc.edu']
>>> ['abc', 'tcsc.edu']
```

Q7. Write a re for extracting the

(1) username from email id.
(2) hostname from email id.
(3) Both username & hostname from email id.

Algorithm:

S1: Import re module and apply a string.

S2: Use findall () to find username, hostname & both of email id.

S3: Use "\w+" for username. Use "\w+.\w+$" for hostname. Use "[\w.-]+" for both as parameter in findall().

S4: Display output.

09/01/2024

CSO PRACTICAL 05

GUI

- **Aim:** To demonstrate GUI controls.

- **Algorithm:** P1

S1: Use the tkinter library for importing the features of the tent widget.

S2: Create a variable from the tent method & position it on parent window.

S3: Use the pack() along with the object created from the tent() and use the parameter.
(1) side = ~~LEFT~~ T, padx = 20
(2) side = LEFT, pady = 30
(3) side = TOP, padx = 40
(4) side = TOP, pady = 50

S4: Use the mainloop() for triggering of the events.

S5: Now repeat above steps with Label() which takes the following arguments.
(1) Name of parent window.
(2) Text attribute which defines string.
(3) Background (bg) colour.
(4) foreground (fg) and then use pack() with padding.

Program 1 : Label Widget :

```
root = TK()
l1 = Label(root, text = "Python")
l1.pack(padx = 20, pady = 50, side = TOP)
l2 = Label(root, text = "GUI").pack(ipadx = 50, ipady = 80, side
                                    = BOTTOM)
t1 = Text(root)
quote = "Python is interpreted language."
t1.insert(END, quote)
t1.pack(padx = 70, pady = 90, side = RIGHT)
root.mainloop()
```

Program 2 : Radio button widget :

```
def sel():
    selection = "You selected the option " + str(var.get())
    label.config(text = selection, justify = LEFT)
root = TK()
var = IntVar()
r1 = Radiobutton(root, text = "option no.1", variable = var, value = 1,
command = sel).pack(anchor = W)
label = Label(root).pack()
root.mainloop()
```

Program 3: Scrollbar:

```
root=TK()
paragraph ="The mainloop is the GUI based application
make the given widget available")
s=Scrollbar(root).pack(side = RIGHT, fill =Y)
t=Text(root,height =10, width = 20).pack(side =RIGHT, fill=
s.config(command =t.yview)
t.config( yscrollcommand =s.set)
t.insert (END, paragraph).
root.mainloop()
```

Program 4 : Frame widget :

```
root=TK()
frame =Frame(root).pack (padx =20, pady = 50, side =TOP)
leftframe = Frame(root).pack( side =LEFT)
rightframe =Frame(root).pack(side =RIGHT)
buttonpush=Button( frame, text="PUSH", activebg="red").
pack (side =BOTTOM)
buttonremove =Button(frame, text="REMOVE").pack(side=
buttonadd = Button(rightframe, text ="Add").pack(side=R
buttonmodify =Button(leftframe, text="Modify").
pack (side =RIGHT)
root.mainloop()
```

Algorithm: P2

S1: Import methods from tkinter library.

S2: Use parent window object with geometry() declaring specific pixel size of parent window.

S3: Now, define a function which tells user about given selection mode from multiple option available.

S4: Now define the parent window and define option available with string variable.

S5: Use the radiobutton widget and place it onto parent window specifying text, variable, value and command attribute.

S6: Now, use pack() to specify anchor attribute.

S7: Use Label widget and place it onto parent window and use pack() method.

S8: Use mainloop() to trigger corresponding events.

IAO PRACTICAL-06

(A)

· Aim: To demonstrate human face with program

· Algorithm:

S1: Import relevant methods from graphics library.
S2: Define the function mainprg.
S3: Assign point with circle attribute to display head.
S4: To display eyes, assign two different points and use attributes set and fill for it.
S5: To display mouth on to the screen, use oval attribute and specify the points.
S6: Use label widget and specify text and points.
S7: Use message to specify text and the point at which it should be displayed.
S8: Finally, use win.close to close the program.
S9: Terminate program using mainprg().

Program :

