26/11/19

IS

PRACTICAL - 1

AIM: Demonstrate the use of different file accessing modes different attributes read methods.

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 then use read(), readline() and readlines() and store the output in variable and finally display contents of variable.

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

```
fileobj = open ("abc.txt", "w")
fileobj.write ("versions of Python" + "\n")
fileobj.write ("Python 2.7" "\n" "Python 3.6" "\n" "Python 3.7")
fileobj.close()
fileobj = open ("abc.txt", "r")
str1 = fileobj.read()
print ("The output of read method", str1)
fileobj.close()
```

```
>>> The output of read method versions of Python
    \n Python 2.7 \n Python 3.6 \n Python 3.7
```

```
# readline()
fileobj = open ("abc.txt", "r")
str2 = fileobj.readline()
print f"The output of readline method", str2)
fileobj.close()
```

```
>>> The output of readline method, 'versions of Python'
```

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

```
>>> The output of readlines method ['versions of Python\n,
    'Python 2.7' \n' Python 3.6' \n 'Python 3.7']
```

```
# file attributes
a = fileobj.name
Print ("name of file (name attribute) :", a)
>>> (name of file (name attribute), 'abc.txt')
```

```
c = fileobj.mode
print ("filemode", c)
>>> ('filemode', 'r')
d = fileobj. softspace
print("softspace", d)
>>> ('softspace', 0)
```

```
# wt mode
fileobj = open("abc.txt","wt")
fileobj.write ("messi")
fileobj.close()
```

```
# write mode
fileobj = open ("abc.txt","w")
fileobj.write [redacted] ("Python")
fileobj.close()
```

```
# read mode
fileobj = open ("abc.txt", "r")
s = fileobj .read ()
print ("output of read
        mode",s)
>>> ('output of read mode),
    'messi')
```

```
#rt mode
fileobj = open ("abc.txt", "rt")
s1 = fileobj. read (5)
print ("output of rt", s1)
fileobj.close()
>>> (output of rt , 'messi')
```

```
# append mode
fileobj = open ("abc.txt", "a")
[redacted]
fileobj. write (" Football ")
fileobj. close()
fileobj = open ("abc.txt", "r")
s3 = fileobj .read()
print ("output of append ", s3)
fileobj .close ()
>>> (output of append , " messi Football")
```

Step 4 : Now open the fileobj in write mode write some another content close subsequently Then again open the fileobj in 'wt' mode that is the update mode and write contents.

Step 5 : Open fileobj in read mode display the update written content and close. Open again in 'r+' mode with parameter passe and display the output subsequently.

Step 6 : Now open file obj in append mode open write method, write content, close the fileobj again open the fileobj in read mod and display the 'appending' output.

Step 7 : Open the file obj in read mode declare a variable and perform fileobject tell method and store the output consequen in variable.

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

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

tell()

```
fileobj = open("abc.txt", "r")
pos = fileobj.tell()
print ("tell()", pos)
fileobj.close
>>> ('tell() :', 0L)
```

seek

```
fileobj = open("abc.txt","r")
st = fileobj.seek(0,0)
print("seek(0,0) is ", st)
fileobj.close
>>> seek(0,0) is , None
fileobj = open ("abc.txt","r")
str1 = fileobj.seek (0,1)
print ("seek (0,1) is ", st1)
fileobj.close()
>>> (seek (0,1) is : , None)
file object = open("abc.txt","r")
stat = fileobj.readlines()
print ("output : "stat)
for line in stat:
        print (len(line))

fileobj.close()
>>> output : ([Messi , Date Football])
```

```
# for loop
mytuple1 = ("Karan", "Jay", "Varun")
for x in mytuple1:
    print(x)
```

```
>>> karan
    Jay
    Varun
```

```
# iter() and next()
mytuple1 = ("banana", "orange", "apple")
myiter1 = iter(mytuple)
print(next(myiter1))
myiter2 = iter(mytuple)
print(next(myiter1))
myiter3 = iter(mytuple)
print(next(myitel))
```

```
>>> banana
    orange
    apple
```

```
# Square & cube.
def square(x):
    y = x * x
    return y
def cube(x):
    z = x * x * x
    return z
funct1 = [square, cube]
```

PRACTICAL - 2

AIM : Iterators

Step1 : Create a ~~typ~~ tuple with elements that we need to iterate using the iter and next method. The number of time we use the iter and next method will get the next iterating element in the tuple. Display the same.

Step 2 : The similar output can be obtained by using for conditional statement. An iterable ~~value~~ variable is to be declared in for loop which will iterate.

Step 3 : Define a function name square with a parameter which will obtain output of square value of the given number. In similar way, declare to get value raise to 3, and return the same.

Step 4 : Call the declared function using functi call.

Step 5: Using for ~~condi~~ conditional statement, specifying the range use the list type casting with map method declare a 'lambda' i.e. annonymus function and print same.

Step 6 : Declare a listnum variable and declare some elements then use the map method with help of lambda defination give two argument display the output.

Step 7 : Define a function with a parameter then using conditional statements do check whether the number is even and odd and return respectively.

Step 8 : Define a class and within that define the iter() method which will initialize the first element within the container object.

Step 9 : Now use the next() and define the logic for displaying odd value.

```
for r in range(5):
    value = list(map(lambda x: x(r), funct1))
    print(value)
```

```
>>> [0, 0]
    [1, 1]
    [4, 8]
    [9, 27]
    [16, 64]
```

```
# map()
listnum = [0,4,5,7,9,11,13,15,20,19,25]
listnum = list(map(lambda x: x*1.5, listnum))
print(listnum)
def even(x):
    if (x%2 == 0):
        return "EVEN"
    else:
        return "ODD"
list(map(even, listnum))
# odd numbers
class odd:
    def __iter__(self):
        self.num=1
        return self
    def __next__(self):
        num.self = num
        num.self += 2
    def __next__(self):
        num = self.num
        self.num += 2
        return num
```

10/12

```
myobj = odd()
myiter = iter(myobj)
x = int(input("Enter a number"))
for i in myiter:
    if(i<x):
        print(i)
```

```
>>> Enter a number: 21
1
3
5
7
9
11
13
15
17
19
```

Step 10 : Define an object of a class.

Step 11 : Accept an number from the user till which ~~use~~ we want to display the odd number.

PRACTICAL -3

TOPIC : Exceptions

Step1: In the try block open an file in the open mode with write mode write some contents in file.

Step2 : In Except (IOError) block , use the error in IO Error use the appropriate message to display.

Step 3 : Else display the operation is sucessful!

Step 4 : In try block , accept an input from user.

Step 5 : In except block use valuerror and print the same message.

Step 6 : Else display the operation is sucessful.

Step 7 : Accept an integer value from the user . In the try use the division method.

IO Error :

```
try:
    fo = open ("abc.txt", "w")
    fo.write ("Python is an interpreted language")
except IO Error:
    print ("Enter appropriate mode for file operation!")
else:
    print ("Operation is sucessful")
```

>>> operation is sucessful !

R output :

>>> Enter appropriate mode for file operation !

Value Error :

```
try:
    x = int (input ("Enter a statement"))
except value error:
    print ("ARITHMETIC ERROR")
else:
    print ("operation is successful")
```

>>> Enter a statement : abc
ARITHMETIC ERROR !

>>> Enter a statement : 123
Operation is successful !

Type error, zero division error

```
a=
input ("Enter:")
try:
  prin(a/b)
except TypeError:
  print("Incompatible value")
except zero division error:
  print("Denominator is zero so operation
        Cannot be performed")
```

```
>>> Enter : t
    Incompatible value
>>> Enter : 2
    0.5
>>> Enter : 0
    Denominator is zero so operation
    cannot be performed.
```

Multiline Exception.

```
a,b =1,0
try:
   print(1/0)
   print(1/0+10)
except (TypeError, zeroDivisionError):
   print("Invalid Input")
>>> 1.0
>>> 1010
>>> Invalid Input!
```

Step 8 : For the exception to the raised use the except keyword (and) i.e. typerror print incompatible values.

Step 9 : Use except with error of zero division error, and print the message according that is if entered number is zero not able to perform operation.

Step 10 : Declare static variables and values

Step 11 : For Multiline exception, use the error types by seperating them with a comma.

Step 12 : Use try block open a file in write mode and subsequently enter values in the file.

CS:

Step 13: Use the IO Error and display appropriate message.

Step 14: Define a function with empty list and calculate the length of the list.

Step 15: Define another function Y, initialise or declare some elements in list and calculate the length of the same and display the same.

Step 16: In try block accept input from the user and if the user enters character values raise an Error that is saying up. Enter integer values.

