Practical no - 1

Objective : Demonstrate the use of different file accessing mode, different attribute read method.

Step 1 :- Create a file object using open method and use the write access mode followed by writing some contents onto the file and than closing the file

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

Step 3 :- Now use the file object for finding the name of the 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

OSU

```
Fileobj = open ("abc.txt", "w")# file open (write mode)
fileobj.write ("computer science subject"+"\n")
Fileobj.write ("DBMS in python in DS\n") # file write
fileobj.close () # file close

fileobj = open ("abc.txt", "r") # read mode
# read()
str1 = Fileobj.read()
print ("The output of read method:", str1)
fileobj.close ()
>>> ('The output of read method:', 'computer science\n
    in DBMS in python in DS\n')

# readline()
fileobj = open ("abc.txt", "r")
str2 = fileobj.readline ()
print ("The output of readline method:", str2)
fileobj.close ()
>>> ('The output of readline method:', 'computer scie
    subject\n')

# readlines ()
fileobj = open ("abc.txt", "r")
str3 = fileobj.readlines()
print ("The output of readlines method:", str3)
fileobj.close ()
>>> ('The output of readlines method:', ['comp
    subject\n', 'DBMS\n', 'python\n', 'DS\n'])
```

Step 4 :- Now open the file object in write mode write some other contents close subsequently then again open the file object in 'w+' mode that is the update mode and write contents.

Step 5 :- open file object in read mode, display the update written contents and close open again in 'r+' mode with parameter passed and display the output subsequent

Step 6 :- Now open file object in append mode write method write conten close the file obj again open the file object in read mode and display append output.

```
# 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 = file obj.mode
print ("file mode", c)
>>> ('file mode', 'r')
d = fileobj.softspace
print ("softspace", d)
>>> ('softspace:', 0)

# w+mode
fileobj = open("abc.txt", "w+")
fileobj.write ("faizan")
file obj.close()

# r+mode
file obj = open ("abc.txt", "r+")
str1 = file obj.read (8)
print ("output of r+", str1)
fileobj.close()
>>> ('output of r+', 'faizan')

# write mode
fileobj = open ("abc.txt", "w+")
fileobj.write ("DBMS")

# read mode
file obj = open("abc.txt"
str2 = file obj.readl
print ("output of re
mode!", st
>>> ('output of rea
mode!', 'faizan
```

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 statements and display the length.

```
#Append mode
fileobj = open("abc.txt", "a")
fileobj.write("data Structure")
fileobj.close()
fileobj = open("abc.txt", "r")
str3 = fileobj.read()
print("output of append mode:", str3)
fileobj.close()
>>> ('output of append mode:', 'Faizan', 'data Structure')
# tell()
fileobj = open("abc.txt", "r")
pos = fileobj.tell()
print("tell():", pos())
fileobj.close
>>> ('tell():', pos)
# seek()
fileobj = open("abc.txt", "r")
str4 = fileobj.seek(0,0)
str0 = fileobj.read(10)
print("the beginning of the file:", str0)
# Finding length of different lines exist within files
fileobj = open("abc.txt", "r")
str9 = fileobj.readlines()
print("output:" str9)
for line in str9:
    print(len(line))
fileobj.close()
>>> ('output:', ['college database'])
```

Aim:

To display elements of a tuple using iterator method.

Algorithem :-

Step 1 :- form a tiple with certain elements inserted in it.

Step 2 :- use iter method with tuple and assign it a variable

Step 3 :- use the next method with variable and print elements.

To use iter method with for loop.

Algorithem.

Step 1: form a tuple with certain element inserted in it.

Step 2 :- use the for conditional statement each element of tuple.

Step 3 :- print the element of tuple.

Code :-

```
mytuple = ("Faizan","Ashitosh","preet","Adarsh")
myiter = iter(mytuple)
print(next(myiter))
print(next(myiter))
print(next(myiter))
print(next(myiter))
```

Output :-

```
Faizan
Ashitosh
preet
Adarsh
```

code :-

```
mytuple = ("faizan","Ashitosh","preet","Adars
for a in mytuple:
    print(a)
```

output:

```
Faizan
Ashitosh
preet
Adarsh
```

Aim: To find odd numbers in given range using iter method

Algorithm:-

Step 1:- Define a class which will contain various function.

Step 2:- Define the iter method with an argument and return the value of argument.

Step 3:- Define a function which increment the value of argument by two.

Step 4:- Create an object which inherit the of class and take the user input.

Step 5:- use the for condition statement followed by if conditional statement and execute.

Program

```
class odd:
    def __iter__(self):
        self.num=1
        return self
    def next(self):
        num = self.num
        self.num+ = 2
        return num
myobj = odd()
myiter = iter(myobj)
x = int(input("Enter range number: "))
for i in range myiter:
    if i<x:
        print(i)
```

output

```
Enter a number: 6
1
3
5
```

Aim: To print first 20 numbers using itr method

Algorithm :-

Step 1 = Define a class which will contain various function.

Step 2 = define iter method with an argument and return the value of argument.

Step 3 = Define next method which increment the value of arguments by 1 & print it

Step 4 = Create a object which inhanits the property of class and take the user input.

Step 5 = Use for loop to print the value of the variable.

Program

```
class myclass:
    def __iter__(self):
        self.a = 1
        return self
    def __next__(self):
        if self.a <= 20:
            x = self.a
            self.a += 1
            return x
        else:
            raise StopIteration
myobj = myclass()
myiter = iter(myobj)
for x in myiter:
    print(x)
```

Output

```
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
```

ass

program :-

```
num = [0, 4, 5, 7, 9, 11, 13, 15, 20, 9]
num = list (map (lambda x: x.%.5, num))
print (num)
def even(x):
    if x % 2 == 0:
        return "even"
    else:
        return "odd"
list (map (even, num))
```

output :-

[0, 4, 5, 7, 9, 11, 13, 15, 20, 9]

['even', 'even', 'odd', 'odd', 'odd', 'odd', 'odd', 'odd', 'even', 'odd'].

Aim :- To Find if the number is odd or eve from given list using map method.

Algorithum :-

Step 1 = Declare a list num variable and declare some elements

Step 2 = Define a function even which consist various conditional statement.

Step 3 = Further use if conditional Statement to check the module number of each element of list and retain message accordingly.

Step 4 = Use the map method to print the value in list format

Aim : To find square of a number without using map method.

Algorithm :-

Step 1 : Define a list which contains certain values.

Step 2 : Define an empty list.

Step 3 : use for loop followed by append method to append the result into the empty list.