```
from graphics import *
def mainprog():
win = GraphWin('Face', 200, 150)
head = Circle(Point(40,100),25)
head.setFill("yellow")
head.draw(win)
eye1 = Circle(Point(32,90),5)
eye1.setFill('blue')
eye1.draw(win)
eye2 = Circle(Point(50,90),5)
eye2.setFill('blue')
eye2.draw(win)
mouth = Oval(Point(30,105), Point(50,110))
mouth.setFill("red")
mouth.draw(win)
label = Text(Point(100,120), 'A face')
label.draw(win)
message = Text(Point(win.getWidth()/2,20), 'Click to quit.')
message.draw(win)
win.getMouse()
win.close()
mainprg().
```

SAO

Program (B):

```
from tkinter import *
window = TK()
farenheit = DoubleVar()
farenheit.set(32.0)
def convert-(celsius):
farenheit.set((9.0/5.0)* celsius +32)
lc=Label (window, text = "Temperature in celsius :")
lc.grid (row=0, column=0)
celsius =IntVar()
e= Entry (window, textvariable = celsius)
e.grid (row = 0, column=1)
b = Button (window, text = 'Convert', command = lambda:
convert (celsius.get()))
b.grid (row=1, column = 0, columnspan = 2)
lf= Label (window, textvariable = farenheit)
lf.grid (row = 2, column = 0, columnspan = 2)
window.mainloop ().
```

Algorithm:

S1: Import relevant methods from tkinter library.

S2: Make a parent window and define a function for conversion of one scale to another.

S3: Define farenheit and celsius and give them variables which hold value.

S4: Use label widget and place it on parent window, also use text to specify the type of conversion.

: Use grid() method and specify row & column attributes.

Use Entry widget and use textvariable following the grid method with specified row and columns.

Use button widget and use command attribute along with text following the grid method specifying row, column & columnspan attributes.

Use mainloop() method at last to terminate the program.

EAD

PRACTICAL-07

(A)

• Aim: Addition/Subtraction/Multiplication of 2 numbers using program.

• Algorithm:

S1: Import relevant methods from tkinter library.

S2: Now define a function to add, multiply and subtract integers seperately and the result in general.

S3: Create a parent window object.

S4: Define the text variable and assign it to string variable.

S5: Create label widget and place it on parent window using text as 'first', 'second', 'result' in three respective Label widgets for all 3 operators, i.e; Addition, Subtraction and Multiplicati

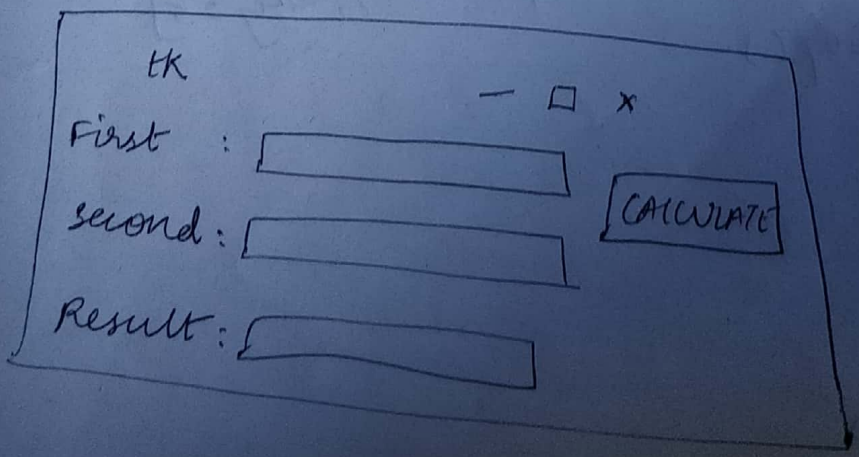
S6: Use grid() method to position it on parent windo specifying attributes row, column sticky for all 3 label widgets for 3 times.

· program :

041

```
from tkinter import *
def addNumbers():
    res = int(e1.get()) + int(e2.get())
    myText.set(res)
master = Tk()
myText = StringVar();
Label(master, text="First").grid(row=0, sticky=W)
Label(master, text="second").grid(row=1, sticky=W)
Label(master, text="Result").grid(row=3, sticky=W)
result = Label(master, text="", textvariable=myText).grid
(row=3, column=1, sticky=W)
e1 = Entry(master).grid(row=0, column=1)
e2 = Entry(master).grid(row=1, column=1)
b = Button(master, text="Calculate", command=addNumbers)
b.grid(row=0, column=2, columnspan=2, rowspan=2,
sticky=W+E+N+S, padx=5, pady=5)
mainloop().
```

· Output :



(B2) Program:

```
from tkinter import *
def subNumbers():
    res = int(e1.get()) - int(e2.get())
    myText.set(res)
master = Tk()
myText = StringVar();
Label(master, text = "First").grid(row = 0, sticky = W)
Label(master, text = "second").grid(row = 1, sticky = W)
Label(master, text = "Result").grid(row = 3, sticky = W)
result = Label(master, text = "", textvariable = myText).
grid(row = 3, column = 1, sticky = W)
e1 = Entry(master).grid(row = 0, column = 1)
e2 = Entry(master).grid(row = 1, column 1)
b = Button(master, text = "Calculate", command = subNum
b.grid(row = 0, column = 2, columnspan = 2, rowspan = 2,
sticky = W + E + N + S, padx = 5, pady = 5)
mainloop().
```

57: Use Label widget and place it onto parent window using text as 'result' for displaying the output / result on screen.

58: Use grid() method to position parent window specifying attributes row, column, sticky.

59: Use two Entry widgets and place it onto the parent window.

Program :

```
from tkinter import *
def multiplyNumbers():
    res = int(e1.get()) * int(e2.get())
    myText.set(res)
master = Tk()
myText = StringVar();
Label(master, text="First").grid(row=0, sticky=W)
Label(master, text="Second").grid(row=1, sticky=W)
Label(master, text="Result").grid(row=2, textvariable=myText).
grid(row=3, column=1, sticky=W)
e1 = Entry(master).grid(row=0, column=1)
e2 = Entry(master).grid(row=1, column=1)
b = Button(master, text="Calculate", command=Multiply Numbers)
b.grid(row=0, column=2, columnspan=2, rowspan=2,
sticky=N+S+W+E, padx=5, pady=5)
mainloop()
```

*Program :

Socket_server.py

```
import socket
def server_program():
host = socket.gethostname()
port = 5000
server_socket = socket.socket()
server_socket.bind((host, port))
server_socket.listen
server_socket.listen(2)
conn, address = server_socket.accept()
print("Connection from :" + str(address))
while True:
    data = conn.recv(1024).decode()
    if not data:
        break
    print("from connected user :" + str(data))
    data = input('->')
    conn.send(data.encode())
    conn.close()
    Server_program()
```

PRACTICAL - 08

Algorithm:

S1: Import socket library for relevant methods in it.

S2: Define server program and assign hostname and to the socket.

S3: Using bind method and specify host and port.

S4: Use accept method after listen(), to obtain conn. address from the socket server.

S5: Use print statement to print string address.

S6: Using while loop, assign data = conn.recv(1024). decode() . The alternative statement ~~being~~ break, is after print statement for connected user.

S7: Using send() to send encoded data in database.

S8: Using conn.close() to close server connectivity.

240

- **Algorithm:**

S1: Import socket to use relevant methods from the library.

S2: Define function client program and assign hostname to the socket. Show port = 5000.

S3: Use connect method on client socket specifying host and port.

S4: Insert a message "->" and another in lower strip → "bye".

S5: Use send() method to send encoded message.

S6: Assign data to client socket to recieve decoded message.

S7: Use print statement to show recieved data from server.

S8: close the server connectivity.

socket - client.py

```
import socket
def client_program():
host = socket.gethostname()
port = 5000
client_socket = socket.socket()
client_socket.connect((host, port))
message = input("->")
while message.lower().strip() != "bye":
client_socket.send(message.encode())
data = client_socket.recv(1024).decode()
print('Received from server:' + data)
message = input("->")
client_socket.close()
client_program().
```

8A0

* CODE :