#using except keyword.

```
try:
    a = open("abc.txt", "w")
    a.write("DBMS")
except IOError:
    print ("Error!")
else:
    print sucessful.
def x():
    l = []
    print (len(l))
def Y():
    li = [2,4,4,1]
    print (len(li))
print(x())
print(Y())
```

Output: Sucessful
0
None
4
None

raise keyword:

```
try:
    a = int(input ("Enter Number"))
    raise value error
    except value error:
    print("Enter integer value")
```

>>> Enter : xyz

Enter integr values!

=match()

```
import re
pattern = r"FYCS"
sequence = "FYCS represents Computer Science"
if re.match(pattern, sequence):
        print("matched pattern found!")
else:
        print("NOT FOUND!")
```

>>> matched pattern found

numerical values

```
import re
pattern = r'\d+'
String = 'hello123, howdy789, 45 howru'
Output = re.findall(pattern, string)
print(output)
```

>>> ['123','789','45']

split()

```
import re
pattern = r'\d+'
string = 'hello123, howdy789, 45howru'
Output = re.split(pattern, string)
print(output)
```

>>> ['hello', 'howdy', 'howru']

29/12/19

PRACTICAL - 4

TOPIC : Regular Expression

Step1: Import re module declare pattern and declare sequence use match method with declare arguments if arguments matched than print the same Otherwise print pattern NOT FOUND!

Step2: Import re module. declare pattern with Literal and meta character. Declare string value use the findall() with arguments and print the Sanae.

Step 3: Import remodule declare pattern with meta character use the split() and ~~prit~~ print the output.

Step4: import re module declare String and accordingly declare pattern replace the blank space with no_space. use sub() with 3 arguments and print the string without spaces.

Step5: import re module declare a sequence use search method for finding subsequently use the group() with dot operator.

Step6: import re module declare list with ~~numbe~~ use the conditional statements. if criteria matches print cell number matches otherwise print failed.

Step7: import re module declare a string use the module with findall() for finding the vowels in the string and declare the same.

```
# no_space :
import re
string = 'abc def ghi'
pattern = r'\s+'
replace = ''
v1 = re.sub(pattern, replace, string)
print(v1)
```

>>> abcdefghi

```
# group()
import re
sequence = 'Python is an intended language'
v = re.search ('\Python', sequence)
print(v)
v1 = v.group()
print(v1)
```

>>> <_sre.SRE_Match object at 0x0281DF00>
Python

```
# Verifying the given set of phone numbers
import re
list1 = ['8004567891', '9145673210', '7865932981',
         '9876543201']
for value in list1:
if re.match(r'[8-9]{1}[0-9]{9}'),
value or len(value)==10):
    print("Criteria matched!")
else:
    print("criteria failed")
```

>>> criteria matched for cell number
Criteria failed!

```
# vowels
import re
Str1 = 'plants are life'
Output = re.findall (r'\b[aeiouAEIOU]\w+', str1)
print(output)
```

>>> ['are']

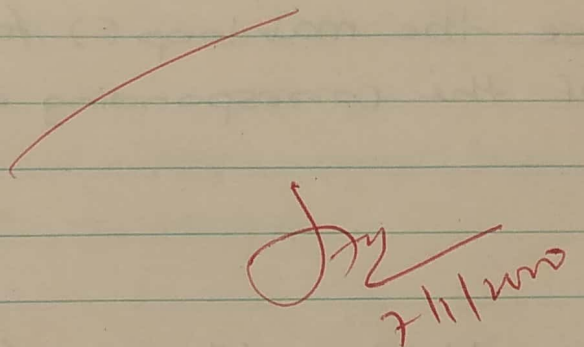
```
#host & domain
import re
seq = 'abc.tcsec@edu.com, xyz@gmail.com'
pattern = r'[\w\.-]+[\w\.-]'
output= re.findall (pattern, seq)
print(output)
```

>>> ['abc.tcse', '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)
```

Step 8: import re module declare a string use the module with findall() for finding host name and print the output respectively.

Step 9: import re module enter a string use pattern to display only two elements of the particular string. use findall declare the variables with intial value as zero use for condition. And display the values subsequently.



>>> ['mr', 'ms', 'ms', 'mr']
(No. of males is : 2)

PRACTICAL 5

Topic : GUI

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 widget label and use text method.

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

2)

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

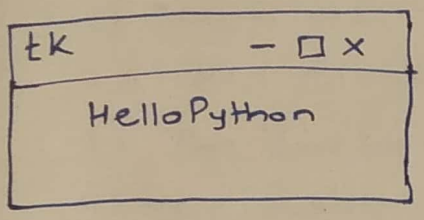
Step 2 : Create a variable from the text method and position it on the parent window.

Step 3 : Use the pack() along with the object created from the text() and use,
1) side = LEFT , padx = 20 2) side = LEFT, pady = 30
3) side = TOP , ipadx = 40 4) side = TOP, ipady = 50

1)

```
from tkinter import *
root =Tk()
l = label (root, text = "Hello Python")
l.pack ()
root . mainloop()
```

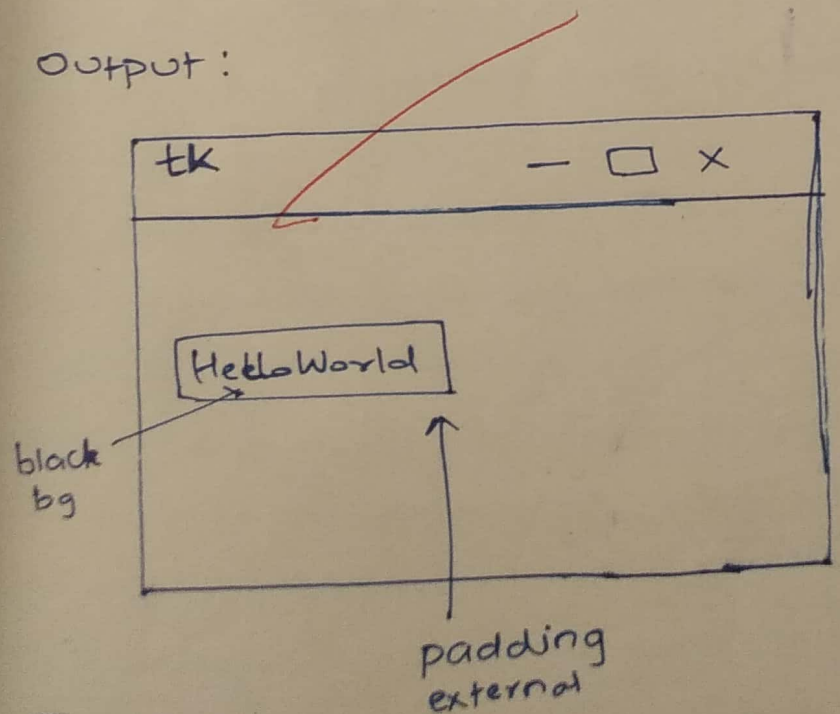
output:



2)