Step 4 : print the value of list.

program

```
list = [1,2,3,4,5]
empty = [ ]
for i in lis 1:
        empty.append(i**2)
print (empty)
```

output :

[1, 4, 9, 16, 25]

program

```
def square (x):
    return (x**2).
def cube (x):
    return (x**3)
funct = [square, cube]
for i in range (4):
    value = list (map (lambda x : x(i), funct(i))
print (list (value))
```

output

[0, 0]

[1, 1]

[4, 8]

[9, 27]

[16, 64]

Aim - To find square and cube of number simultaneously using map method.

Algorithm:

Step 1: Define a fuction which returns the square of given number

Step 2: Define another function which returns cube of given numb

Step 3: Store the output since (tuple) in a list.

Step 4: Use the for loop followed by map fuction and print the val of result.

Aim: Program to demonstrate exception handling.

① program for demonstrating the use of IOError.

Step 1 = Use the try block to define the normal course of action. for eg. Define the file object and open the file in the write error mode and write some content onto the file.

Step 2 = Use the except block with the I/O error as an environment error and convey the appropriate message to the user, else display the message that the operation is carried out successfully.

② program to demonstrate the multiply exception IOerror and value Error.

Step 1 = Use the try block and define the file object and open the file in write or read mode and write some content onto the file

step 2 = Also, accept the value from the user if it is a valid value display the entered value and terminate the condition by using the break statement

step 3 = Define the except blocks for I O Error and value error.

code

① try:
    file obj = open ('abc.txt", "w")
    fileobj.write ("python is an interpreted language. In python is an identified language")
except IOError:
    print ("There is an environment error")
else:
    print ("operation succesfull")

>>> operation succesful

② while True:
    try:
        file obj = open ("abc.txt", "w")
        file obj.write ("python is an identified language \n python is an interpreted language")
        a = int (input ("Enter a number:"))
        print (a)
        break
    except IOError
        print ("There is an environment error")
    except value Error:
        print ("The value is invalid")

>>>
Enter a number: abc
The value is invalid
Enter a number: paz
The value is invalid
Enter a number: 73

② try:

```
fileobj = open("abc.txt", "w")
fileobj.write("python is an inclinted language")
except IOE error:
        print("It is an Environment Error")
else:
        print("operation is succesful");
```

output

operation is succesful.

Topic : Regular expression

Step 1 = Import re module declare pattern and declare sequence use match method with declare arguments if arguments matched then. print the same otherwise print pattern NOT FOUND!

Step 2 = Import re module declare pattern with literal and into character Declare string value use the findall() arguments and print the same

Step 3 = Import re module declare pattern with meta character use the split () and print the output.

match()

```
import re
pattern = r"fycs"
sequence = "fycs represents computer science stream"
if re.match(pattern, sequence):
    print("matched pattern found!")
else:
    print("NOT FOUND")
```

>>> matched pattern found!

numerical values (segregation)

```
import re
pattern = r'\d+'
string = 'hello123, howdy 709, 4 shocore'
output = re.findall(pattern, string)
print(output)
```

>>> ['123', '709', '4']

split()

```
import re
pattern = r'\d+'
string = 'hello123, howdy 709, 4 shocore'
output = re.split(pattern, string)
print(output)
```

>>> ['hello', ' howdy ', ' shocore']

SEO

```
# no space:
import re
String = 'abc defghi'
pattern = r'|st'
replace = ''
V1 = re.sub (pattern, replace, string)
print (V1)
>>> abc defghi
```

```
# group()
import re
sequence = 'python is an intersting language
V = re.search ('^Python', sqeuence)
print (V)
V1 = V.group()
print (V1)
>>> <_sre.SRE_Match object at 0x02811BF00>
    python
```

```
# verfying the given set of phone number
import re
list1 = ['0004567891', '9145873210', '7865432901',
         '9076543201']
For value in list1:
    if re.match (r'[0-9]{3}[0-9]{7}',
    value or len(value) == 10):
    print ("criteria matched for cell number")
    else:
      print ("criteria failed!")
```

Step 4 = import re module declare string and according and accordingly declare pattern replace the blank space with no-space. use scale() with 3 arguments and print the string with spaces.

Step 5 = import re modules declare a sequence use search method for finding subsequently use the group() with dot operator as search() gives memory coretion using group() it will show up the matched string.

Step 6 = import re module declare list with numbers.
use the conditional statement here we have used up the for condition statement. use if condition for checking first number is either 8 or 9 and next number are in range of 0 to 9 and check whether equal to 10 if ceritrea matches print cell number matches otherwise print failed.

Step 7 = import re module declare a string use the module with findall() for finding the values in the string and declare the same.

Step 8 = import re module declare ch host and domain name declare pattern for separating the host & domain name. use the find() and print the output respectively.

Step 9 = import re module enter a strin use pattern to display only two elements of the particular string use findall() declare two variables with initial value as zero use the if condition check whether condit satisfy add up the one else increm values and display the values subsequently.

criteria matched for cell number
criteria Failed!
criteria matched for cell number

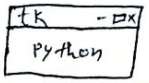
```
# vowels
import re
str1 = 'plant is life overall'
output = re.findall (r'\b[aeiouAEIOU]\w+', str1)
print (output)
>>> ['is', 'overall']
#host & domain
import re
seq = 'abc.tcsc@edu.com, xyz@gmail.com'
pattern = r'[\w.-]+ [\w.-]'
output = re.findall (pattern, seq)
print (output)
>>> ['abc.tcsc', 'edu.com', 'xyz', 'gmail.com']
# counting of first 2 letters:
import re
s = 'mr.a, ms.b, ms.c, mr.t'
p = r'[ms|mr]+'
o = re.findall (p, s).
print (o)
m = 0
f = 0
for v in o:
if (v == (ms')):
        f = f + 1
else m = m + 1
print ("No of males is:", m)
print ("No of females is:", f)
print ('ms', 'mr')
```

P.E.0

creation of parent window

```
from Tkinter import *
root = TK()
l = Label (root, text = "python")
l.pack()
root.mainloop()
```

output



2:

```
from Tkinter import *
root=TK()
l = Label (root, text = "python")
l.pack()
l1 = label (root, text = ("CS!", bg = "grey", fg = "black", font=
l1 . pack (side = LEFT, padx = 20)
l2 = label (root, text = "CS", bg = "light blue", fg = "black", font
l2 . pack (side = LEFT, pady = "30")
l3 = label (root, text = "CS! " bg = yelloco", fg = "black", font = "
l3 . pack (side = top, ipadx = 40)
```

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.

