```
file.obj=open("abc.txt","w") # File open(write mode)
File.obj=write("computer science subjects" + "\n")
File.obj.write("DBMS \n Python \n DS\n")
File obj.close()
File obj = open("abc.txt","r")
str / = File obj read()
print ("The output of read method :",str\)
fileobj.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 :"str2)
file obj.close()
```

>>> The output of readline method: Computer Science

```
# readlines
file obj = open("abc.txt","r")
str3 = file obj.readlines()
print ("The output of readlines method :","str3)
file obj.close()
```

>>> The output of readlines method: Computer Science
Subject
DBMS
Python
DS

PRACTICAL : 01

* **Objective** : Demonstrate the use of different file accessing mode, different attributes and read method.

Step 1 : Create a file object using open method and use the write accessing mode followed 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() 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.

Step 4 : Now open the file object in write mode. Write some another content close subsequently then again open the file object in 'wt' mode that is the update mode and write contents.

Step 5 : Open file object in lead mode, display the updated written contents and close. Again open the file object in 'r+' mode with parameters passed & display the output subsequently.

Step 6 : Now open file object in append mode open write method. Write content close the file object again. Open the file object in read mode & display the append output.

```
# file attributes
a = file obj.name
print("Name of file(name attribute):", a)
>>> (Name of file(name of attribute, abc.txt)
b = file obj.closed
print((close) attribute = ", b)
>>>(close) attribute = ; True
c = file obj. mode
print("file mode," c)
>>> ("file mode", 'r')
d = file obj. softspace
print ("soft space", d)
>>> ("softspace :", 0)
```

```
# w+ mode
file obj = open("abc.txt", "w+")
file obj write ("saurabh")
file object.close()
```

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

```
# r+ mode
file obj = open("abc.txt", "r+")
str1 = file obj. read (v)
print ("output of r+", str1)
file.object . close()
>>> ('output of r+', 'saurabh')
```

```
# read mode
file obj = open ("abc.txt", "r")
str2 = file obj.read()
print ("output of read mode)",
        str 2)
>>>('output & read mode !,
'saurabh')
```

```
#append mode
file obj '. open("abc.txt", "a")
fileobj.write("Data structure")
file.close()
file object = open("abc.txt", "r")
str 3 = file obj. read()
print("output of append mode", str 3)
fileobj. close()
>>>("output of append mode:", "Saurabh", 'Data structure")

# tell()
file obj = open("abc.txt", "r")
Pos = fileobj. tell()
print("tell():", Pos)
file object. close()
>>> ('tell():', Pos)

# seek()
file obj = open("abc.txt", "r")
str 4 = file obj. seek(0,0)
str 8 = file obj. read(1,0)
print(The beginning of the line is =", str 8)
```


Step 7 : Open the file object in read mode declare a variable & 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 & closing subsequently.

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

PRACTICAL 2 :

Aim : Demonstrate the use of iteration & iterators.

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

Q.1] Write a program using iterable objects for displaying the odd numbers in range 1 to 10.

Algorithm :

Step 1 : Define a iter() with argument and initialize the value and return that value.

Step 2 : Define the next() with an argument and compare the upper limit by using a condition statement.

Step 3 : Now create an object of the given class and pass this object in the iter method.


```
# 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
```

```
# code :
class Power:
    def __iter__(self):
        self.p = 0
        return self
    def next(self):
        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()
>>> n = iter(p)
>>> n.next()
    2 ** 0 = 1
>>> n.next()
    2 ** 2 = 2
>>> n.next()
    2 ** 2 = 4
>>> n.next()
    2 ** 3 = 8
```

g.2] Write a program using an iterator for calculating the power of a given no. for instance number entered is 2 then value calculated should be $1, 2^1, 2^2, 2^3, 2^4$.

Algorithm :

Step 1 : Define iter() with argument and initialize value and return the value.

Step 2 : Now define next() with an argument and compare the upper limit by using conditional statement.

Step 3 : Now create an object of the given class & pass this object in the iter method.

Q3] Write a program using iterable concept to find factorial of no. in range 1 to 10.

Algorithm:

Step 1: Define a iter() with argument & initialize the value and return the value.

Step 2: Define the next() with an argument and compare the upper limit by using a conditional statement.

Step 3: Now create an object of the given class & pass this object in the iter method.


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

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

#code:

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

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


Q.4] Write a program using iterable concept to display multiple of 2 in range 1 to 10:

Algorithm:

Step 1: Define a iter() with argument & initialize the value and return the value.

Step 2: Define the next() with an argument & compare the upper limit by using a conditional statement.

Step 3: Now create an object of the given class & pass this object in the iter method.

PRACTICAL : 3

Aim : Demonstrating the use of exception handling

Theory : An exception is an event which occurs during execution of program which disrupt the normal flow of program. Thus an exception represents object which represents an error. This object is derived from given class & when the python script raises an exception, it must be handled immediately. Otherwise it will terminate and close the program.

Q:1 Write a program to check the range of the age of the students in given class and if age does not fall in given range. Use value error exception otherwise return the valid no.

Algorithm :

Step 1 : Define a function which will accept the age of the student from standard input.

Step 2 : Use if conditional to check whether the input age falls in range & so return the age else use value error exception.

Step 3 : Define the while loop to check whether the boolean expression holds true. Use the try block to accept the age of student and terminate the looping condition.

code :

```
def accept_age():
    age = int(input("Enter your age:"))
    if age > 30 or age < 16:
        raise ValueError:
    else:
        print("Your age is ", age)
valid = False
while not valid:
    try:
        age = accept_age()
        valid = True
    except ValueError:
        print("Your age is not in 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.
```

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: 12
valid number.
```


Step 4: Use except with ValueError and print the message, not a valid range.

q.2] Write a program to check whether the number in given class and if the number is a floating point use value error as exception for the given input.

Algorithm:

Step 1: Use try block & accept the input using input() & convert it into integer datatype and subsequently terminate the block.

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

q.3]

Q.3] Write a program to demonstrate use of zerodivision Error.

Algorithm:

Step 1: Use the try block and accept the input using input() & then convert it into integer datatype.

Step 2: Define a function with 2 parameter to divide the number given by user.

Step 3: Define while loop to check whether the boolean expression holds true.

Step 4: Use except with zerodivision error & print the message.

#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)
        break
    except ZeroDivisionError:
        print("Error!")
```

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

24/12/19

CODE 1

```
import re
String = "All might 1 2 34"
result = re.findall ("\d+", string)
result = re.findall ("\D+", string)
Print (result)
Print (result 1)
```

output :

```
>>> ['1 2 34']
>>> ['ALL Might']
```

PRACTICAL : 4 :

Aim : Demonstrate the use of regular expression

Theory : R.E represents the sequence of character which is mainly based for finding & replacing the find pattern in the string & for this we import re -module & common usage of re expression, involved following functionalities :

- Searching a given string.
- Finding a given string.
- Breaking a string into smaller substring.
- Replacing part of string.