```
from tkinter import *
root = Tk()
L= label (root, text ="Hello Python")
l.pack()
l1= Label(root, text = "Hello World", bg= "black", fg ="wh
l1.pack(side = LEFT, padx =20)
root.mainloop()
```

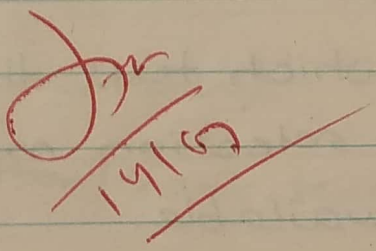
output:



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

Step 5: Now repeat above steps with the label() which takes the arguments

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



PRACTICAL - 5(2)

AIM :- RadioButton and Scrollbar using tkinter

1) Radiobutton

Algorithm :-

Step 1 : Use the tkinter module to import the relevant methods.

Step 2 : Define a function which tells the user about the given selection made from multiple option available.

Step 3 : Use the config() along with the label object and call the variable as an argument within the method.

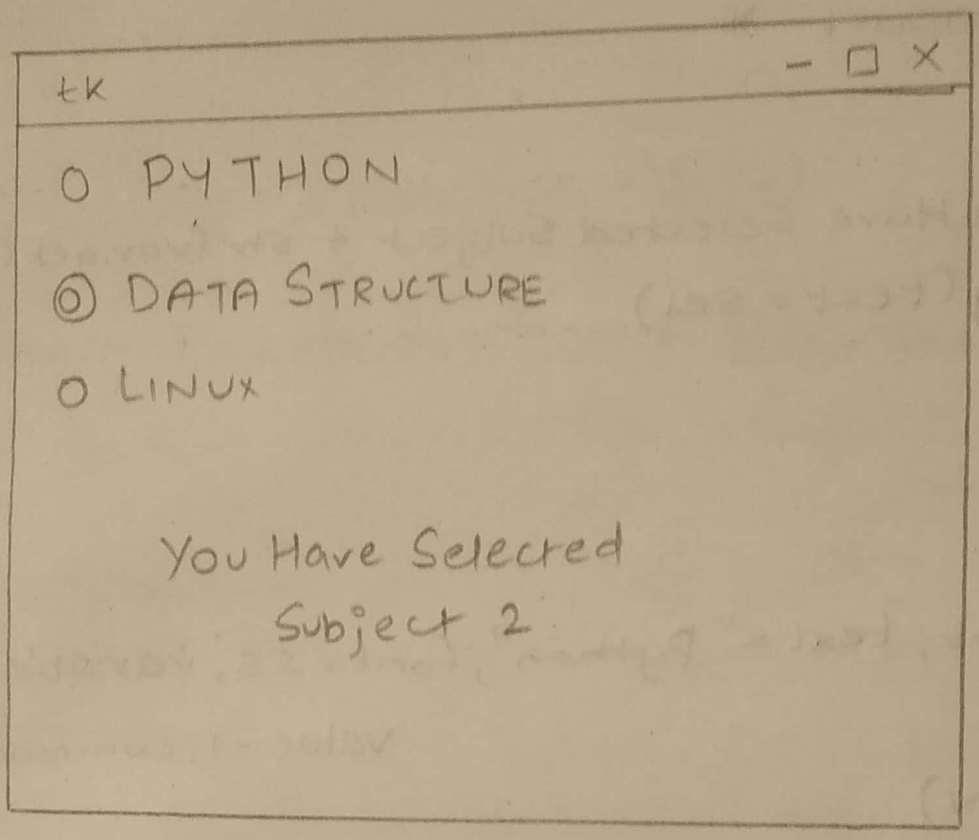
Step 4 : Now define the parent window and define the option using the control variable.

Step 5 : Create object from the Radiobutton() which will take the following arguments;
i) Positioning on parent window

Source Code:-

```
from tkinter import *
def sel():
    sel = "\n You Have Selected Subject" + str(var.get())
    label.config(text = sel)

root = Tk()
var = IntVar()

R = Radiobutton (root, text = "Python", font = '25', variable = var,
                 value = 1, command = sel)
R.pack (anchor = W)

R1 = Radiobutton (root, text = "Data Structure", font = '25',
                  variable = var, value = 2, command = sel)
R1.pack(anchor = W)

R2 = Radiobutton(root, text = "Linux", font = '25', variable = var
                 value = 3, command = sel)
R2.pack(anchor = W)

label = Label (root)
label.pack()
root.mainloop()
```

Output :-



ii) Defining the text variable which will take the values option 1, 2, 3, 4, ...

iii) Define the variable argument. The corresponding value and trigger the given function.

Step 6: For corresponding Radiobutton object, call the pack method and specify the argument as an anchor attribute.

Step 7: Now define the Label object from corresponding method and place on parent window. Selbsequently vse the pack method for this widget and finally call the mainloop method.

2) Scrollbar

Step 1: Import relevant methods from tkinter library.

Step 2: Create an object corresponding to the parent window and create an object from scrollbar method and place it onto the parent window.

Step 3: Create an object from the text method and place it onto the parent window with height and width specified.

Step 4: Use the pack method along with the object of the scrollbar method and use the argument as side and fill.

Step 5: Now use the text object along with the pack method and again use the side and the fill attribute

Step 6: Now use the config() method along with the object on the scrollbar and use the command attribute.

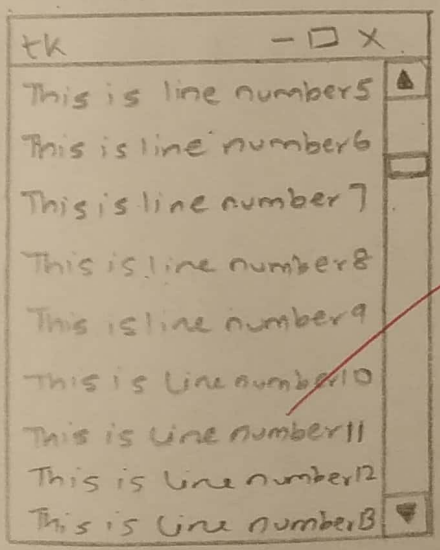
Step 7: Similarly use the config() method along with text object and use yscrollcommand argument.

Step 8: Now define the textual information in terms of paragraph and use the insert() method with two arguments and call the mainloop method.

Source Code :-

```
from tkinter import *
root= Tk()
scrollbar = Scrollbar(root)
scrollbar.pack( side = RIGHT, fill=Y)

mylist = Listbox(root, yscrollcommand= scrollbar)
for line in range(100):
    mylist.insert(END, "This is line number " + str(var())

mylist.pack( side = LEFT, fill = BOTH)
scrollbar.config( command = mylist.yview)
root.mainloop()
```

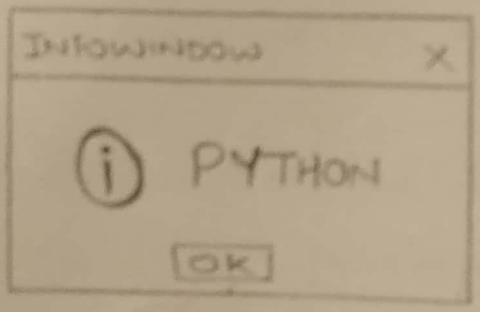
Output:-



21/01/22

Source Code:-

```
from tkinter import *
import tkMessageBox
root=Tk()
def fun():
    tkMessageBox.showinfo("InfoWindow","Python")
bl=Button(root,text="title",command=fun)
bl.pack()
mainloop()
```



PRACTICAL 5(3)

AIM: Messagebox, Relief, Traversing Window

1) Message Box Method.

Algorithm:-

Step 1: Input the relevant method from Tkinter library.

Step 2: Define a function and use the message box along with the different methods available which contains one or more argument.

Step 3: Thus different options available are, show info(), show warning(), showerror(), askyesno(), ask question(), askokcancel().

Step 4: Create an object from the button method and place on the parent window with the title of button specified and corresponding event called for triggering.

Step 5: Use the pack method to display the button widge and finally use the mainloop method.

Step 6: If the user wants to hide the parent window and only the info window should be visible corresponding to six options given above the withdraw() method is used.

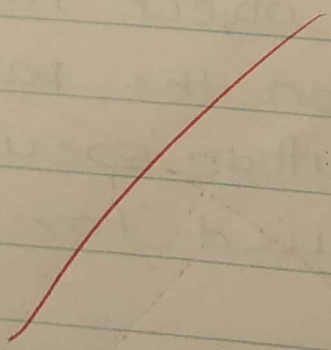
* Relief Style :

Step1 : Use the button object with the following attributes.

1) Parent Window
2) Text Attribute
3) Relief.

Step 2 : Use the corresponding pack method for the respective button objects and trigger the corresponding event.

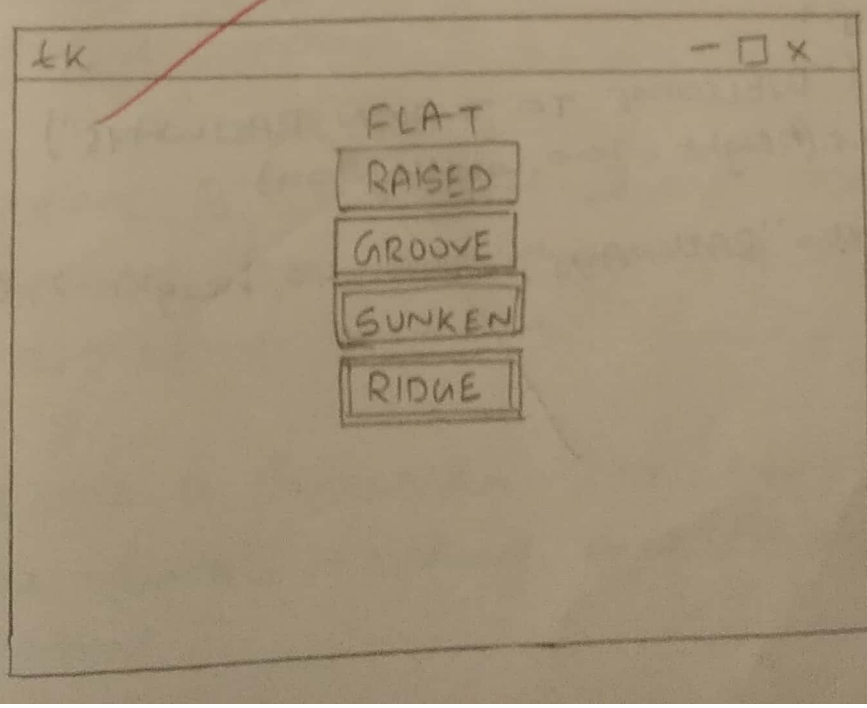
Step3 : Finally use the mainloop method.



Some Code :-