#2 :

step 1 = use the .tkinter library for importi the features of the text widget.

step 2 = create a variable from the text method and 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

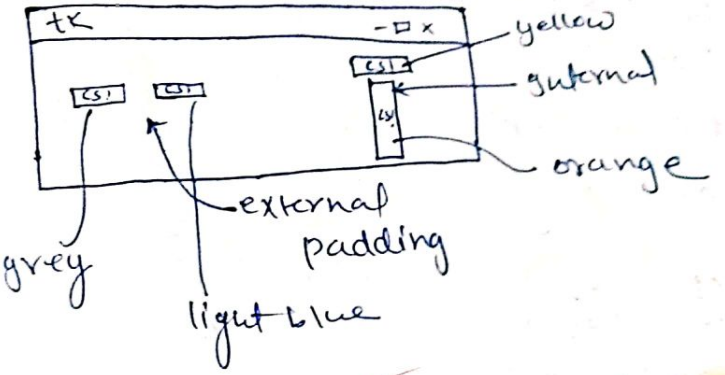
Step 4 : use the mainloop() for the triggering of the corresponding events

step 5 : Now repeat above steps with the label() which takes the following arguements.

1) Name of the parent window
2) text attribute which defines the string.
3) the background colour (bg)
4) the foreground fg and then use the pack(). with a relevant padding attributes.

```
l4 = Label ( root, text = "CS!", bg = "orange", fg = "black", font = ("Arial", 30))
l4 . pack ( side = TOP , ipady = 50 )
root. mainloop( )
```


output :



Radio button

```
from tkinter import *
root = TK()
root.geometry ("500x500")
def select():
    selection = "you just selected" + str(var.get())
    l1 = label (text = selection, bg = "white",
    bg = "green")
    l1.pack(side = TOP)
var = string var()
l1 = listbox()
l1.insert (1, "list1")
l2.insert (2, "list2")
l1.pack (anchor = N)
r1 = Radio Button (root, text = "option 1"
    variable = var, value = "option 1",
    command = select)
r2.pack (anchor = N)
r2 = Radio Button (root, text = "option 2",
    variable = var value = "option 2",
    command = Select)
r2 = pack (anchor = N)
root.mainloop()
```

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 the 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 parentwindow and define the option with control variable.

Step 5 = use the listbox() and insert options on the parentwindow along with the pack() with specifying another attribute

Step 6 = Create an object from radio button which will take following arguments Parentwind object, text, variable which will tak the values_option no, 1, 2, 3 Variable argument, correspondion values and

driggen the function declared.

Step 7 = Now call the pack() for scales object so created and specify the arguments using anchor attribute

Step 8 = finally make use of the mainloop along with parent object.

#2

Step 1 = Import relevant methods from the tkinter library.

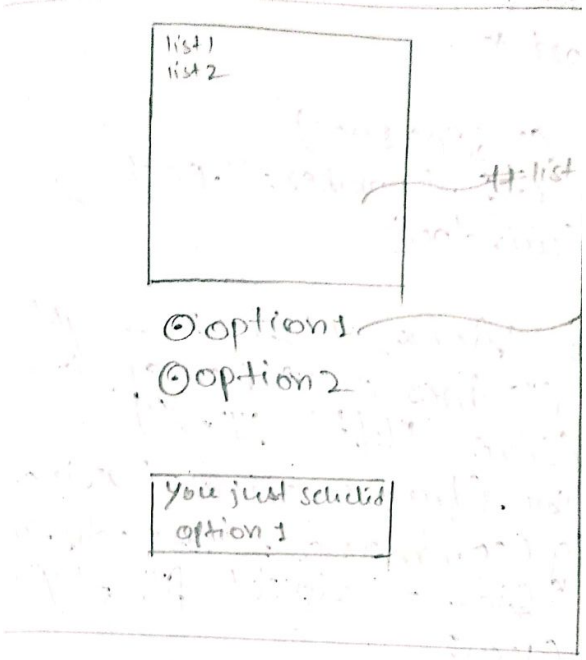
Step 2 = create a parent object corresponding to the parent window

Step 3 = use the geometry() for saying of the windows.

Step 4 = create an object and use the scrollbar()

Step 5 = use the pack() along with the scrollbar object side and fill attributy.

Step 6 = use the mainloop with the parent object.




```
scrollbar()
om tkinter import *
ot = Tk()
oot.geometry ("500x500")
= Scrollbar()
pack (side = "right", fill = "y
ot.mainloop()
```

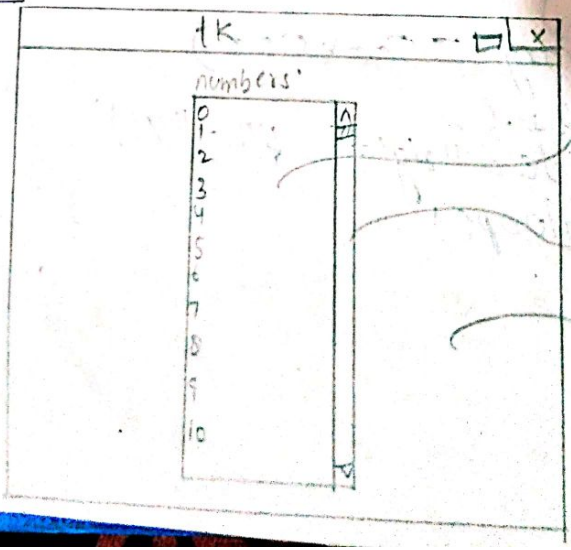
output :

geom

#3

```
#using frome widget
from tkinter import *
windows = TK()
windas.geometry ("600x500")
label1 (window, text = "nomber:").pack()
frame = Frame (windows)
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):
    list Nodes. insert (END, str (x))
windas.mainloop()
```

output :-



Q=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 (600x500) or any other suitable pixel values.

Step 4 = use the label widget along with the parent object created and subsequently use the pack method.

Step 5 = use the frame widget along with parent object created and use the pack method.

Step 6 = use the listbox method along with the attributes like width, height font. Do creat listbox methods object use pack() for th

Step 7 = use the scrollbar() with an objec attribute of vertical then configure with object created from the scroll and use pack().

... events using main

#14

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 objects from the methods and place it on to the parent window.

Step 4 = create another frame object named as the left frame and put it on the parent window on its LEFT side.