Q1 WAP on R.E seperating numeric & alphabetical value from a string.

Algorithm :

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

Step 2 : \d is used for matching all decimal digit where as \D is used to match non-decimal digits.

Q.2 W.A. R.E for finding match string at the beginning of the sequence.

Algorithm:

Step 1 : Import re module & apply a string.

Step 2 : Use search() with "\A python" and string string as 2 parameters.

Step 3 : Now display the output.

Step 4 : Now use if statement to check the match is found or not.

```
# Code : 2
import re
String = "Python is an Amazing Language"
result = re.search ('\A Amazing', String)
Print(result)
 if   result:
        Print ('Match found')
 else :
        Print ('Match not found')

# output
>>> (re.match object : span = (0, 6);
                match = 'python' >

>>> match found
```

Code 3:

```
import re
Li = ['9870450768 ","8169260459","9011869675",
      "9920105706","7030734259"]

For element in Li:
    result = re.match("[8-9]{1}
                       [0-9]{9}",element)

    if result:
        print ("Correct Mobile")
        print ((result.group()))

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

Output:

```
>>> Correct Mobile No:
    9876554321 0
    Correct mobile:
    8169260459
    Correct mobile:
    9011869675
    Incorrect mobile.
    Incorrect mobile.
```


Q3 WAP on R.E to check whether the given mobile no. start with 8 or 9 & total length of digits should be 10.

Algorithm:

Step 1: Import re module & apply a string of mobile no.s

Step 2: Now use for conditional statement to find if the starts with 8 or 9 and the total no.s should be of 10. Use match inside for statement to find the match in given string.

Step 3: Use of conditional statement to know whether we have a match or not if we have use group() to display the output and if false display in count Mobile no.

Q.4 W.R.E for extracting a word from given string along with space char in between words & subsequently extract the words.

Algorithm:

Step 1: Import re module & apply a string.

Step 2: Use find all() to extract a word from the string.

Step 3: Use "\w" to extract word along with space & use "\w+" to extract word without space.

Step 4: Now display the output.

Code 4:

```
import re
string = "Python is sugoi, subarashi"
result1 = re.findall ("\w+", string)
result2 = re.findall ("\w+", string)
Print (result 1)
Print (result 2)
```

Output:

```
>>> ['Python', ' ', 'is', ' ', 'sugoi', 'subarashi']
    ['Python', 'is', 'sugoi', 'subarashi']
```

Code 5:

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

Output:

```
>>> ('Python')
>>> ["Important"]
```

Q.5 W.A.P.R.E for extracting first and last word from a string

Step 1: Import re module and apply e string.

Step 2: Use findall() in which use "^\w+" as one parameter to find first word of string then use "\w+$" as parameter to find last word of string.

Step 3: Now display result.

Code 6:

```
import re
string = 'Amit 201  24-12-2019"
result = re.findall ['\d{2} -\d{2}-\d{4}',string]
print (result)
```

output:

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

Q.6 W.R.E for extracting the date in format. dd-mm-yyyy by using the findall(). Where the string has following format Amit 201 24-12-2019.

Algorithm:

Step 1: Import re module & apply string.

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

Step 3: Now display the output.

Q.7. W.A.R.E for extracting :-
① Username from email-id
② host name from email-id
③ Both hostname & username from email-id.

Algorithm :
Step 1 : Import re module & apply a sting.

Step 2 : Use findall() to find username & hostname & both of email_id.

Step 3 : Use "\w+" for username use "t\w+.\w+$" for hostname & username.
"[\w\.-]+" for both as parameter in findall()

Step 4 : Display the output.

Code 7 :

```
import re
string = 'abc @tcsc.edu
result1 = re.findall ("^\w+",sting)
result2 = re.findall ("t\w+.\w+$",sting)
result3 = re.findall ("[\w\.-]+",sting)
Print (result 1)
Print (result 2)
Print (result 3)
```

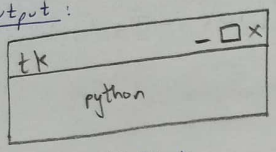
output :

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

07/01/22

```
#Creation of parent window
from Tkinter import*
root = Tk()
l = Label(root, text = "python")
l.pack()
root.mainloop()
```

Output :




```
# 2: label, attributes
from Tkinter import*
root. Tk()
l = label(root, text = "python")
l.pack()
l1 ~~pack~~ = Label(root, text = "CS!", bg = "grey", fg = "black",
                    font = "10")

l1.pack(side = LEFT, padx = 20)
~~l2.pack(side~~
l2 = Label(root, text = "cs", bg = "light blue", fg = "black",
           font = "20")
l2.pack(side = LEFT, pady = "30")
l3 = Label(root, text = "CS!", bg = "yellow", fg = "black",
           font = "10")

l3.pack(side = TOP, ipadx = 40
l4 = Label(root, text = "CS", bg = "orange", fg = "black",
           font = "10")

l4.pack(side = TOP, ipady = 50)
root.mainloop()
```


PRACTICAL : 05

Topic : GUI components

Step 1 : Use the tkinter library for importing the features of the text widget.

Step 2 : Create an object using the Tk()

Step 3 : Create a variable using the widget label and use the text method.

Step 4 : Use the mainloop() for triggering of the corresponding above mention events.

~~Solve~~ # 2 :

Step 1 : Use the tkinter library for importing the features of the text widget.

Step 2 : Create a variable from the text method & position it on the parent window.

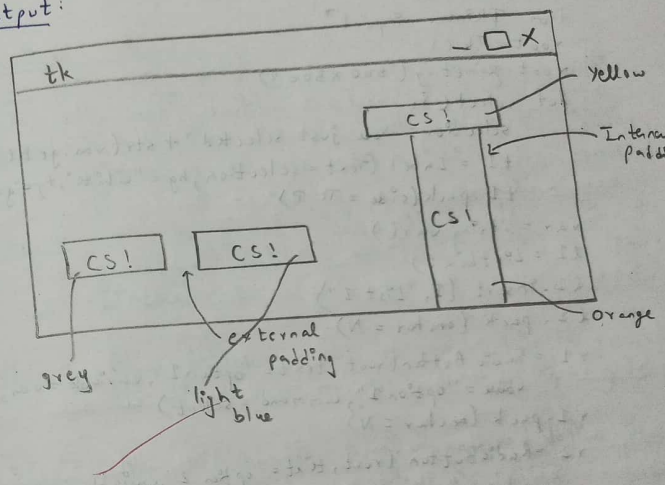
Step 3 : Use the pack() along with the object created from the text() and use the parameter.

1) side = LEFT, padx = 20
2) side = LEFT, pady = 30
3) side = TOP, ipadx = 40
4) side = TOP, ipady = 50

Step 4 : Use the mainloop() for the triggering of the corresponding events.

Step 5 : Now repeat above steps with the label() which takes the following arguments
1) Name of the parent window
2) Text attribute which define the string
3) The background color (bg)
4) The foreground fg and then use the pack() with a relevant padding attributes.

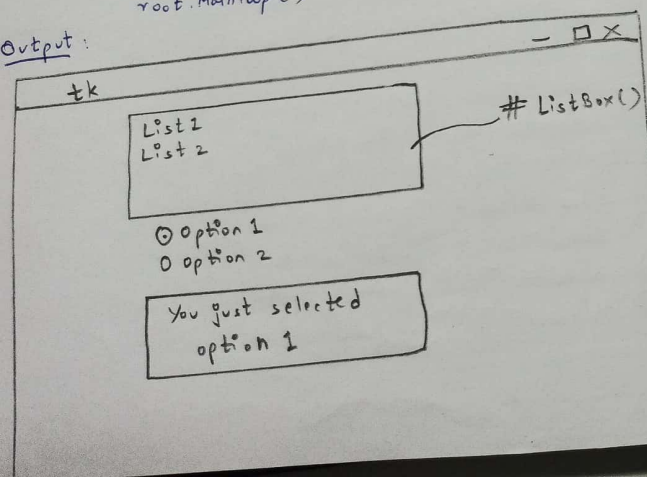
Output:



Radio Button:

```
from Tkinter import*
root = Tk()
root.geometry("500x500")
def select():
    selection = "You just selected " + str(var.get())
    t1 = Label(Text = selection, bg = "white", fg = "green")
    t1.pack(side = TOP)
var = stringvar()
l1 = Listbox()
l1.insert (1, "List 1")
l2. pack (anchor = N)
r1 = RadioButton(root, text = "option 1", variable = var,
    value = "option 1", command = select)
r1. pack (anchor = N)
r2 = RadioButton (root, text = "option 2", variable = var,
    value = "option 2", command = select)
r2. pack(anchor = N)
root.mainloop()
```

Output:



Practical - 5(B)

Aim: GUI components

1

Step 1: Import the relevant methods from the Tkinter library create an object with the parent window.

Step 2: Use the parent window object along with the geometry() declaring specific pixel size of parent window.

Step 3: Now define a function which tells the user about the given selection made from multiple option available.

Step 4: Now define the parent window and define the option with control variable.

Step 5: Use the listbox() and insert options on the parent window along with the pack() with specifying anchor attribute.

Step 6: Create an object from radio button which will take following arguments & parent window object, text variable which will take the values option no. 1, 2, 3,.. variable argument, corresponding value & trigger the function ~~lower~~ declared.

#2:
Step 1: Import relevant methods from the Tkinter library.

Step 2: Create a parent object corresponding to the parent window.

Step 3: Use the geometry() for laying of the window.

Step 4: Create an object and use the scrollbar()

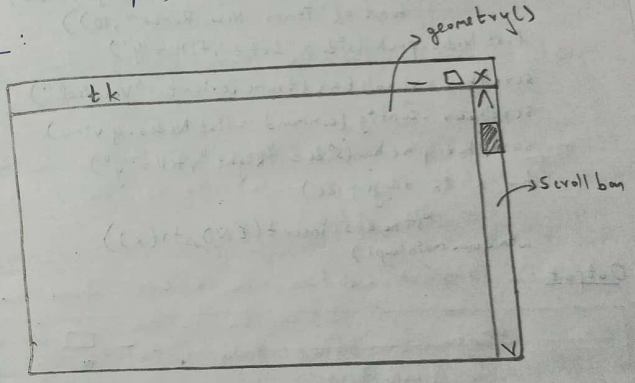
Step 5: Use the pack() along with the scrollbar object with side and fill attribute.

Step 6: Use the mainloop with the parent object.

~~#3~~

#2:
```
Scrollbar()
from tkinter import*
root = Tk()
root.geometry ("500x500")
s = scrollbar()
s.pack (side = "Right", file = "y")
root.mainloop()
```

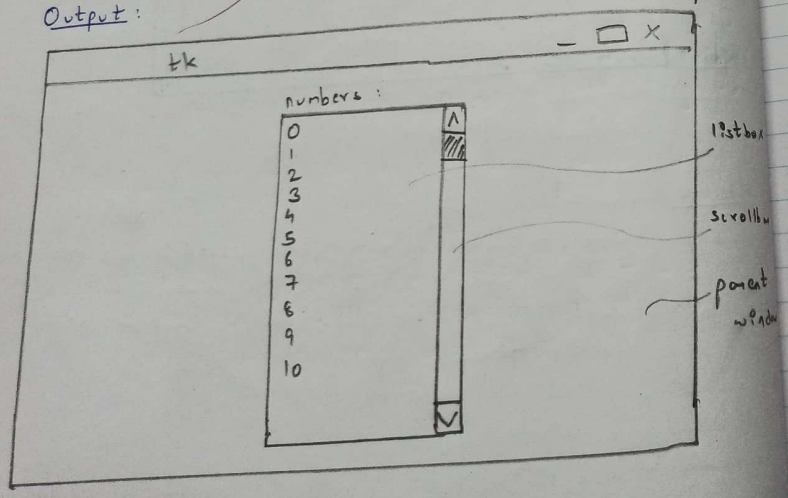
Output:



#3:
Using frame widget