```
from tkinter import *
root = Tk()

b1 = Button (root, text="FLAT", relief = FLAT)
b1.pack()

b2 = Button (root, text="RAISED", relief = RAISED)
b2.pack()

b3 = Button (root, text="GROOVE", relief = GROOVE)
b3.pack()

b4 = Button(root, text="SUNKEN", relief = SUNKEN)
b4.pack()

b5 = Button (root, text="RIDGE", relief = RIDGE)
b5.pack()

mainloop()
```

Output :-



SourceCode:-

```
from tkinter import *
#AIRPORT
def air():
    root=Tk()
    root.config()
    root.title("WELCOME TO INTERNATIONAL AIRPORT")
    root.minsize(height=700,width=700)

wl=Tk()
l=Label(wl,text="\n TICKET RESERVATION SYSTEM")
l.pack()

b=Button(wl,text="AIRPORT",width=40,height=2,
                        command=air)
b.pack()

#RAILWAYS
def rail():
    root=Tk()
    root.config()
    root.title("WELCOME TO INDIAN RAILWAYS")
    root.minsize(height=700,width=700)
b=Button(wl,text="RAILWAYS",width=40,height=2,command=rail
b.pack()

mainloop()
```

* Traversing from one window to another and making use of geometry layout manager.

Algorithm:

Step1: Define a function and create a object of the given window by using the three method 1) config 2) title 3) minsize.

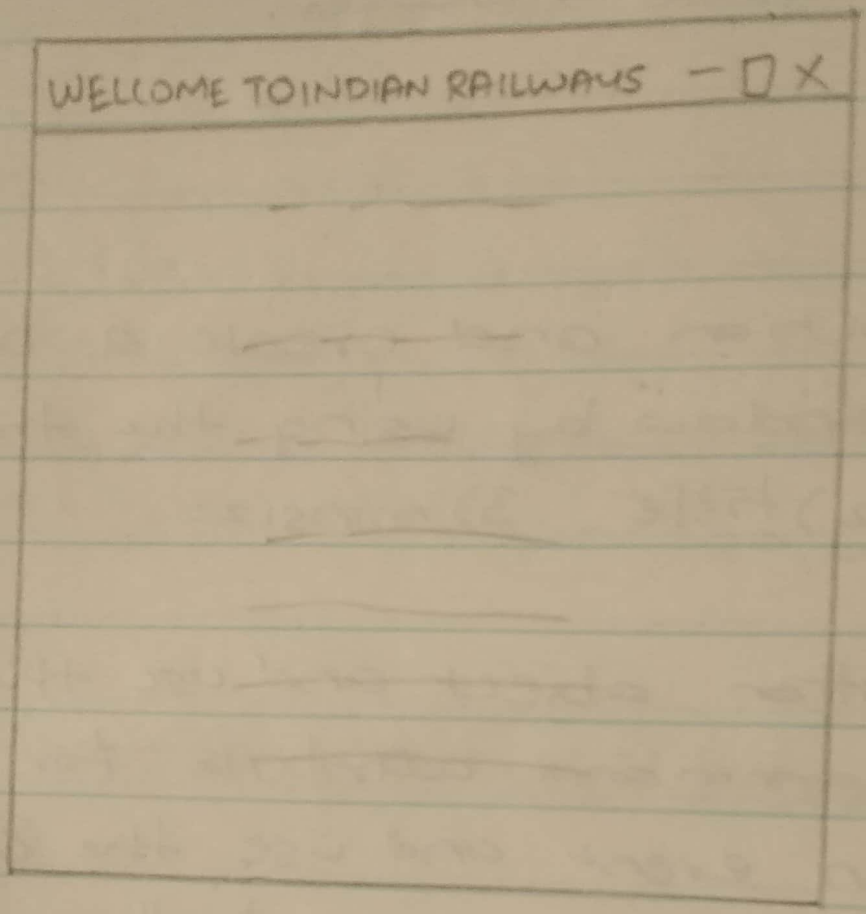
Step2: Create a Button object and use the text and the command attribute for triggering of the given event and use the grid method Similarly, create another button object which will allow the application to terminate

Step3: Define the second function Corresponding to the second window with the attribut for the window object and define one button object which will shift focus onto third window.

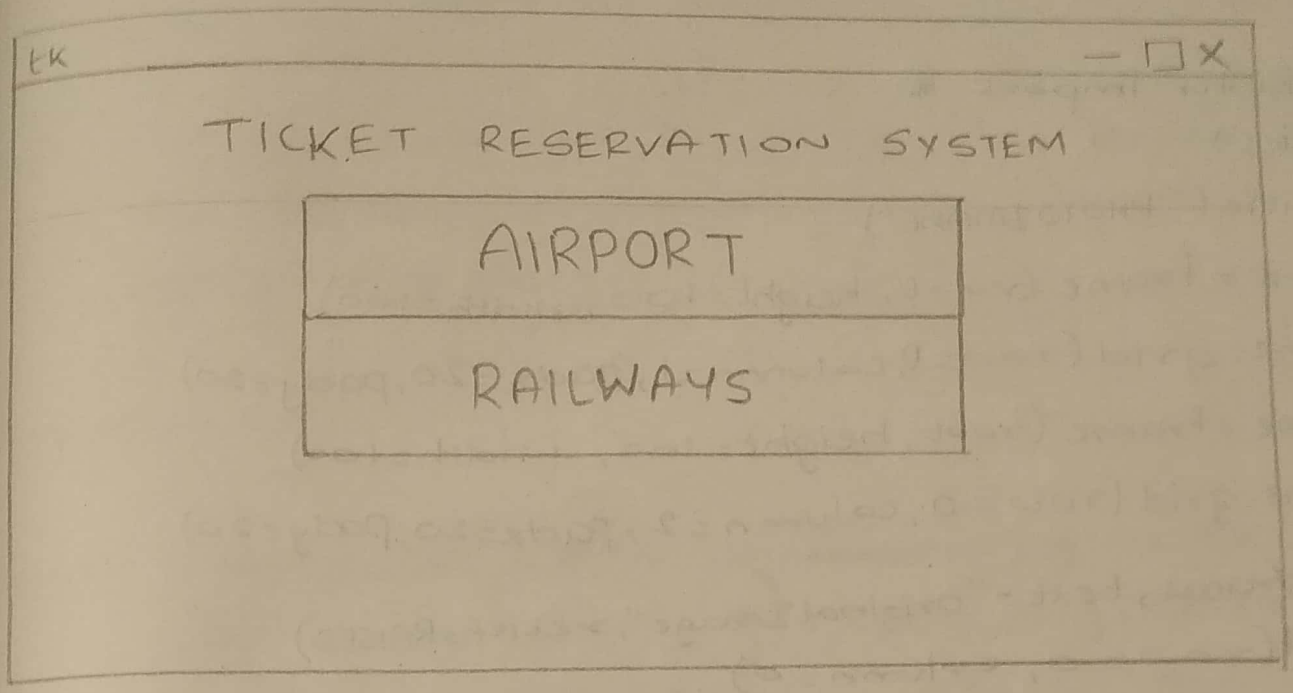
Step4: Create 3rd window Object and create two buttons for moving onto first window for restarting process.

Step5: Define a function for termination and ca the quit() method and finally call the mainlo method.

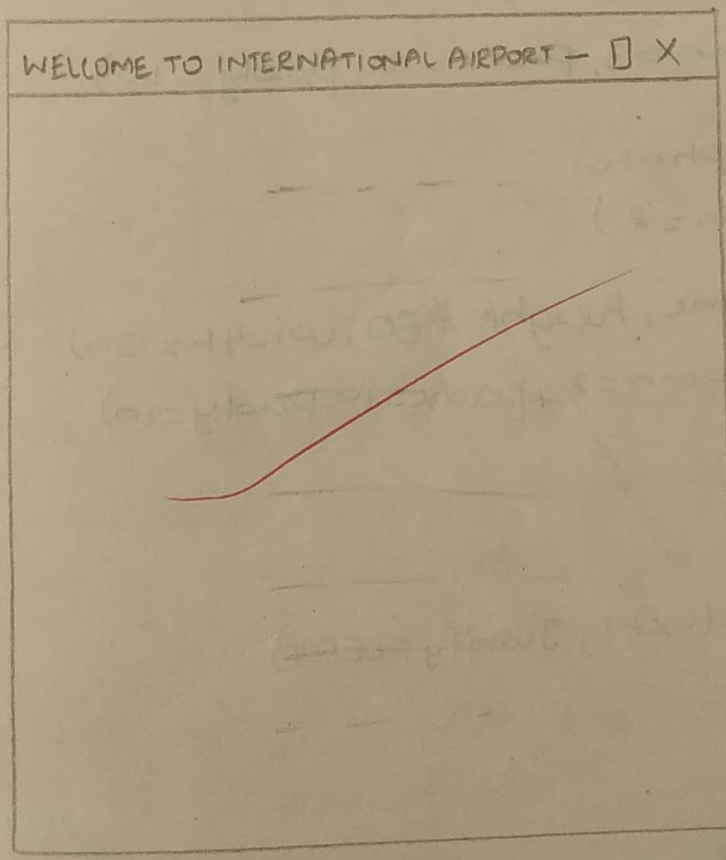
11 on CLICKING RAILWAYS



#WINDOW 1



#ONCLICKING AIRPORT



SA.

Source Code :-