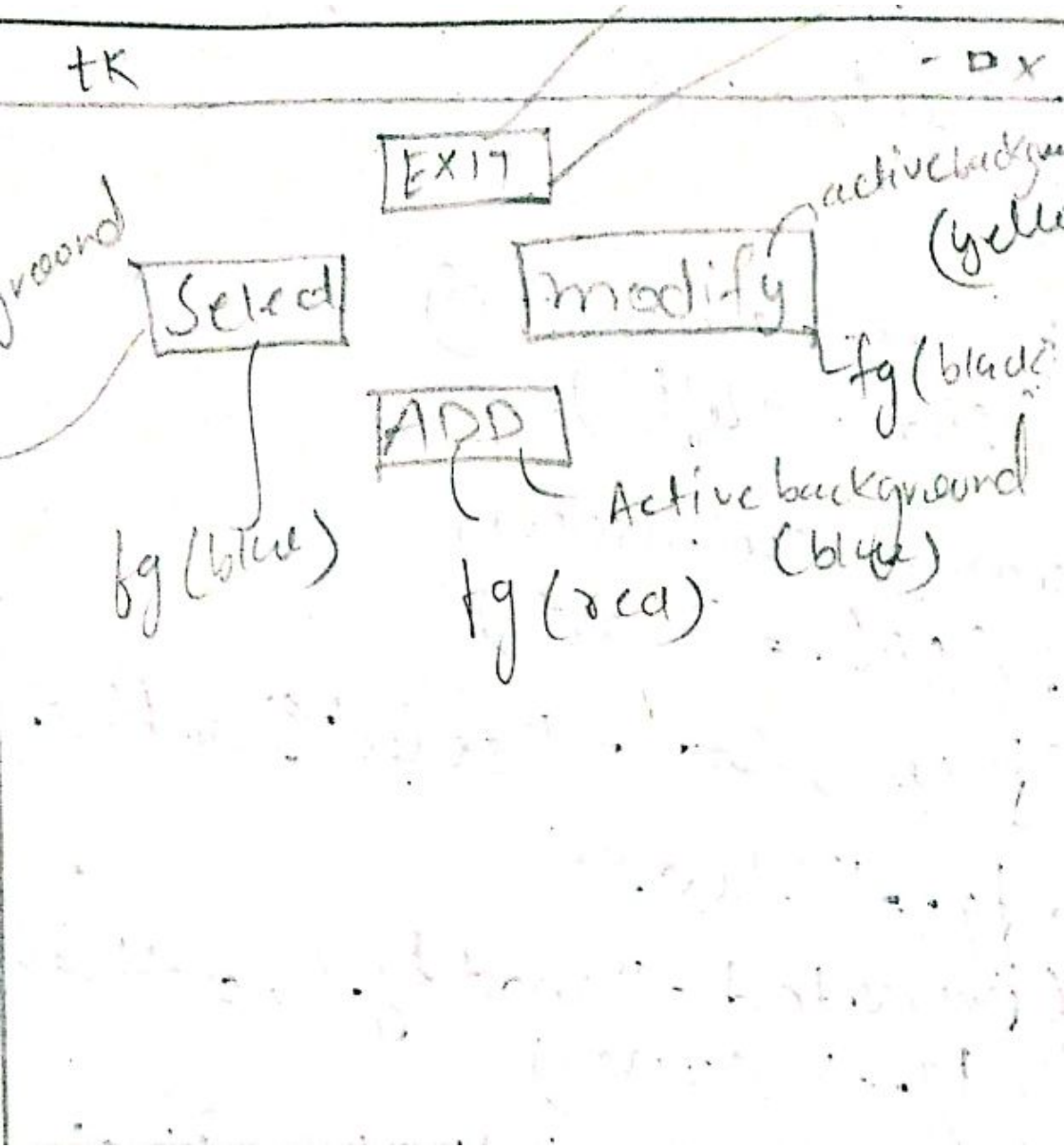
Step 5 = similarly define the RIGHT frame and subsequently define the button object placed onto the given frame with the attribute as text. active background and foreground

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

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

11-4

```
from tkinter import *
window = Tk()
window.geometry("600x500")
frame = Frame(window)
frame.pack()
leftframe = Frame(window)
leftframe.pack(side="left")
rightframe = frame.(window)
rightframe.pack(side="right")
b1 = Button(frame, text="select", activebackground
    ="red", fg="blue")
b2 = Button(frame text="modify", activebackground
    =yellow", fg="black")
b3 = Button(frame, text="ADD", activebackground
    ="blue", fg="red")
b4 = Button(frame, text="EXIT", activebackgro
    ="red", fg="green")
b1.pack(side="LEFT", padx=20)
b2.pack(side="right", padx=30)
b3.pack(side="bottom", pady=20)
b4.pack(side="top")
```



Step 08- Create another button object & place is on to the RIGHT frame & label the button as ADD.

Step 09:- Add another button & pute 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().

Aim: gui component : Message Box and multiple window

Steps 1 : import the relevant method from tkinter library.

Steps 2 : import tk messagebox

Steps 3 : Define a parent window object along with the parent window

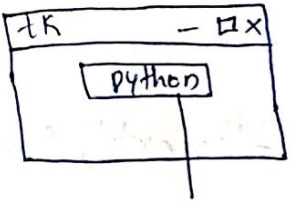
step 4 : define a function which will use tk massebox with showinfo method along with info window attribute.

Steps 5 : Declare a button with parent window object along with the commond attribute.

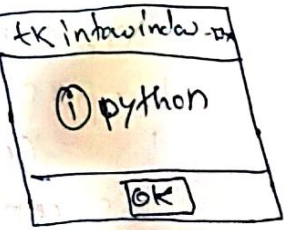
step 6 : Place the button widget onto the parent window and finally call mainloop() for triggering of the events called above.

```
# message box
from Tkinter import *
import tkmessage Box
root = TK()
def function():
    tk messageBox.showinfo("info window",
    python")
    b1 = Button (root, text = "python", commor
                 =function.
    b1 = pack()
    root.mainloop()
```

output:



click than this window is popup



Different 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
    In place of interest; In Golden gate
    Bridge In Lombeard Street In Chinatow
    In coit Tower")
    L.pack()
    b1 = Button(top, text = "next", comm
    = second)
    b1.pack(side= RIGHT)
    b2 = Button(top, text = "exit", comm
    = 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