```
from tkinter import*
window = Tk()
window.geometry ("680x500")
Label(window, text = "numbers:").pack()
frame = Frame(window)
frame.pack()
listNodes = Listbox(frame, width = 20, height = 20,
            font =("Times New Roman",10))
listNodes.pack (side = "Left", fill = 'y')
scrollbar = scrollbar (frame, orient = "Vertical")
scrollbar.config (command = listNodes.yview)
scrollbar.pack (side = "Right", fill = "y")
for x in range (100):
        listnodes.insert(END,str(x))
window.mainloop()
```

Output:



#3:

Step 1: Import the relevant libraries from the tkinter method.

Step 2: Create an corresponding object of the parent window.

Step 3: Use the geometry manager with pixel size (680x500) or any other suitable pixel value.

Step 4: Use the label widget along with the parent object created and subsequently use the pack().

Step 5: Use the frame widget along with the parent object created and use the pack().

Step 6: Use the listbox method along with the attributes like width, height font Do create a listbox methods object. Use pack() for the same.

Step 7: Use the ~~screen~~ Scrollbar() with an object. Use the attribute of vertical.

Step 8: Trigger, the events using mainloop.

#4:

Step 1: Import relevant methods from tkinter library.

Step 2: Define the object corresponding to parent window and define the size of parent window in terms of no. of pixels.

Step 3: Now define the frame object from the method and place it on to the parent window.

Step 4: Create another frame object termed as the left frame and put it on the parent window on its LEFT side.

Step 5: Similarly, define the RIGHT frame and subsequently define the button object placed onto the given frame with the attribute as text, active background and foreground.

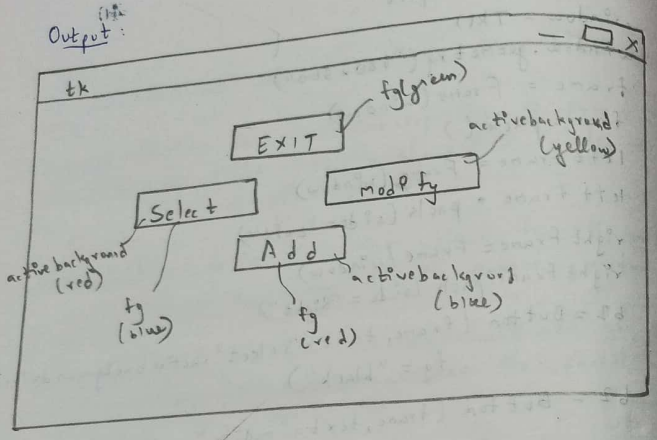
Step 6: Now use the pack() along with the side attribute.

Step 7: Similarly, create the button object corresponding to the MODIFY operation put it into frame object on side = "right".

#4:

```
from Tkinter import*
window = Tk()
window.geometry("680x500")
frame = Frame(window)
frame.pack()
leftFrame = Frame(window)
left frame = pack(side="Left")
right Frame = Frame(window)
right frame.pack(side="Right")
b1 = Button(frame, text="select", activebackground="red",
            fg="black")
b2 = Button(frame, text="modify", activebackground="blue",
            fg="black")
b3 = Button(frame, text="ADD", activebackground="blue",
            fg="red")
b4 = Button(frame, text="EXIT", activebackground="yellow",
            fg="green")
b1.pack(side="LEFT", padx=20)
b2.pack(side="right", padx=30)
b3.pack(side="bottom", pady=20)
b4.pack(side="top")
```

Output:



Step 8: Create another button object & place it on to the RIGHT frame & label the button as ADD.

Step 9: Add another button & put it on the top of frame and label it as EXIT.

Step 10: Use the ~~pack~~() simultaneously for all the objects & finally use the mainloop().

PRACTICAL : 5(c)

Aim : GUI components

Step 1 : Import the relevant methods from tkinter library.

Step 2 : Import tk Message Box

Step 3 : Define a parent window object along with the parent window.

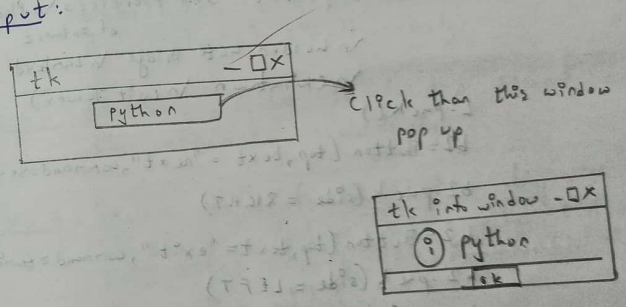
Step 4 : Define a function with will use tk message box with showinfo method along with info window attribute.

Step 5 : Declare a button with parent window object along with the command attribute.

Step 6 : Place the button widget onto the parent window and finally ~~loop~~ call mainloop() for triggering of the events called above.

```
#message box
    from tkinter import*
    import tk Message Box
    root = Tk()
    def function():
        tk Message Box.showinfo("info window", "python")
        b1 = Button(root, text = "python", command = function)
        b1.pack()
        root.mainloop()
```

Output :



```
# Multiple window
# Different button (Relief())
from Tkinter import*
root = Tk()
root.minsize (300,300)
def main():
    top = Tk()
    top.config(bg="black")
    top.title("HOME")
    top.minsize(300,300)
    l= Label(top,text="San Francisco \n Places
                        of Interest
        \n Golden Gate Bridge \n Lombard Street
        \n Chinatown \n Coit Tower")
    l.pack()
    b1= Button(top,text ="next",command=second)
    b1.pack(side =RIGHT)
    b2=Button(top,text="exit",command=terminate)
    b2.pack(side = LEFT)
    top.mainloop()
```


Step 1: Import the relevant methods from the tkinter library, along with parent window object declared.

Step 2: Use parentwindow object along with minsize function for window size.

Step 3: Define a function main, declare parent window object and use config(), title(), minsize(), Label() as well as button() and use pack() & mainloop() simultaneously.

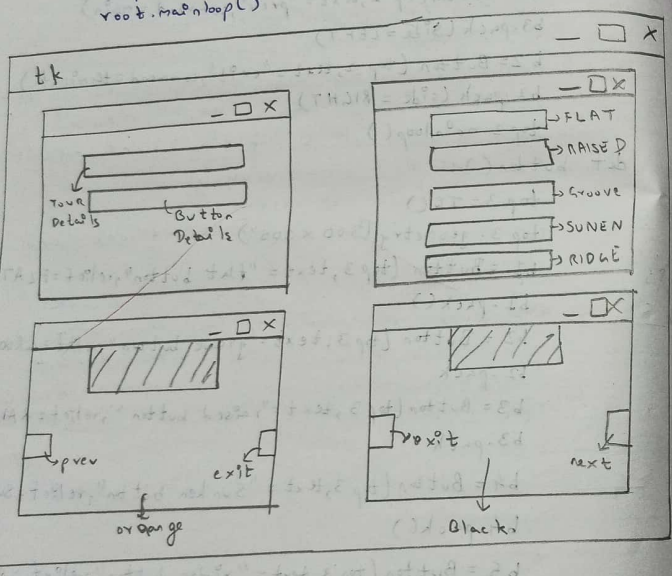
Step 4: Similarly, define the function second and use the attributes accordingly.

Step 5: Declare another function button along with parent object and declare button with attributes like FLAT, RIDGE, GROOVE, RAISED, SUNKEN along with the relief widget.

Step 6: Finally called the mainloop() for event driven programming.