```
from tkinter import *
root = Tk()
root.title("PHOTOIMAGE")
leftframe = Frame(root, height=100, width=100)
leftframe.grid(row=0, column=1, padx=20, pady=20)
rightframe = Frame(root, height=100, width=100)
rightframe.grid(row=0, column=2, padx=20, pady=20)
Label(leftframe, text="Original Image", relief=RAISED)
label.grid(row=0, column=0)
photo = PhotoImage(file="sample.gif")
Label(leftframe, image=photo)
label.grid(row=2, column=3, padx=10, pady=10)
Label(rightframe, image=photo)
label.grid(row=1, column=2)
toolbar = Frame(leftframe, height=50, width=50)
toolbar.grid(row=2, column=2, padx=10, pady=10)
def name():
        text1="KARAN"
        label.Config(text=text1, Justify=LEFT)
def age():
        text2="22"
        label.config(text=text2, justify=LEFT)
def weight():
        text3="65"
        label.config(text=text3 justi
```

Practical 5(4)

Aim: Write a program for displaying the image by using the concept of frame, toolbar, grid method and button method.

Algorithm:

Step 1: Create an object corresponding to the parent window and use the following three methods.
1) title() 2) maxsize() 3) config()

Step 2: Create a leftframe object from frame method and place it onto the parent window with height & width specified. Subsequently use the grid method with row, column, padx, pady specified.

Step 3: Now create a rightframe object from frame method with height & width specified and the row and column value should be specified.

Step 4: Crefate a Label object from the Label method and place it onto the leftframe with the text attribute denoting original image with relief attribute and subsequently use the grid method with row, column value (0,0) with some external padding.

Step 5: Now use the photoImage method with the file attribute specified.

Step 6: Use the subsample method with the object of the image and give the x & y co-ordinate values.

Step 7: Use the label method and position it onto the leftframe and place the image after the sampling and use the the grid method for positioning in the first row.

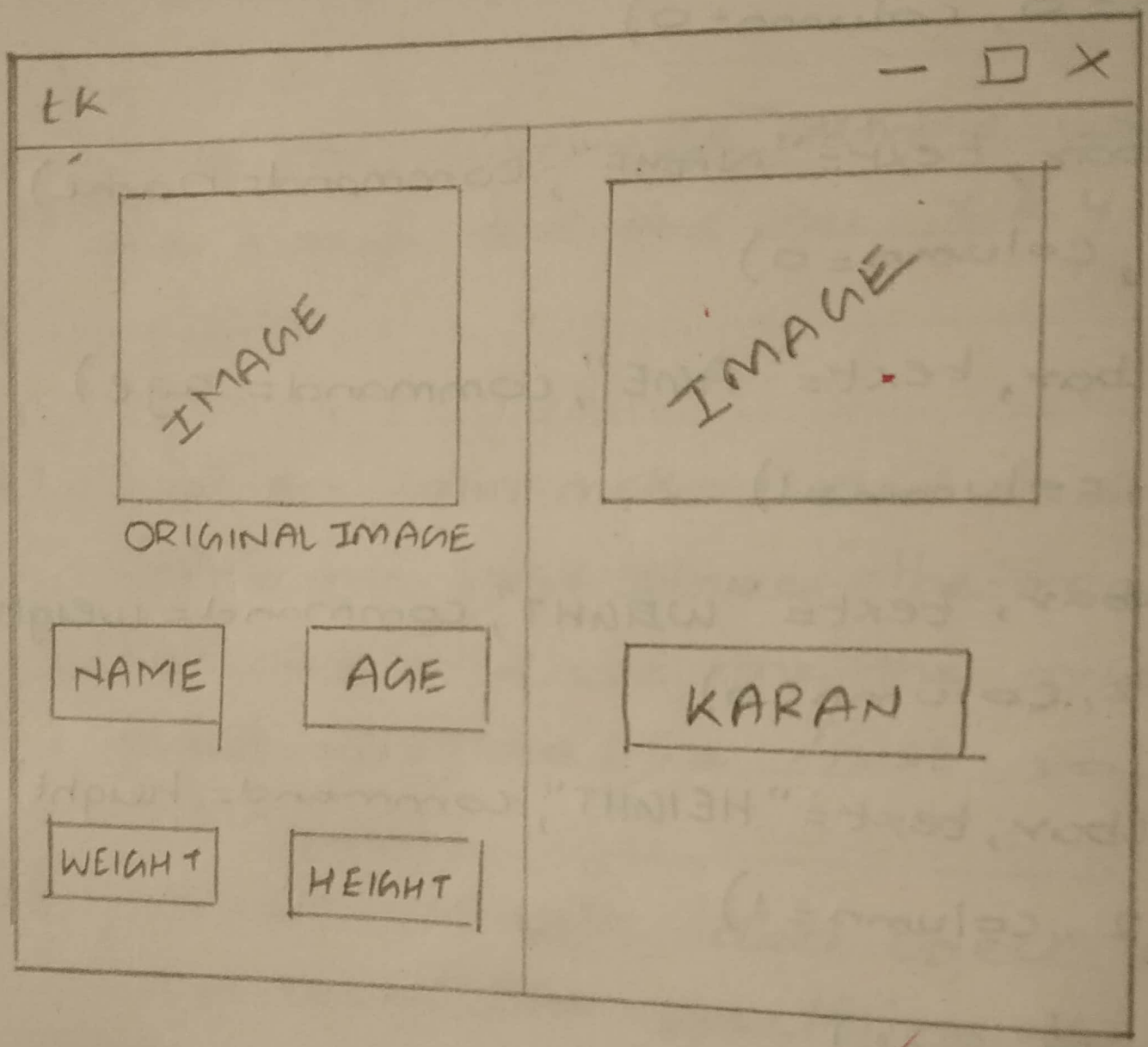
Step 8: Create another label object positioning onto rightframe and specifying the image and background with row and column specified.

Step 9: Now create a toolbar object from the frame method and position it onto the leftframe with height & width specified and position it onto the second row.

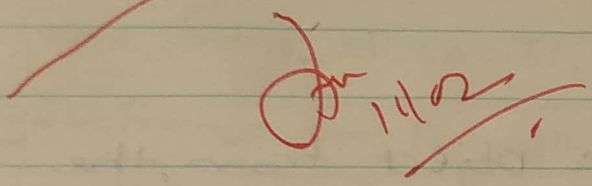
Step 10: Now define the various functions for the different toolbar ~~obt~~ options provided in the leftframe

Step 11: From the label method position text onto the toobar. Use relief and corresponding grid value.