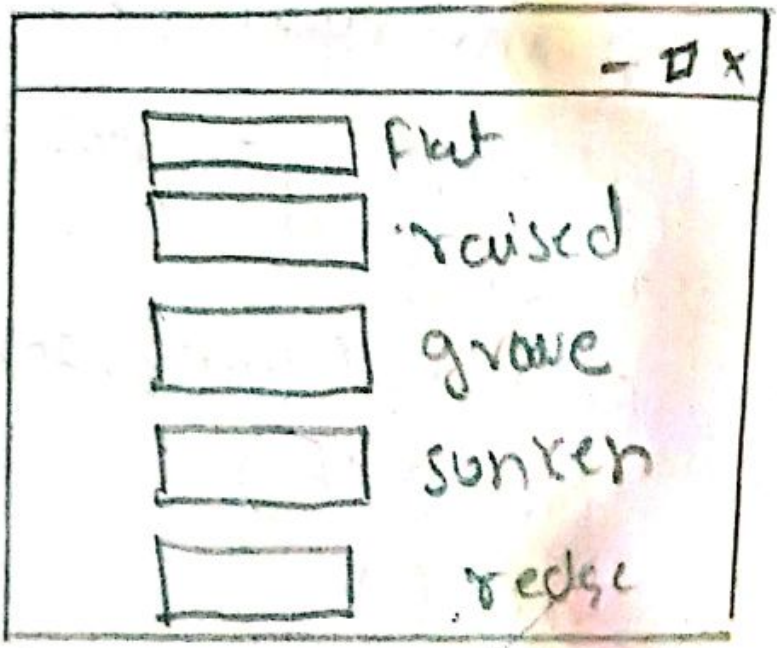
step 3: define a function main declare parent window object and use config(), title(), minisize(), Label() as well as button() and use pack() & mainloop() simultaneously

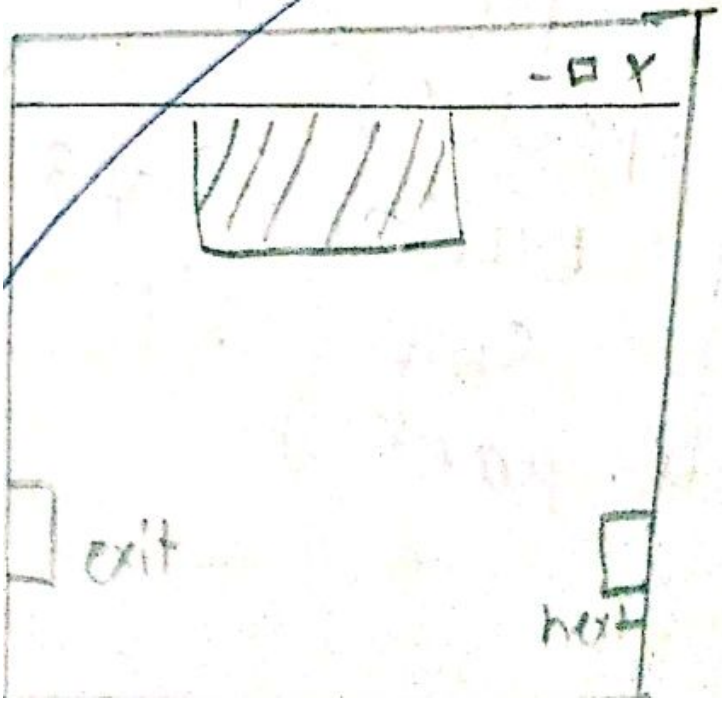
step 4: similary define the function second and use the attribute accordingly

step 5: Declare another function button along with parent object and declare button with attribute like FLAT RIDGE GROOVE, RAISE SUNKEN along with the relief widget.

step 6: finally called the mainloop for event driven programming

```
def second():
    top2 = TK()
    top2.config(bg="orange")
    top2.title("About US!")
    top2.minsize(300,300)
    L=Label(top2,text="Created by: Faizan salmani\n
    For more detail contact to our official account")
    L.pack()
    b3 = Button(top2, text='prev', command=main)
    b3.pack(side=LEFT)
    b2 = Button(top2,text="exit", command=terminat
    b2.pack(side=RIGHT)
    top2.mainloop()
def button():
    top3 = TK()
    top3.geometry("300x300")
    b1 = Button(top3, text="flat button", relief=FL
    b1.pack()
    b2 = Button(top3, text="groove button", relief
        GROOVE)
    b2.pack()
    b3 = Button(top3, text="raised button", re
        =RAISED)
    b3.pack()
    b4 = Button(top3,text="sunken button",
        =SUNKEN)
    b4.pack()
```





practical - 5(D)

Aim: Gui components: Grid method and photoim

Steps: Import relevant methods from the Tkinter library.

Step2: create parent window object and use the config method along with background colour attribute speci

Step3: define a fonction finish with th messagebox widget which will display a message i.e a warni message and subsequently termin the program.