```
def second():
    top 2 = Tk()
    top 2.config(bg="orange")
    top 2.title("About us!")
    top 2.minsize(300,300)
    L=Label(top 2, text= "Created by: Tanvi More.\n
    For more details contact to our official account")
    L.pack()
    b3 = Button(top 2, text = "prev", command = main)
    b3.pack(side = LEFT)
    b 2= Button(top 2, text = "exit", command = terminate)
    b2.pack(side = RIGHT)
    top 2.mainloop()
def button():
    top 3 = Tk()
    top 3. geometry("300 x 300")
    b1 = Button(top 3, text= "flat button", relief = FLAT)
    b1.pack()
    b2 = Button(top 3, text = "groove button", relief = GROOVE)
    b2.pack
    b3 = Button(top 3, text = "raised button", relief = RAISED)
    b3.pack
    b4 = Button(top 3, text = "Sunken button", relief = SUNKEN)
    b4.pack()
    b5 = Button(top 3, text = "ridge button", relief = RIDGE)
    top 3.mainloop()
```

```
def terminate():
    quit()
b5 = Button(root, text="TOUR DETAILS", command=main)
b5.pack()
b6 = Button(root, text="BOUTTON DETAILS", command=button)
b6.pack()
root.mainloop()
```



PRACTICAL - 5(D)

Aim : GUI components

Step 1 : Import relevant methods from the tkinter library.

Step 2 : Create parent window object and use the config method along with background colour attribute specified.

Step 3 : Define a function finish with the messagebox widget which will display a message i.e. a warning message and subsequently terminate the program.

Step 4 : Define a function info use a listbox widget along with the object of the same. Use the lpstbox object along with insert method and insert the same and finally use the grid() with ipadx attribute.

Step 5 : Define a function about us with label object and text attribute and subsequently use the grid().

Step 6 : Use photo image widget with file and filename with gif attribute.

Step 7 : Create a frame object along with the Frame() along with parent window object height & width specified. and subsequently use the grid() with row & column attribute specified.

Step 8 : Similarly, create another frame object as declared by step 7.

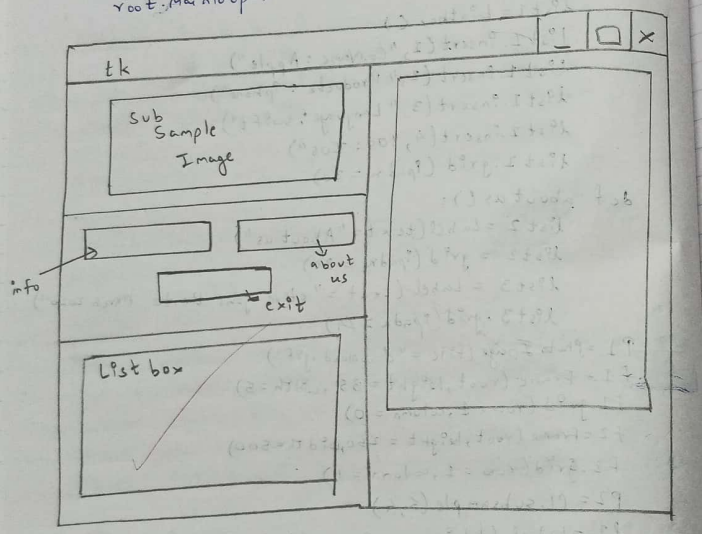
Step 9 : Create another object & use the subsample (5,4)

Step 10 : Use label widget along with frame object, relief attribute and subsequently use the grid().

Step 11 : Now create button object dealing with different section of frame.

```
from Tkinter import*
root = Tk()
root = config(bg="grey")
def finish():
    messagebox.askokcancel("Warning","This will end program")
    quit()
def info():
    list1 = Listbox()
    list1.insert(1,"Co.Name : Apple")
    list1.insert(2,"Products : iphone")
    list1.insert(3,"Language : swift")
    list1.insert(4,"OS : IOS")
    list1.grid(padx = 30)
def about us():
    list2 = Label(text="About us")
    list2 = grid(padx = 30)
    list3 = Label(text="steve jobs theatre March 2020")
    list3.grid(padx = 24)
P1 = PhotoImage(file = "download.gif")
f1 = Frame(root,height = 35,width = 5)
f1.grid(row = 1,column = 0)
f2 = Frame(root,height = 250,width = 500)
f2.grid(row = 1,column = 1)
P2 = P1.subsample(5,4)
l1 = Label(f1,image = P2,relief = FLAT)
l1.grid(row = 1,column = 0,padx = 20,pady = 15)
l2 = Label(f2,image = p2,relief = SUNKEN)
l2.grid(padx = 25,pady = 10)
b1 = Button(f1,text = "Information",relief = SUNKEN,
    command = info)
b1.grid(row = 1,column = 0)
```

```
b2 = Button (f1, text = "About us", relief = SUNKEN, command = Abou
b2.grid (row = 1, column = 2, padx = 5)
b3 = Button (f1, text = "EXIT", relief = RAISED, command = FIN
b3.grid (row = 2, column = 1, ipadx = 15)
root.mainloop()
```



PRACTICAL = 5(E)

Aim : GUI components

1 : Spinbox :

Step 1 : Import relevant ()'s from the tkinter library

Step 2 : Create parent window object along with Tk()

Step 3 : Create an object from spinbox method and use the attributes parent window object, from - & to attributes).

Step 4 : Subsequently call the pack method along with spinbox object & called the mainloop().

2. Paned window :

Step 1 : Import relevant methods from the tkinter library also create a parent window object.

Step 2 : Create an object along with paned window and subsequently use the pack() along with the paned window object along with attributes like fill & expand.

Step 3: Create an Label object along with Label() use paned window object, orient & background.

Step 4: Similarly, create another label method and use the add() subsequently.

Step 5: Finally called the mainloop() for event trigger.

#3: Canvas:

Step 1: Import relevant ()'s from tkinter library declare a parent window object.

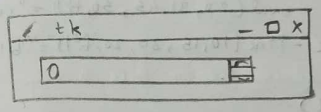
Step 2: Create a canvas object along with the canvas() with attributes parent window object height, width & background colour.

Step 3: Use create src() along with canvas object declared along with start & extent & co-ordinate.

Step 4: Similarly, for oval & line use the pack() and call mainloop() for event driven programming.

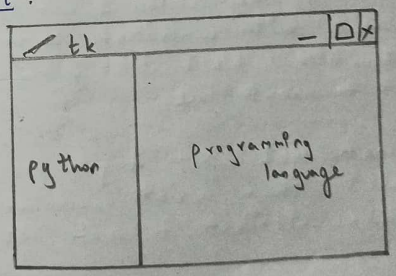
```
from Tkinter import*
master = Tk()
S = spinbox(master, from_=0, to 10)
S.pack()
master.mainloop()          #spinbox widget
```

Output:




```
from tkinter import*
root = Tk()
p = paned window()
p.pack(fill = BOTH, expand = 10)
l = Label(p, text = "PYTHON")
p.add(L)
p1 = paned window(p, orient = VERTICAL, bg = "black")
p1.add(p1)
l1 = Label(p1, text = "Programming Language")
p1.add(L1)
root.mainloop()
```

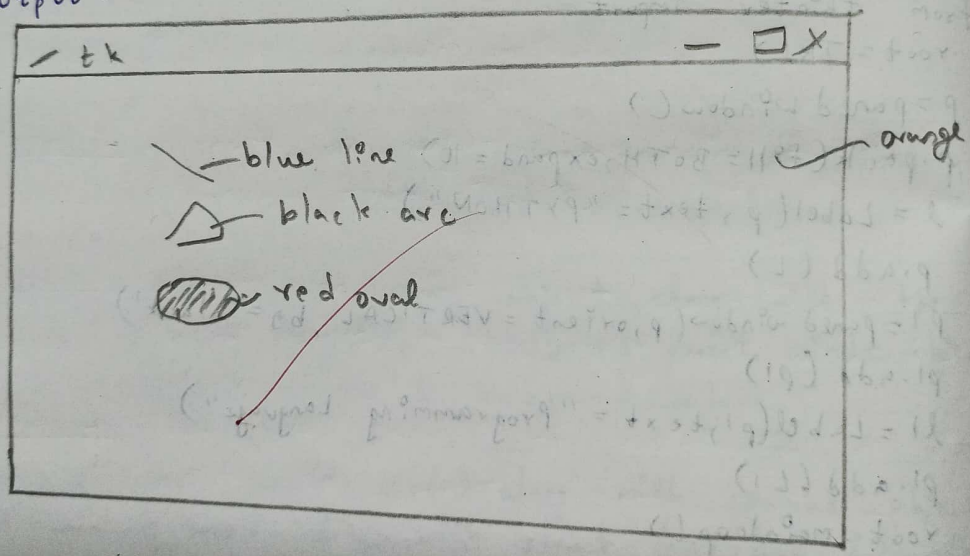
Output:



#3

```
from Tkinter import*
root = Tk()
C = Canvas(root, height = 100, width = 200, bg = "orange")
carc = C.create_arc(10, 20, 30, 40, start = 10, extent = 50,
      fill = "black")
Coval = C.create_oval(20, 31, 45, 50, fill = "red")
Cline = C.create_line(10, 15, 30, 20, fill = "blue")
C.pack()
root.mainloop()
```

Output:



Practical - 6

Aim : Demonstrate the use of GUI by creating a human face and converting celsius into fahrenheit.

Q.1] Write a program to draw human face using GUI.

Algorithm :

Step 1 : Import relevant methods from tkinter library.

Step 2 : Create an object corresponding to the parent window from Tk().

Step 3 : Create an object from canvas() & place it onto parent window along with height & width.

Step 4 : Now use pack() for positioning of widget on to the parent window.

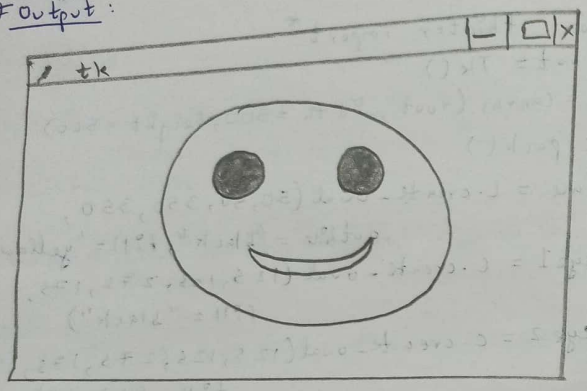
Step 5 : Now create an object face & use object create_oval() with co-ordinate 50, 50, 350, 350 & outline = "black", fill = "yellow" as attribute to create face.

Step 6 : Now create eye1 object & again use object. Create_oval() with appropriate co-ordinates along with fill as attribute to create left eye.

Code :

```
from tkinter import*
root = Tk()
C = Canvas(root, width = 500, height = 500)
C.pack()
face = C.create_oval(50,50,350,350,
                     outline = "black", fill = "yellow")
eye1 = C.create_oval(125,125,275,175,
                     fill = "black")
eye2 = C.create_oval(125,125,275,175,
                     fill = "black")
mouth = C.create_arc(125,225,275,275,
                     start = 0, extent = -180, width = 5,
                     fill = "red")
root.mainloop()
```

#Output:



Step 7: Now repeat the same step 6 to create right eye.

Step 8: Create an object mouth & use object.create_arc() with appropriate co-ordinates, start=0, extent=-180 & fill="red", width=5 as attribute to create mouth.

Step 9: Finally use the mainloop()

Code :