```
def height():
        text4 = "5ft 7in"
        label.config(text = text4, justify = LEFT)

Label(toolbar, text = "INFO")
label.grid (row=0, column=0)

b1=Button (toolbar, text="NAME", command= name)
b1.grid (row =1, column = 0)

b2 = Button (toolbar, text="AGE", command = age)
b2.grid(row=1, column =1)

b3 = Button (toolbar, text= "WEIGHT", command = weight)
b3.grid(row = 2, column=0)

b4 = Button ( toolbar, text= "HEIGHT", command= height)
b4.grid(row = 2 , column = 1)

label = Label (rightframe)
label.grid
mainloop()
```



Step 12: Create the label method, position it onto toolbar with the next title as personal information and position it onto the same row, but next column.



PRACTICAL 5(5)

* Spinbox

Algorithm:

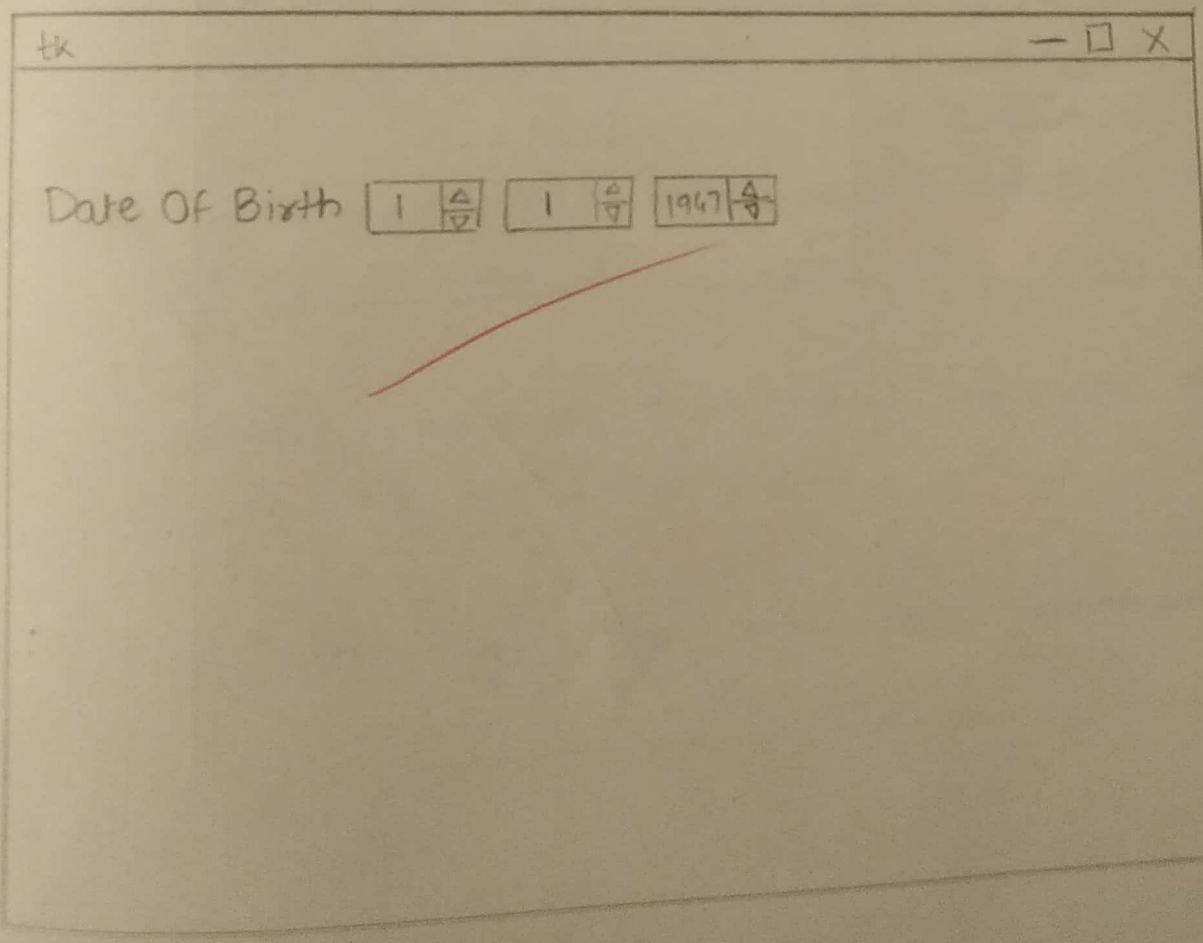
Step 1: Create an object from the Tk() and subsequently create an object from Spinbox().

Step 2: Make the object created on the parent window.

Source Code:-

```
from tkinter import *
root = Tk()
root.config()
root.minsize(height = 500, width=600)
Label(root, text = "Date Of Birth").place(x = 50, y=100)
s1 = Spinbox(root, from_=1, to=31, width=5).place(x=140,y=100)
s2 = Spinbox(root,from_=1, to=12, width=5).place(x=190, y=100)
s3 = Spinbox(root, from_=1947, to=2020, width=5).place(x=240, y=100)
mainloop()
```

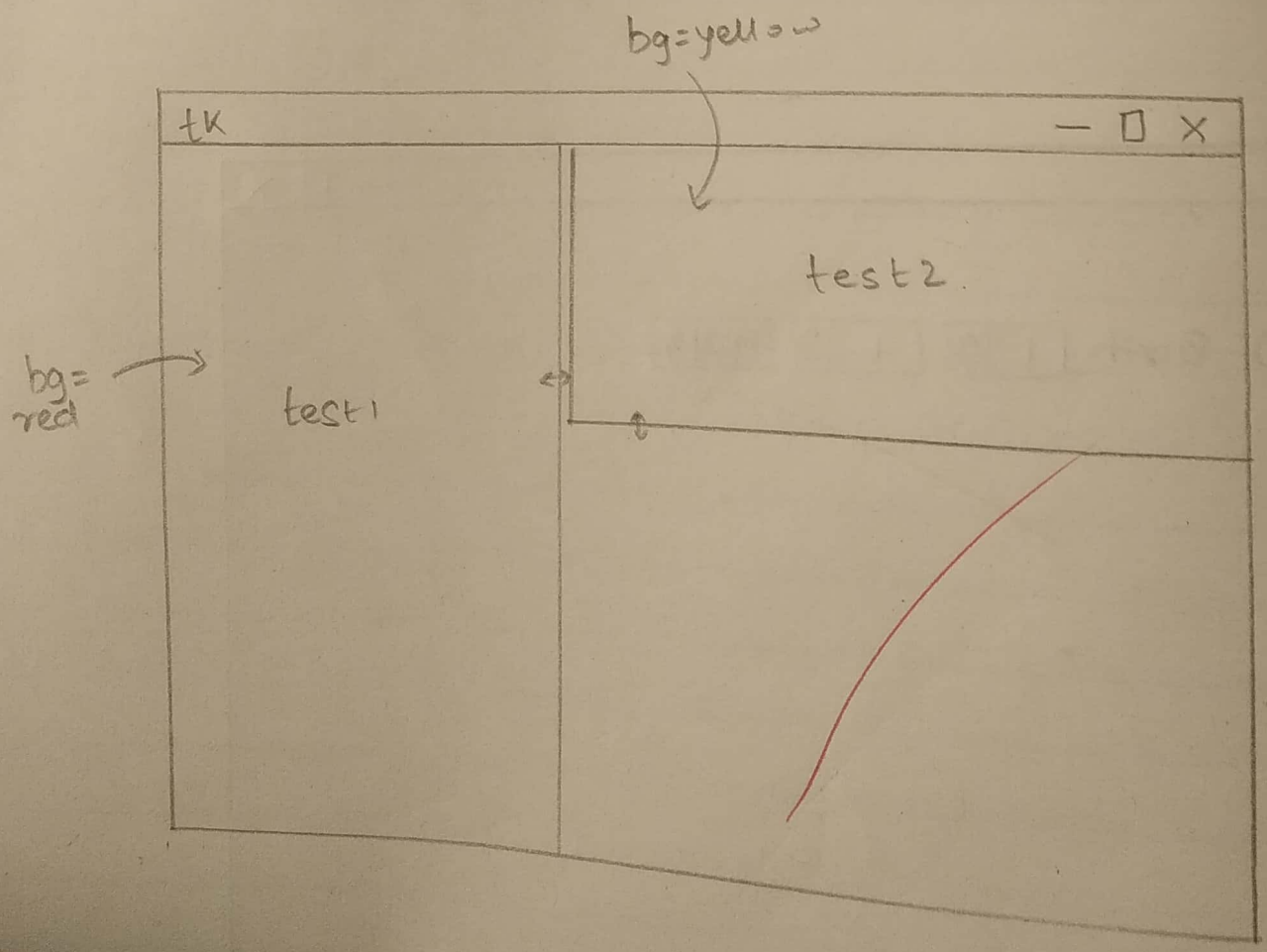
Output:-



Source Code:-