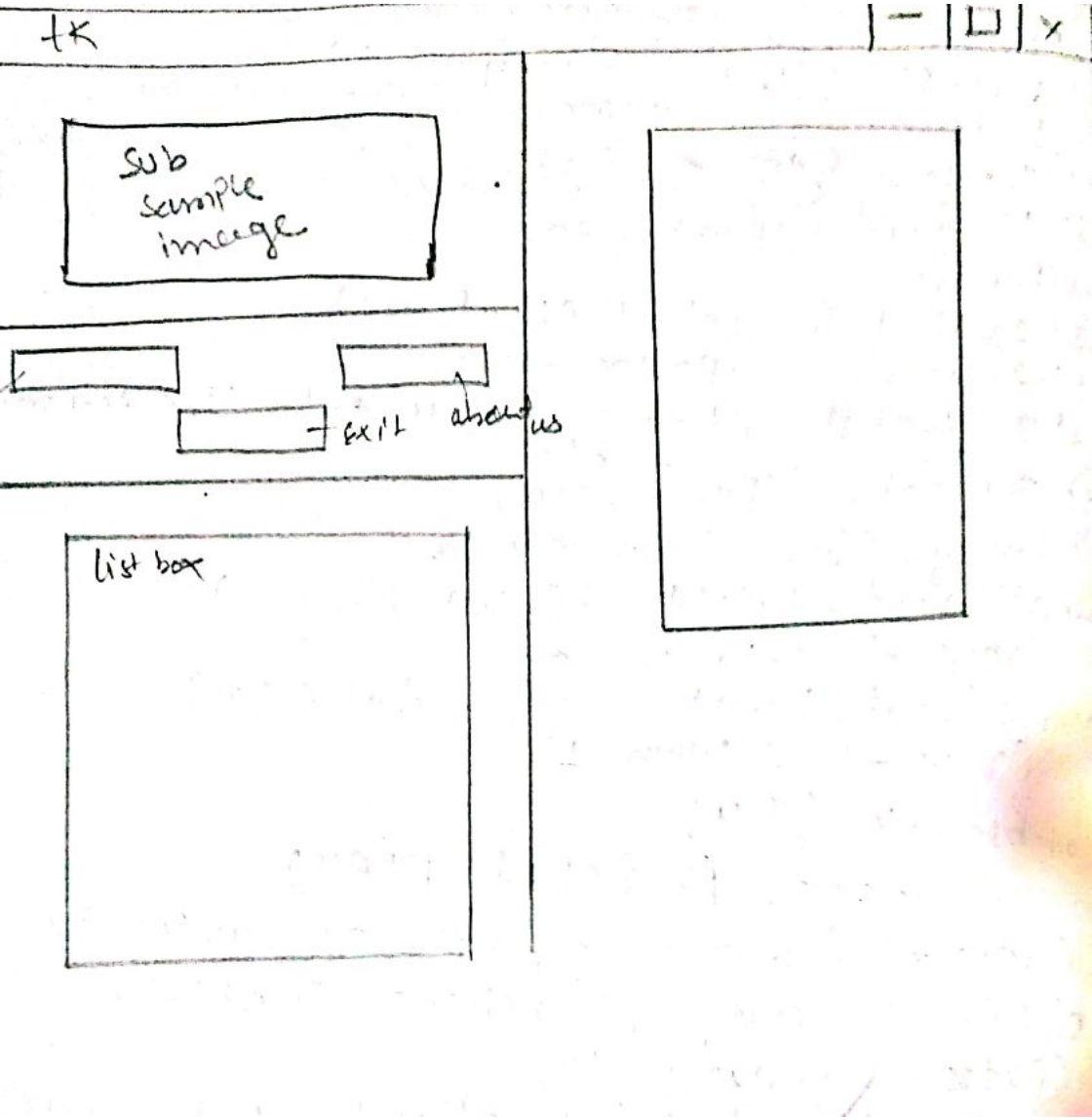
Step4: define a fonction info use a a listbox widget along with object of the same. use the listbox object along with insert method and insert the same and finally use the grid() with ipadx attribute

Steps: Define a fonction about us with label widget and text attribute and subsequently use the grid()

Step6: use photoimage widget with file and filename with gif attribute

Step7: create a frame object along with gif attribute. frame() along with parent window object height & width specified and subsequently use the grid() with row & column attribute specified.

```
root = Tk()
root.config(bg = "grey")
def finish():
    messagebox.askokcancel("warning", "this will end the program")
    quit()
def info():
    list1 = listbox()
    list1.insert(1," Co. Name : Apple")
    list1.insert(2," products : iphone")
    list1.insert(3," language : swift")
    list1.grid(padx = 30)
    list1.grid(ipadx = 30)
def aboutus():
    list2 = Label(text = "About us")
    list2.grid(ipadx = 30)
    list3 = Label(text = "steve jobs the creative mind,
    list3.grid(ipadx = 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 = p1, relief = SUNKEN)
l2.grid(padx = 25, pady = 10)
b1 = Button(f1, text = "information", relief = SUNKEN,
    command = info)
b1.grid(row1, column = 0)
b2 = Button(f1, text = "About us", relief = SUNKEN,
    command = aboutus)
b2.grid(row = 1, column = 2, padx = 5)
b3 = Button(f1, text = "EXIT", relief = RAISED,
    command = finish)
b3.grid(row = 2, column = 1, ipadx = 15)
mainloop()
```



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

step 9: Create another object of use the subsample (S14).

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

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

Aim: To perform program on traversing of windo

Algorithm:-

Step 1: import the relevant method from tKinter library.

Step 2: Define a function and create object of a given window by using the three methods namely config title minsize.

Step 3: Difine a Button object which will be placed on the current window to traverse and define another button which will be used to exit from the window and place it onto current window.

Step 4 :- Define another function which will use the quit method to terminate the program

Step 5:- Now create an object of main window and use various methods like config title geometry etc.

Source code:-

```
from tkinter import *
def main():
    root = TK()
    root.geometry ("500 x300")
    root.config (bg = "lightgreen")
    root.title ("GUI")
    root.minsize (height = 200, width = 300)
    B3 = Button (root, text = "Next", command = main)
    B3.grid (ipadx = 50, ipady = 40, padx = 20, pady = 30)
    B2 = Button (root, text = "Exit", command = term)
    B2.grid (ipadx, ipady = 40, padx = 20, pady = 30)
def term():
    quit()
tos = TK()
tos.geometry ("500 x500")
tos.config (bg = "red")
tos.title ("Trial")
tos.minsize (height = 200, width = 300)
B1 = Button (tos, text = "continue", command = main)
B1.grid (ipadx = 50, ipady = 40, padx = 20, pad = 30)
B2 = Button (tos, text = "Exit", command = term)
B2.grid (ipadx = 50, ipady = 40, padx = 20, pad
```

```
def main1():
    top = TK()
    top.geometry ("500x500")
    top.config(bg = "blue")
    top.title ("Browcrsing")
    top.minsize (height = 200, width = 300)
    B4 = Button (top, text = "mainpage", command=...)
    B4.grid (ipadx = 50, ipady = 40, padx = 20, pady=...)
    B2 = Button (top, text = "exit", command = ...)
    B2.grid (ipadx = 50, ipady = 40, padx = 20, pady...)
def main2():
    obj = TK()
    obj.geometry ("500x500")
    obj.config (bg = "red")
    obj.title ("python")
    obj.minsize (height = 200, width = 300)
mainloop()
```

Step 6: Difine two buttons which will be placed on the main window one to secure another to terminate the program.

Step 7 :- Difine another fuction which will carry various button placed on third window. Define two buttons respectively and use the grid method along with the button.

Step 8 :- finally call the mainloop method.

Spinbox

Aim: To implement use of spinbox widget.

Step 1: Creat an object from the tk method and subsequently create an object from the spinbox method

Step 2: make the object so created onto the parent window and trigger the corresponding events

Step 3: use the pack method to provide the direction using anchor method.