```
from tkinter import *
window = Tk()
fahrenheit = DoubleVar()
fahrenheit.set(32.0)
def convert(celsius):
    fahrenheit.set((9.0/5.0)*celsius + 32)
l1 = Label(window, text="Temperature in Celsius")
l1.grid(row=0, column=0)
e = Entry(window, textvariable=celsius)
e.grid(row=0, column=1)
celsius = IntVar()
l2 = Label(window, textvariable=fahrenheit)
l2.grid(row=2, column=0, columnspan=2)
B = Button(window, text="Calculate", command=lambda:
    convert(celsius.get()))
B.grid(row=1, column=0, columnspan=2)
window.mainloop()
```

Q.2] Write a program to convert celsius to fahrenheit using GUI.

Algorithm :

Step 1 : Import all the relevant methods in tkinter library.

Step 2 : Create object convertPending to the parent window from Tk().

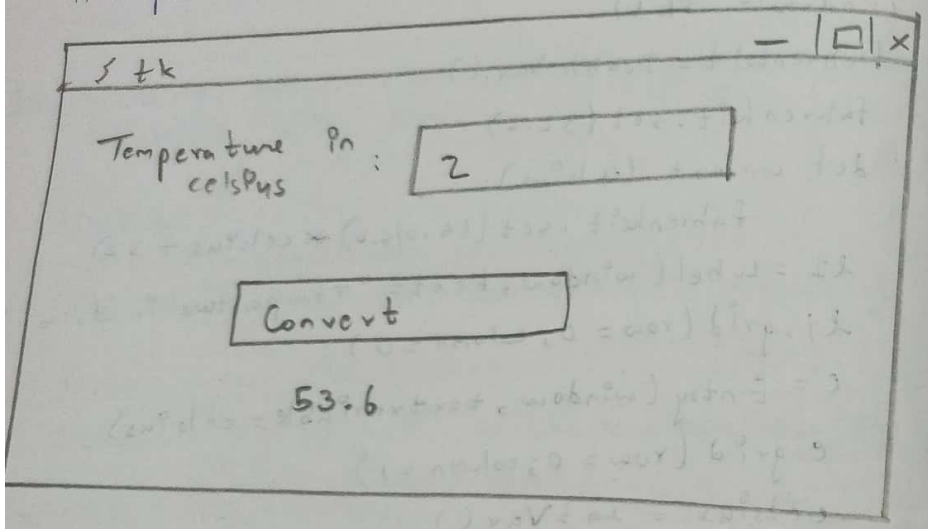
Step 3 : Define function fahrenheit as DoubleVar()
And set it to 32.0

Step 4 : Now define a function convert with argument celsius. And convert celsius fahrenheit using set().

Step 5 : Now create an object L2 using Label() & place it onto parent window And use text attribute as entry a No. :

Step 6 : Now use grid() by position the object onto the parent window.

#Output:



Step 7 : Initialize celsius as integer using intvar()

Step 8 : Create another object & use entry widget to entry the input and place it onto the parent window.

Step 9 : Now use grid() for positioning the object onto parent window with text variable attribute.

Step 10 : Now again use label() along with text variable attribute to display output & use grid() for positioning.

Step 11 : Finally use mainloop().

Practical No. 7

Aim: Write a program to find factorial of number & use arithmetic operations on two numbers using GUI

Algorithm: [# 1] :- fact using GUI

step 1: Import relevant methods from tkinter library.

step 2: Now Define a function factorial to calculate factorial using recurrsive function.

step 3: Define another function calculate to call factorial function.

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

step 5: Now create an object with button() along with command = attribute to calculate factorial.

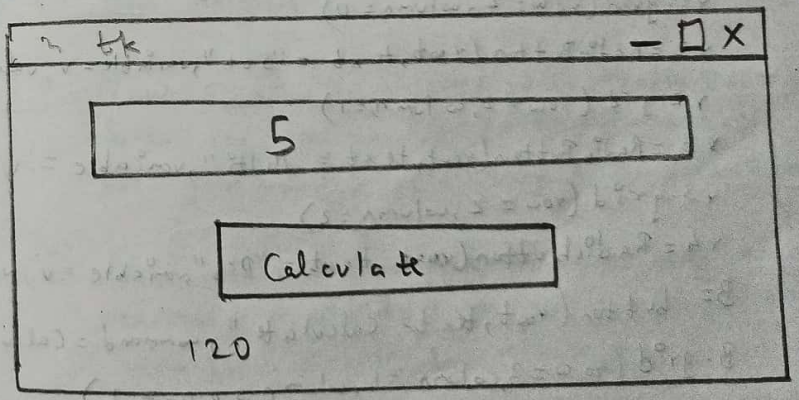
step 6: Now again create an object with label() to show output.

Step 7: Finally use the mainloop().

Coding :- [# 1] : fact using GUI

```
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_text.get()))
    info.config(text=result)
root=Tk()
entry_test=Entry(root)
entry_test.pack()
bth=Button(root,text="Calculate",command=Calculate)
bth.pack()
info=Label(root,text="factorial")
info.pack()
root.mainloop()
```

Output : [# 1]



#code:

```
from tkinter import*
def calculate():
    if int(v.get())==1:
        res=int(e1.get())+int(e2.get())
        f3.config(text=res)
    elif int(v.get())==2:
        res=int(e1.get())-int(e2.get())
        f3.config(text=res)
    else:
        res=int(e1.get())/int(e2.get())
        f3.config(text=res)

root=Tk()
l1=Label(root,text="Enter a no.")
l1.grid(row=0,column=0)
e1=Entry(root)
e1.grid(row=0,column=1)
l2=Label(root,text="Enter 2nd no.")
l2.grid(row=1,column=0)
v=IntVar()
r1=RadioButton(root,text="add",variable=v,value=1)
r1.grid(row=2,column=0)
r2=RadioButton(root,text="Sub",variable=v,value=2)
r2.grid(row=2,column=1)
r3=RadioButton(root,text="Multi",variable=v,value=3)
r3.grid(row=2,column=2)
r4=Radiobutton(root,text="Div",variable=v,value=4)
B=button(root,text="Calculate",command=calculate)
B.grid(row=3,column=1,columnspan=2)
```


[# 2] : Arithmetic Operation on 2 Numbers using GUI.

Step 1 : Import relevant methods from Tkinter library.

Step 2 : Now create an object corresponding to parent window

Step 3 : Now define a function calculate to carry out arithmetic operations on 2 numbers.

Step 4 : Now create an object with label() as num1 & num 2 and use the grid() to place it onto parent window.

Step 5 : Create object with entry() to take input from user().

Step 6 : Now initialize v as integer using IntVar().

Step 7 : Now create 4 objects with RadioButton() to choose any one of the arithmetic operation & use grid() for positioning onto parent window.

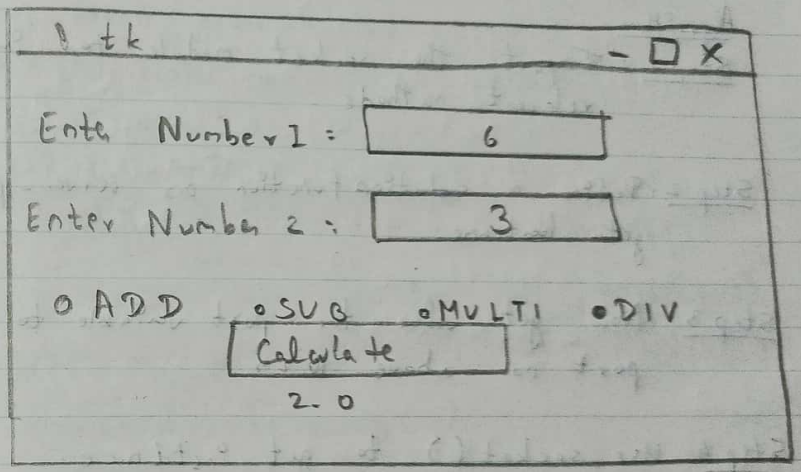
Step 8 : Now create a object with button() along with command & attribute to carryout the arithmetic operation of user's choice.

Step 9 : Now create a object with label() to show the output.

Step 10 : Use the mainloop().