```
from tkinter import *
root = Tk()
P1 = PanedWindow()
P1.pack(fill = BOTH, expand = 10)
L1 = Label(P1, text = "Test1", bg = "red")
P1.add(L1)
P2 = PanedWindow(P1, orient = VERTICAL)
P1.add(P2)
L2 = Label(P2, text = "test2", bg = yellow")
P2.add(L2)
mainloop()
```

Output:



* Paned Window

Algorithm :-

Step 1 : Import the relevant method from tkinter library.

Step 2 : Create an object of parent window and also an object of PanedWindow method.

Step 3 : Now use attributes fill and expand in pack method.

Step 4 : Create an object of Label method and place it on the PanedWindow.

Step 5 : Now use add() along with object of Paned Window specified.

Step 6 : Similarly create few more objects from Pane Window and Label.

Step 7 : Finally, trigger the main loop method.

* Canvas

Algorithm :-

Step1 : Import the relevant method from tkinter library.

Step2 : Create an object of canvas() and place it on the parent window with height, width, bg, specified.

Step3 : Create an arc using create_arc method along with canvas object and use the co-ordinate values.

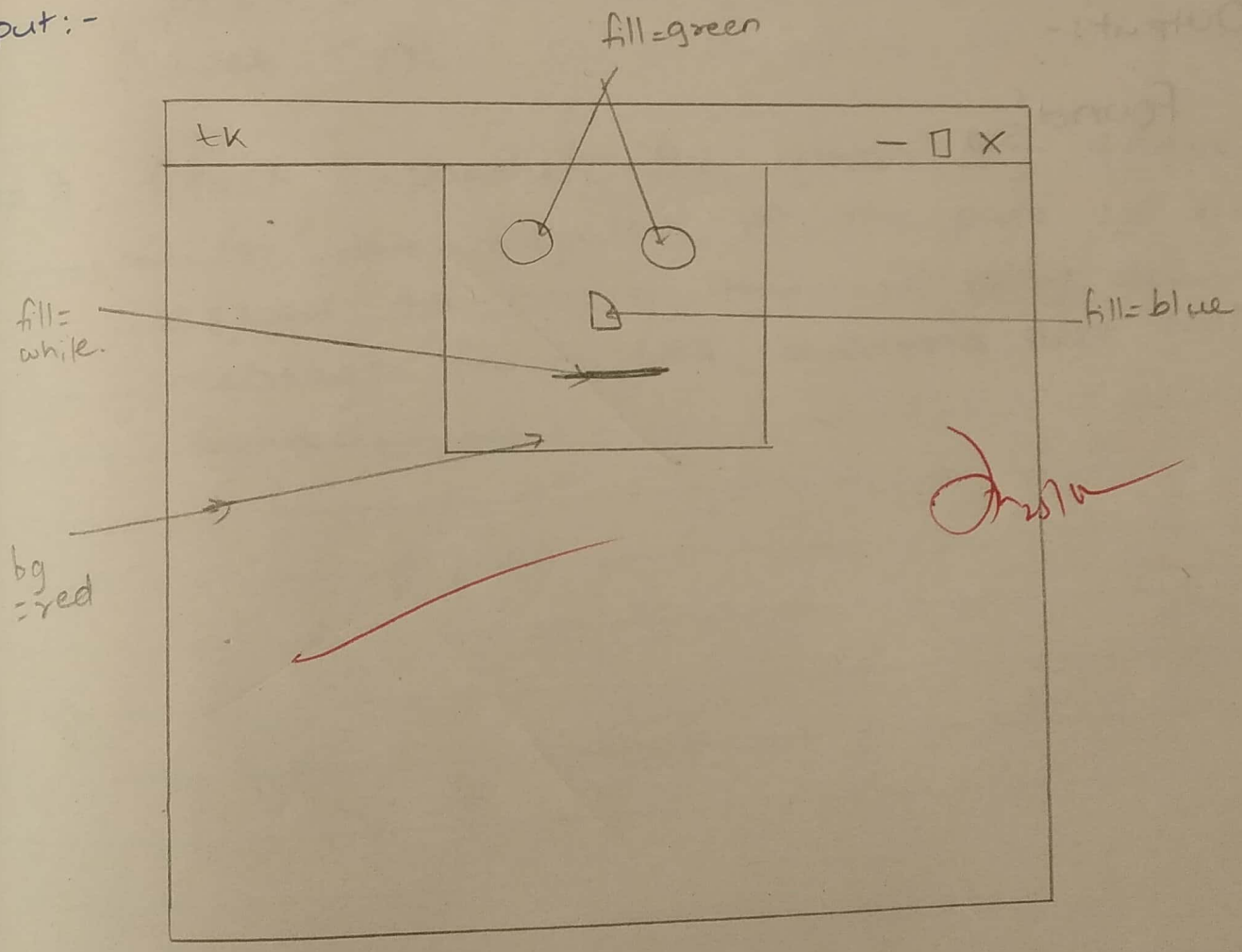
Step4 : Similarly use create_line, create_oval method along with the coordinate values.

Step5 : Call the pack() and mainloop().

Source Code:-

```
from tkinter import *
root=Tk()
c1=Canvas(root,height=100, width=100, bg="red")
c_arc=c1.create_arc(30,60,70,100,fill="blue")
c_line=c1.create_line(30,90,80,90,fill="white")
c_oval=c1.create_oval(10,30,40,60, fill="green")
c_oval=c1.create_oval(60,30,90,60, fill="green")
c1.pack()
mainloop().
```

Output:-



Source Code :-

```
import dbm
db = dbm.open("Demo", flag, "c")
db["Demo record"] = "Database"
if db["Demo record"] != None:
    print("Found!")
else:
    print("Not Found")
db.close()
```

Output :-

Found!

Practical - 6

Topic : Database Connectivity

Step 1 : Import the library and use the open for creating the database by specifying name of database.

Step 2 : Use the objects for accessing to give size and corresponding regular for the web size.

Step 3 : Check whether the given URL address with the regular of the page is not equal to none than display the message from URL address else NOT formed.

Algorithm :-

Step 1: Import coressponding libraries for making database connection.

Step 2: Now create cursor object using cursor() from the connection object created.

Step 3: Now use the execute() for creating the table with the column name & respective data type.

Step 4: Use the commit() to complete the transaction using the connection object.

Step 5: Use the execute statement along with cursor object for acessing the values from database using the select from where clause.

Step 6: Finally use fetchall() for displaying values.

Step 7: Use the execute() and drop table.

Source code :-

```
import os
import sqlite3
connection = sqlite3.connect("Employee.db")
cursor1 = connection.cursor()
cursor1.execute('create table employee(Name CHAR,
                                       Department CHAR)')
cursor1.execute('insert into employee values("Dharan",
                                              "IT")')
cursor1.execute('insert into employee values ("Karan","Medi
connection.commit()
for row in cursor1.execute('select Name. from Employee')
    print(row)
cursor1.fetchall()
```

Print " " , row[0]
row[1]

Output:

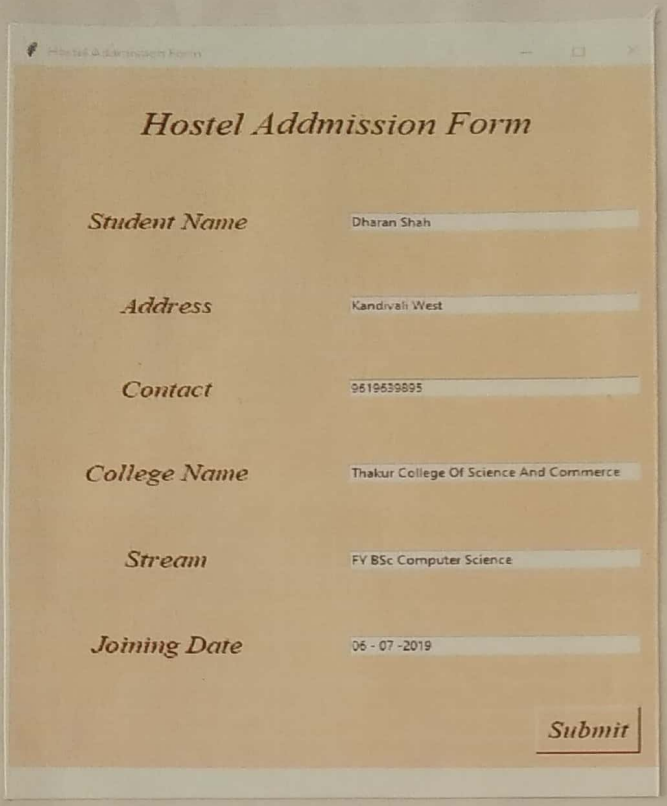
```
('Dharan',)
('Karan',)
```

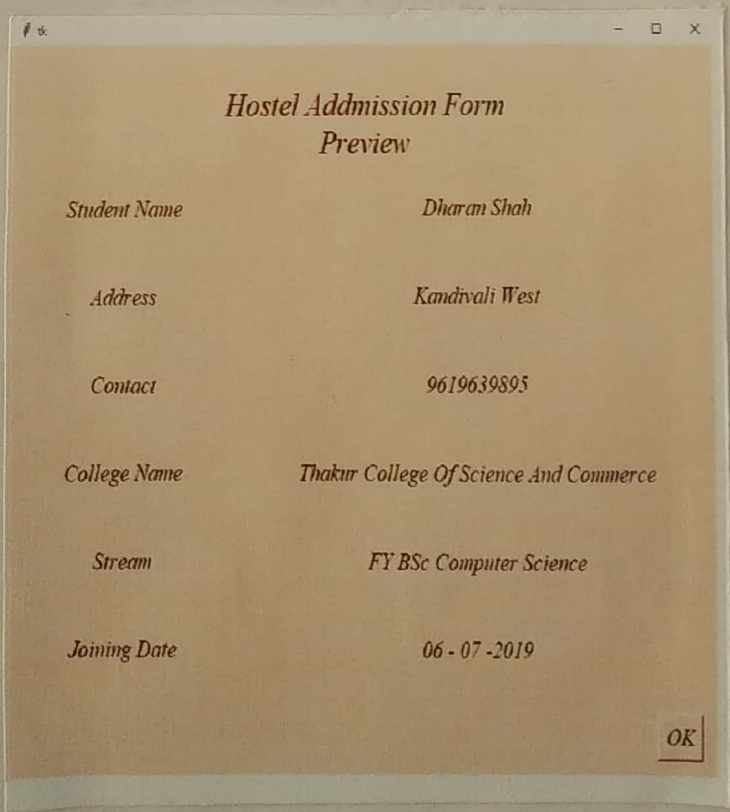
PROJECT

GUI - Tkinter