```
import sqlite3
conn = sqlite3.connect ("student.db")
cur.conn.cursor()
cur.execute ('create table abc (Name text, Rollno
① <sqlite 3. cursor object at 0x02DCF4E0 >
('Insert into abc VALUES ("Nidhi", 3)')
② 'Insert into abc value ("Nidhi", 3)'}
conn.commit ()
cur.fetchall()
③ []
cur.execute ('select name from abc')
④ < sqlite 3. Cursor object at 0x02DCF4E0 >
('Insert into abc value ("Jazz", 4)')
⑤ 'Insert into abc value ("Jazz", 4)'
. ("Insert into abc value ("Sam", 6)')
⑥ 'Insert into abc value ("Sam", 6)'
cur.execute ('select name from abc')
⑦ < sqlite3. Cursor object at 0x02DCF4E0>
con.commit()
cur execute ('select Name from abc')
⑧ <sqlite 3. Cursor object at 0x02DCF4E0>
cur. fetchall()
⑨ []
('update into abc value ("Harry", 5)')
⑩ 'update into abc value ("Harry", 5).
```

PROGRAM-09

SQLITE3

Algorithm:

S1: Import sqlite the relevant library for the database connectivity & operating system functionality.

S2: Now, make an object for making connection to the given database.

S3: Further create an object corresponding to the cursor area for execution of the different query statement.

S4: Use the cursor object so created for implementing the structure of database & values within the database.

S5: Use execute () method for implementation of the select clause for entering the values.

S6: Now use the fetchall method also with the cursor object for display the value on to the screen.

CAO

PRACTICAL-07

(B) Factorial :

- **Aim:** WAP to find factorial of number & use arithmetic operations ~~of~~ on two numbers using GUI.

- **Algorithm:**

S1: Import relevant methods from tkinter library.

S~~4~~: Now create an object with entry widget and use pack() for positioning on parent window.

S2: Now define a function factorial to calculate factorial using recursive function.

S3: Define another function calculate to call factorial function.

~~S5~~: Now create ~~and object~~ with button widget ~~along with~~ command() ~~along with~~

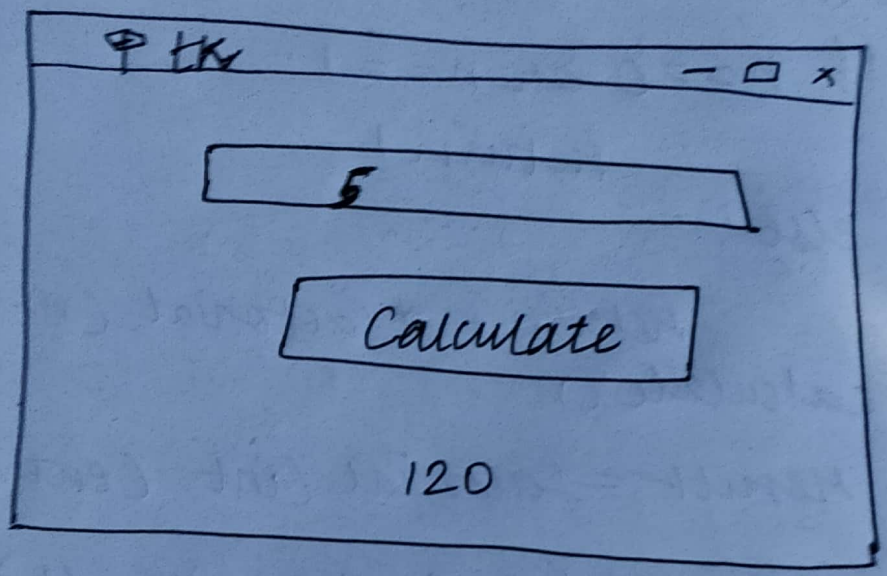
S5: Now create object with button widget along with command attributes to calculate factorial

*CODE:

```
from tkinter import *
def factorial (n):
    if n==0 or n==1:
        return 1
    else:
        return n*factorial (n-1)
def calculate ():
    result=factorial (int (entry_tent.get ()))
    info.config (text = result)
root =TK()
entry test = Entry(root)
entry test.pack()
Btn =Button(root, text="calculate",
        command = calculate)
Btn.pack()
info = Label (root, tent = "factorial")
info.pack()
root.mainloop().
```

030

#OUTPUT :



6: Now again create an object with label() to show output.

7: finally use the mainloop().