Step 4: Use the mainloop method to terminate

source code:

```
from tkinter import *
root = Tk()
s1 = spinbox(root, from_=0, to_=10)
s1.pack(anchor=s)
root.mainloop()
```

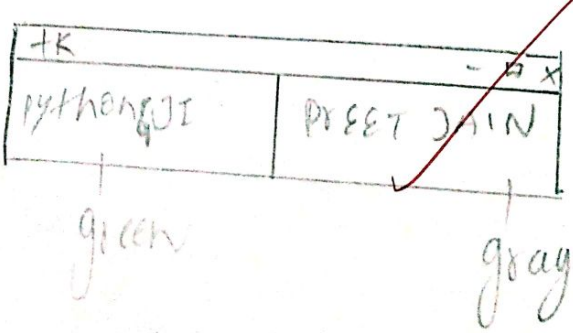
output

020

Source code:

```
from tkinter import *
root = Tk()
p = PanedWindow(bg="red")
p.pack(fill=BOTH, expand=1)
L1 = Label(p, text="PYthon GUI"), bg="gre
p.add(L1)
P1 = PanedWindow(p, orient=VERTICAL, bg=
p.add(P1)
L2 = Label(P1, text="PREET JAIN", bg="gra
P1.add(L2)
root.mainloop()
```

output:



anned window:

im: To implement use of panned window

ep1: create an object from panel window and use the pack method with the attribute fill and expand.

ep2: Create an object from the label method and put it onto the panel window with the text attribute and use the add method to embed the new object.

ep3: similarly create a second panel window object and it onto the 1st panel window with orientation specified.

p4: Now create another label object and place it onto the 2nd panel window object and add it onto the 2nd pane.

p5: Now use the mainloop method to terminate.

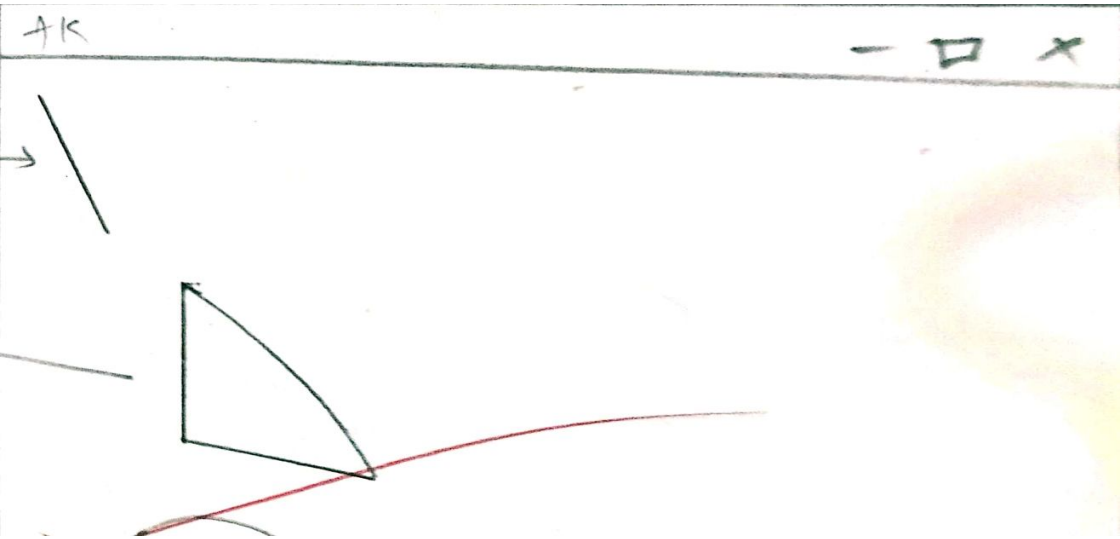
canvas widget:

Aim: use of canvas ~~method~~ widget.

Step 1: Use the tkinter method and create an object from the canvas method and use the attribute height weight, bg colour and the parent window object.

Step 2: Use the method create oval, create line and create arc along with the canvas object so created and use the co-ordinate value. Also use the fill attribute to assign various colour.

Step 3: Now call the pack method and mainloop method.



```
import dbms
db = dbm.open("Database", flag="c")
if db["www.it"] != None:
    print("good")
else:
    print("Not good")
```

output:

good.

Aim:- to make use of statement of database library.

Step 1 = import db. library and use the open method for creating the data base by specify name of the data base along with the corresponding flag.

Step-2 = use the objects the accessing to given used size and the corresponding regulation for used the size.

Step-3 = check whether the given URL address with the regular of the pages is not equal to none then the display the message from URL address else not formed.

Aim: To make use of Revision statements of data

S-1 = import the corresponding library taking house connection.

Step2 = Now create connection objects using sqlite library and connecting method for create the New database.

step 3 = Now create the cursor object using cursor method from the connection objects create in steps.

Step 4 = Now use the executing method for creating the tables with the column name and respective datatype.

Step 5 = Now with cursor object use insert statment for giving the values co-ordinating the diffent field considering the datatypes.

step 6 = use the commit method to complete the transtion the connection object.

program :-

```
import os, sqlite3
connection = sqlite3.connect("studentdb")
c1 = connection.cursor()
c1.execute('create table student(name,dob)')
c1.execute('insert into student values("preet",
                                1840, 26-07-2000)')
c1.execute('insert into student values("faiz",
                                1850, 15-05-1999)')
c1.execute('insert into student values("jain",
                                1860, 16-6-1998)')
connection.commit()
c1.execute('select * from student')
c1.fetchall()
c1.execute('Drop table student')
```

output :

```
[('preet', 1840, 26-07-2000), ('faiz', 1850,
15-05-1999) ('jain', 1860, 16-6-1998)]
```

step 7 = Use the execute statement along with the cursor objects for executing the values the database using selecting from when close.

step 8 = finally use the fetchall method for displaying the values for the tables using the cursor objects.

step 9 = use the execute method and the relevant tables syntax for terminating the database finally use the close method.