```
from tkinter import *

def submit():
    y=Tk()
    y.config(bg='Moccasin')
    name=str(e1.get())
    address=str(e2.get())
    contact=str(e3.get())
    college=str(e4.get())
    stream=str(e5.get())
    jdate=str(e6.get())

    Label(y, text='\nHostel Addmission Form\nPreview', font='Times 25
italic',bg='Moccasin').grid(row=0, columnspan=2)

    Label(y, text="Student Name", width=20, height=3,font='Times 18
italic',bg='Moccasin').grid(row=1, column=0)
    Label(y, text=name, width=40, height=3,font='Times 18
italic',bg='Moccasin').grid(row=1, column=1, padx=20)

    Label(y, text='Address', width=20, height=3,font='Times 18
italic',bg='Moccasin').grid(row=2, column=0)
    Label(y, text=address, width=40, height=3,font='Times 18
italic',bg='Moccasin').grid(row=2, column=1, padx=20)

    Label(y, text='Contact', width=20, height=3,font='Times 18
italic',bg='Moccasin').grid(row=3, column=0)
    Label(y, text=contact, width=40, height=3,font='Times 18
italic',bg='Moccasin').grid(row=3, column=1, padx=20)

    Label(y, text='College Name', width=20, height=3,font='Times 18
italic',bg='Moccasin').grid(row=4, column=0)
    Label(y, text=college, width=40, height=3,font='Times 18
italic',bg='Moccasin').grid(row=4, column=1)

    Label(y, text='Stream', width=20, height=3,font='Times 18
italic',bg='Moccasin').grid(row=5, column=0)
    Label(y, text=stream, width=40, height=3,font='Times 18
italic',bg='Moccasin').grid(row=5, column=1)

    Label(y, text='Joining Date', width=20, height=3,font='Times 18
italic',bg='Moccasin').grid(row=6, column=0)
    Label(y, text=jdate, width=40, height=3,font='Times 18
italic',bg='Moccasin').grid(row=6, column=1)

    Button(y, text='OK', command=y.destroy,font='Times 18
italic',bg='Moccasin').grid(row=8, column=1, pady=20, padx=20, sticky=E)

root=Tk()
root.config(bg='Moccasin')
```

```
root.title("Hostel Addmission Form")
root.minsize(height=600,width=500)
Label(root, text='\nHostel Addmission Form\n', font='Times 25
italic',bg='Moccasin').grid(row=0, columnspan=2)
e1=Entry(root, width=40)
e2=Entry(root, width=40)
e3=Entry(root, width=40)
e4=Entry(root, width=40)
e5=Entry(root, width=40)
e6=Entry(root, width=40)

Label(root, text="Student Name", width=20, height=3,font='Times 18
italic',bg='Moccasin').grid(row=1, column=0)
e1.grid(row=1, column=1, padx=20)

Label(root, text='Address', width=20, height=3,font='Times 18
italic',bg='Moccasin').grid(row=2, column=0)
e2.grid(row=2, column=1, padx=20)

Label(root, text='Contact', width=20, height=3,font='Times 18
italic',bg='Moccasin').grid(row=3, column=0)
e3.grid(row=3, column=1, padx=20)

Label(root, text='College Name', width=20, height=3,font='Times 18
italic',bg='Moccasin').grid(row=4, column=0)
e4.grid(row=4, column=1)

Label(root, text='Stream', width=20, height=3,font='Times 18
italic',bg='Moccasin').grid(row=5, column=0)
e5.grid(row=5, column=1)

Label(root, text='Joining Date', width=20, height=3,font='Times 18
italic',bg='Moccasin').grid(row=6, column=0)
e6.grid(row=6, column=1)

Button(root, text='Submit', command=submit,font='Times 18
italic',bg='Moccasin').grid(row=8, column=1, pady=20, padx=20, sticky=E)

mainloop()
```





GUI And Database - Tkinter and SQLite3

```
from tkinter import *
from tkinter import messagebox
import sqlite3

#Register Window
def register():
    def insert():
        user=str(Username.get())
        passw=str(Password.get())
        conn=sqlite3.connect("loginpage.db")
        cur=conn.cursor()
        cur.execute('CREATE TABLE IF NOT EXISTS "login" ("Username" TEXT,"Password"
TEXT)')

        with conn:
            cur.execute('INSERT INTO login (Username,Password) VALUES
(?,?)',(user,passw))

        conn.commit()
        messagebox.showinfo("Success","You have been registered .")
        root.destroy()

    root=Toplevel()
    root.config(bg="steelblue")
    root.title("Register Now !")
    root.minsize(height=400,width=600)
    Username=StringVar()
    Password=StringVar()
    Label(root,text="Username :",font='Times 25',bg='steelblue').place(x=50,y=50)
    Label(root,text="Password :",font='Times 25',bg='steelblue').place(x=50,y=130)
    Button(root,text="Submit",font='Times
25',bg='steelblue',width=10,command=insert).place(x=200,y=250)
    Entry(root,font='Times 20',bg='steelblue',textvar=Username).place(x=220,y=54)
    Entry(root,font='Times
20',show="*",bg='steelblue',textvar=Password).place(x=220,y=134)

def welcome():
    def destroy():
        root.destroy()
        root=Toplevel()
        root.config(bg='steelblue')
        root.title("Expense Manager : Dashboard")
        root.minsize(height=900,width=1600)
        Button(root,text="Add Income",font='Times 30
italic',bg='steelblue',width=15,height=1,relief=GROOVE).place(x=50,y=50)
        Button(root,text="Add Expense",font='Times 30
italic',bg='steelblue',width=15,height=1,relief=GROOVE).place(x=435,y=50)
        Button(root,text="Add Savings",font='Times 30
italic',bg='steelblue',width=15,height=1,relief=GROOVE).place(x=820,y=50)
        Button(root,text="Logout",font='Times 30
```

```
italic',bg='steelblue',width=15,height=1,relief=GROOVE,command=destroy).place(x=120
0,y=50)

def login1():
    s=e1.get()
    t=e2.get()
    conn=sqlite3.connect("loginpage.db")
    cur=conn.cursor()
    cur=conn.execute("SELECT  Username,Password from login")
    cur.fetchone()
    for row in cur:
        if (s==row[0] or t==row[1]):
            response1=messagebox.showinfo("Success","You have logged in.")
            welcome()
        else:
            messagebox.showwarning("Warning Message","No Record Found !!!")



# Main Login Window
root=Tk()
root.config(bg='steelblue')
root.title("Expense Manager")
root.minsize(height=900,width=1600)
Label(root,text="Welcome To Expense Manager !\nPlease Login To
Proceed....",font='Times 50 italic',bg='steelblue').place(x=350,y=150)
Label(root,text="Enter Username :",font='Times 30
italic',bg='steelblue').place(x=450,y=400)
Label(root,text="Enter Password :",font='Times 30
italic',bg='steelblue').place(x=450,y=500)
e2=Entry(root,show="*",font='Times 20 italic',bg='steelblue')
e2.place(x=750,y=510)
e1=Entry(root,font='Times 20 italic',bg='steelblue')
e1.place(x=750,y=410)
Button(root,text="Login",font='Times 25
italic',bg='steelblue',width=10,relief=GROOVE,command=login1).place(x=650,y=590)
Label(root,text="New User ?                                    Now !",font='Times 20
italic',bg='steelblue').place(x=580,y=750)
Button(root,text="Register",font='Times 20 italic
underline',bg='steelblue',width=6,relief=FLAT,command=register).place(x=720,y=741)

mainloop()
```