```
l3.grid(row=4, column=1)
root.mainloop()
```

Output : [# 2]



PRACTICAL : 8

Aim : Demonstrate the use of socket module and server client programs.

Algorithm:

Step 1 : Import the socket module to import relevant methods.

Step 2: Define a ~~solution~~ function as server - program to get hostname.

Step 3: Now get value for port variable to initialize port no. above 1024

Step 4. Use .socket() to get instance.

Step 5: Now use bind() function to bind host address and port together to configure how many client the server can list simultaneously.

Step 6: Use accept() to accept new connection.

Step 7: Now print the address.

Step 8: Use while loop as true to receive data stream.

#code:

```
import socket
def server-program
    host = socket.get hostname()
    port = 5000
    server .socket = so ut.socket()
    server socket = socket.bind (root.port))
server.socketlist (no. 2)
conn, address = server - socket.accept()
print ("connection from :" - to str(address))
while True:
    data = conn. rec(1024) decode()
    if not data:
        break
    print ("from connected user: " tst (data))
← data = input("→")
← conn. send (data.encode())
← conn. close()
```

Output:

```
Python 36 socket-server.py
connection from : ('127.0.0.1', 57822)
from connected user: Hi
→ Hello
from connected user: How are you?
→ Good
from connected user : Ame 30
→ ok then, Bye
```

#Code 2:

```
import socket:
def client_prog :-
    host = socket.get host name()
    host = 5000
client_socket = socket.socket()
client_socket.connect((host, port))
message = input(" → ")
while message.use() ~~set~~ send() != 'bye':
    client = socket.send(message.encode())
    data = client_socket.rcv(1024).decode()
print("Received from server", data)
message = input(" → ")
client_socket:close()
```

Output:

Python 3.6 socket_client.py

→ Hi
Received from server: Hello
→ How are you?
Received from server: Good
→ Awesome.
Received from server: Ok then, Bye!
→ Bye.

Step 9: Now close the program

[#2]: [For Socket Client program]

Algorithm:

Step 1: Import Socket module to import methods that are relevant.

Step 2: Define a function client_program get the hostname and give port a value 5000.

Step 3: Now again initiate by using socket.socket()

Step 4: Use connect() to connect the server.

Step 5: Now take the input(" > ")

Step 6: Use while conditional loop to send a message.

Step 7: Now use decode to receive response

Step 8: Now show the data.

Step 9: Again take input.

Step 10: Close the program by using close().

code in shell environment :-

```
>>> import sqlite3
>>> conn = sqlite3.connect("student.db")
>>> conn = conn.cursor()
>>> curr = execute ('create table student (roll_no int(5),
    primary key, name varchar(50) not null, class varchar(50),
    dob date)')
<sqlite3.cursor object at 0x0322EBED>
>>> conn.execute (insert into students value (101, 'Abhay',
    "Vasai", FyC.S", 24/01/2002"))
<sqlite3.anchor object at 0x0322EBED>
>>> conn.execute (insert into students values (102, 'Shubham',
    'Borivali', "FYCS", "02/08/2002"))
<sqlite3.cursor object at 0x0322EBED
>>>cur.execute ("select * from student")
<sqlite3.cursor object at 0x0322EBED>
>>> cur.fetchall()
[(101, 'Abhay', 'Vasai', 'FYCS', '24/01/2002'),
 (102, 'Veera', 'Borivali', FYCS', '02/08/2002')]
>>> cur.execute ('update student set dob = "13/09/2001',
        where roll_no = 101')
<sqlite3.cursor object at 0x0322EBED>
```


PRACTICAL : 9

Aim : Demonstrate the use of database connectivity

Algorithm :

Step 1 : Import sqlite3 module to import relevant methods.

Step 2 : Now initialize a variable connect to connect by using connect() to a new database using extension.

Step 3 : Now initialize a variable to connect to cursor().

Step 4 : Now use cur.execute() to create a table, insert values into table & use DML, DDL statement to manipulate the data in this database.

Step 5 : Use fetchall() to show the output.

Step 6 : Use commit to save all change.

Step 7 : Use close() to terminate the program.

```
>>> cur.fetchall()
[(101, 'Abhay', 'Vasai', 'Fycs', '13/08/2001')]
>>> conn.execute('commit')
<sqlite3.Cursor object at 0x032EBED0>
>>> cur.close()
```

```
from tkinter import *

phonelist = [['Chris', 'Meyers', '241-343-4349'],
             ['Robert', 'Smith', '202-689-1234'],
             ['Janet', 'Jones', '609-483-5432'],
             ['Ralph', 'Barnhart', '215-683-2341'],
             ['Eric', 'Nelson', '571-485-2689'],
             ['Ford', 'Prefect', '703-987-6543'],
             ['Mary', 'Zigler', '812-567-8901'],
             ['Bob', 'Smith', '856-689-1234']]

def which_selected():
    print("At {0}".format(select.curselection()))
    return int(select.curselection()[0])


def add_entry():
    phonelist.append([fnamevar.get(), lnamevar.get(), phonevar.get()])
    set_select()


def update_entry():
    phonelist[which_selected()] = [fnamevar.get(),
                                   lnamevar.get(),
                                   phonevar.get()]


def delete_entry():
    del phonelist[which_selected()]
    set_select()


def load_entry():
    fname, lname, phone = phonelist[which_selected()]
    fnamevar.set(fname)
    lnamevar.set(lname)
    phonevar.set(phone)


def  make_window ():
    global  fnamevar ,  lnamevar ,  phonevar ,  select
    win  =  Tk ()

    frame1 = Frame(win)
    frame1.pack()

    Label(frame1, text="First Name").grid(row=0, column=0, sticky=W)
    fnamevar = StringVar()
    fname = Entry(frame1, textvariable=fnamevar)
    fname.grid(row=0, column=1, sticky=W)
```

```
    Label(frame1, text="Last Name").grid(row=1, column=0, sticky=W)
    lnamevar = StringVar()
    lname = Entry(frame1, textvariable=lnamevar)
    lname.grid(row=1, column=1, sticky=W)

    Label(frame1, text="Phone").grid(row=2, column=0, sticky=W)
    phonevar = StringVar()
    phone = Entry(frame1, textvariable=phonevar)
    phone.grid(row=2, column=1, sticky=W)

    frame2 = Frame(win)
    frame2.pack()
    b1 = Button(frame2, text=" Add  ", command=add_entry)
    b2 = Button(frame2, text="Update", command=update_entry)
    b3 = Button(frame2, text="Delete", command=delete_entry)
    b4 = Button(frame2, text="Load  ", command=load_entry)
    b5 = Button(frame2, text="Refresh", command=set_select)
    b1.pack(side=LEFT)
    b2.pack(side=LEFT)
    b3.pack(side=LEFT)
    b4.pack(side=LEFT)
    b5.pack(side=LEFT)

    frame3 = Frame(win)
    frame3.pack()
    scroll = Scrollbar(frame3, orient=VERTICAL)
    select = Listbox(frame3, yscrollcommand=scroll.set, height=6)
    scroll.config(command=select.yview)
    scroll.pack(side=RIGHT, fill=Y)
    select.pack(side=LEFT, fill=BOTH, expand=1)
    return win


def set_select():
    phonelist.sort(key=lambda record: record[1])
    select.delete(0, END)
    for fname, lname, phone in phonelist:
        select.insert(END, "{0}, {1}".format(lname, fname))


win  =  make_window ()
set_select ()
win . mainloop ()
```