Project-1

Travel Management System

```
from tkinter import*
from tkinter import ttk
import random
import time;
import datetime
import tkinter.messagebox
class Travel:
        def __init__(self,root):
                self.root=root
                self.root.title("Travel Management Systems")
                self.root.geometry("1350x750+0+0")
                self.root.configure(background='black')


                DateofOrder=StringVar()
                DateofOrder.set(time.strftime("%d/%m/%y"))
                Receipt_Ref=StringVar()
                PaidTax=StringVar()
                SubTotal=StringVar()
                TotalCost=StringVar()


                Var1=IntVar()
                Var2=IntVar()
                Var1=IntVar()
                Var3=IntVar()
```

```
Var5=IntVar()
Var6=IntVar()
Var7=IntVar()
Var8=IntVar()
Var9=IntVar()
Var10=IntVar()
Var11=StringVar()
Var12=StringVar()
Var13=StringVar()

Firstname=StringVar()
Surname=StringVar()
PostCode=StringVar()
Address=StringVar()
Telephone=StringVar()
Mobile=StringVar()
Email=StringVar()



AirportTax=StringVar()
Mile=StringVar()
Travel_Ins=StringVar()
Luggage=StringVar()


Standard=StringVar()
```

```
        Economy=StringVar()
        FirstClass=StringVar()

        AirportTax.set("0")
        Mile.set("0")
        Travel_Ins.set("0")
        Luggage.set("0")

        Standard.set("0")
        Economy.set("0")
        FirstClass.set("0")

#=============================================================================

        MainFrame=Frame(self.root)
        MainFrame.grid()

        Tops=Frame(MainFrame,bd=20,width=1350,relief=RIDGE)
        Tops.pack(side=TOP)

        self.lblTitle=Label(Tops, font=('arial',70,'bold'),text="Travel Management Systems")
        self.lblTitle.grid()

#=============================================================================

CustomerDetailsFrame=Frame(MainFrame,width=1350,height=500,bd=5,pady=5,relief=RIDGE)
```

```
CustomerDetailsFrame.pack(side=BOTTOM)

ameDetails=Frame(CustomerDetailsFrame,width=880,height=400,bd=10,relief=RIDGE)
FrameDetails.pack(side=LEFT)

CustomerName=LabelFrame( FrameDetails,width=150,height=250,bd=5,
Name',relief=RIDGE)                font=('arial',12,'bold'),text='Customer
CustomerName.grid(row=0,column=0)

TravelFrame=LabelFrame( FrameDetails,width=300,height=250,bd=10,
font=('arial',12,'bold'),text='TravelDetails',relief=FLAT)
TravelFrame .grid(row=0,column=1)

Ticket_Frame=LabelFrame( FrameDetails,width=300,bd=10,height=200,
font=('arial',12,'bold'),text='Ticket_Frame',relief=RIDGE)
Ticket_Frame.grid(row=1,column=0)

Cost_Frame=LabelFrame( FrameDetails,width=150,bd=10,height=150,
font=('arial',12,'bold'),text='Cost_Frame',relief=FLAT)
```

```
Cost_Frame .grid(row=1,column=1)


#=============================================================================

Receipt_ButtonFrame=Frame(CustomerDetailsFrame,width=450,bd=10,height=400,relief=RIDGE)
Receipt_ButtonFrame.pack(side=RIGHT)

ReceiptFrame=LabelFrame(Receipt_ButtonFrame,width=350,bd=20,height=300,
font=('arial',12,'bold'),text='Receipt',relief=RIDGE)
ReceiptFrame.grid(row=0,column=0)

ButtonFrame=LabelFrame(Receipt_ButtonFrame,width=350,bd=20,height=300,
font=('arial',12,'bold'),text='Buttons',relief=RIDGE)
ButtonFrame.grid(row=1,column=0)

#============================CustomerName=================================

self.lblFirstname=Label(CustomerName,
font=('arial',14,'bold'),text="Firstname",bd=7)
self.lblFirstname.grid(row=0,column=0,sticky=W)

self.txtFirstname=Entry(CustomerName,
font=('arial',14,'bold'),text="Firstname",textvariable=Firstname,bd=7,
```

```
                                                insertwidth=2,justify=RIGHT)
        self.txtFirstname.grid(row=0,column=1)


    self.lblSurname=Label(CustomerName, font=('arial',14,'bold'),text="Surname",bd=7)
    self.lblSurname.grid(row=1,column=0,sticky=W)


            self.txtSurname=Entry(CustomerName,
font=('arial',14,'bold'),text="Surname",textvariable=Surname,bd=7,
                                                insertwidth=2,justify=RIGHT)
        self.txtSurname.grid(row=1,column=1)


    self.lblAddress=Label(CustomerName, font=('arial',14,'bold'),text="Address",bd=7)
    self.lblAddress.grid(row=2,column=0,sticky=W)


            self.txtAddress=Entry(CustomerName,
font=('arial',14,'bold'),text="Address",textvariable=Address,bd=7,
                                                insertwidth=2,justify=RIGHT)
        self.txtAddress.grid(row=2,column=1)


    self.lblPostCode=Label(CustomerName, font=('arial',14,'bold'),text="PostCode",bd=7)
    self.lblPostCode.grid(row=3,column=0,sticky=W)
```

```
self.txtPostCode=Entry(CustomerName,
font=('arial',14,'bold'),text="PostCode",textvariable=PostCode,bd=7,
insertwidth=2,justify=RIGHT)
self.txtPostCode.grid(row=3,column=1)

self.lblTelephone=Label(CustomerName,
font=('arial',14,'bold'),text="Telephone",bd=7)
self.lblTelephone.grid(row=4,column=0,sticky=W)

self.txtTelephone=Entry(CustomerName,
font=('arial',14,'bold'),text="Telephone",textvariable=Telephone,bd=7,
insertwidth=2,justify=RIGHT)
self.txtTelephone.grid(row=4,column=1)

self.lblEmail=Label(CustomerName, font=('arial',14,'bold'),text="Email",bd=7)
self.lblEmail.grid(row=5,column=0,sticky=W)

self.txtEmail=Entry(CustomerName,
font=('arial',14,'bold'),text="Email",textvariable=Email,bd=7,
insertwidth=2,justify=RIGHT)
self.txtEmail.grid(row=5,column=1)
```

```
        self.lblMobile=Label(CustomerName, font=('arial',14,'bold'),text="Mobile",bd=7)
        self.lblMobile.grid(row=6,column=0,sticky=W)


        self.txtMobile=Entry(CustomerName, font=('arial',14,'bold'),text="Mobile",bd=7,
                             insertwidth=2,justify=RIGHT)
        self.txtMobile.grid(row=6,column=1)


      #====================Flight
Information=====================================
        self.lblDeparture=Label(TravelFrame,font=('arial',14,'bold'),text="Departure",bd=7)
        self.lblDeparture.grid(row=0,column=0,sticky=W)


elf.cboDeparture=ttk.Combobox(TravelFrame,textvariable=Var11,state='readonly',font=('arial',20,'bold'
                              width=14)


        self.cboDeparture['value']=(' ','Healthrow','Lagos MM','Dew','Lutton')
        self.cboDeparture.current(0)
        self.cboDeparture.grid(row=0,column=1)


#=========================================================================
```

```
if __name__ =='__main__':
                    root=Tk()
                    application=Travel(root)
                    root.mainloop()
```