☆☆ INDEX ☆☆

```
fileobj = open("word.txt","w")
fileobj.write("CS subjects"+"\n")
fileobj.write("DM \n FOSS \n DS \n")

fileobj = open("word.txt","r")
str1 = fileobj.read()
print("The output : ", str1)
fileobj.close()
>>('The output : ','CS subjects \n DM \n FOSS
    \n DS ')
fileobj = open("word.txt","r")
str2 = fileobj.readline()
print("The output of readline method : ", str2)
fileobj.close()
>> ('The output of readline method : ', 'CS
   subjects \n')
fileobj = open("word.txt","r")
str3 = fileobj.readlines()
print("The output : ", str3)
fileobj.close()
>>('The output of readlines method :',
  ['CS subjects \n', 'DM \n FOSS \n DS \n'])
a = fileobj.name
print("name (name attribute) : ", a)
>>> (name(name attribute)', 'ab word.txt')
b = fileobj.closed
print("(close) attribute : ", b)
>>> ('(close) attribute : ', 'True')
```

PRACTICAL - 1

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

Algorithm:

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 the 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.

Step 4: Now open the fileobj in write mode write some another content close subsequently then again open the fileobj in 'w+' 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 passed and display the output subsequently.

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

Step 7: Open the fileobj in read mod. declare a variable and perform fileobj dot tell() and store the output consequently 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 fileobj with read mode also use the readline method and store the output consequently in and print the same for counting the length we the for conditional statement and display the length.

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

fileobj=open("word.txt","w+")
fileobj.write("ABC")
fileobj.close()

fileobj=open("word.txt","r+")
s1=fileobj.read(2)
print("output of r+ ", s1)
fileobj.close()
>>>(output of r+ :'AB')

fileobj=open("word.txt","a+")
fileobj.write("data")
fileobj.close)
fileobj=open("word.txt","r")
s3=fileobj.read()
print(Output: ", s3)
fileobj.close()
>>> output : 'ABC data'

fileobj=open("word.text","w+")
fileobj.write("abc")
fileobj.close()

fileobj=open("word.txt","r")
s=fileobj.read()
print("output :", s)
>>>(output: 'ABC')
```

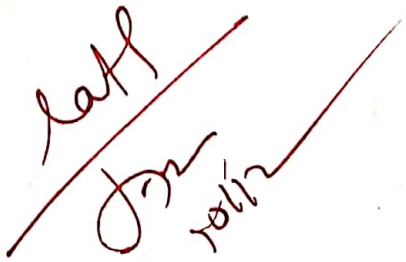
SS :-

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

```
fileobj = open("word.txt", "r")
k = fileobj.seek(0,0)
print("seek(0,0) is: ", k)
fileobj.close()
>>>('seek(0,0) is: ", none)
```

```
fileobj = open("word.txt", "r")
k1 = fileobj.seek(0,1)
print("seek(0,1) is: ", k1)
fileobj.close()
>>>("seek(0,1) is: ", none)
```

```
fileobj = open("word.txt", "r")
k2 = fileobj.seek(0,2)
print("seek(0,2) is: ", k2)
fileobj.close()
>>>('seek(0,2) is: ; none)
```



PRACTICAL - 2

Aim : Iterators

Step 1 : Create a tuple. The number of time we use the iter and next method we will get the iterating element in the tuple.

Step 2 : Define a function with a parameter which will return square of the given number. Similarly declare a function to get the cube of a number. Call the declared function.

Step 3 : Use for loop to specify the range then use the list typecasting with map method declare anonymous function and print.

Step 4 : Declare a list with some elements. Use the map method with the help of anonymous function - Display the output.

```
t = ("Monday", "Tuesday", "Sunday")
m = iter(t)
print(next(m))
m1 = iter(t)
print(next(m))
m2 = iter(t)
print(next(m))
>>> Monday
    Tuesday
    Sunday

def s(x):
    y = x * x
    return y
def c(x):
    z = x * x * x
    return z
f = [s, c]

for n in range(4):
    v = list(map(lambda x: x(n), f))
    print(v)
>>> [0, 0]
    [1, 1]
    [4, 8]
    [9, 27]
```

```
l=[0,4,5,7,9,11,13,15,20,19,25]
l = list(map(lambda x: x%5, l))
print(l)
def e(x):
    if (x%2==0):
        return "even".
    else:
        return "odd"
list(map(even, l)) >>>[0,4,0,2,4,1,3,0,0,4,0]
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
    myobj=odd()
    myiter=iter(myobj)
x=int(input("Enter a number:"))
for i in myiter:
    if (i<x):
        print(i)
```

o/p

9,

Step 5 : Use appropriate logic to check whether the number is even or odd

Step 6 : Define a class and iter() method to initialize the first element within the object

Step 7 : Now use next() and define logic for displaying odd value. Define object.

Step 8 : Accept input from the user.

10/12

Step 9 : Define a function with if condition if exactly equal to one return else one parameter.

Step 10 : Use append method for appending the input value. call the map using two parameter. print the factorial for input variable.

```
>>> Enter a number: 10
1
3
5
7
9
```

```
def f(x):
    if(x==1):
        return 1
    else:
        return (x*f(x-1))
x=int(input("Enter a number: "))
list=[]
list.append(x)
n=map(f,list)
print("The Factorial is: ",n)
```

```
>>> Enter a number: 5
('The Factorial is: ', [24])
```

```
# IOError
try:
    fo = open("abc.txt", "r")
    fo.write(" If I were a butterfly ")
except IOError:
    print("Error")
else:
    print("Successful")
>>> Error

# ValueError
try:
    x=int(input("enter: "))
except ValueError:
    print("Arithmetic Error")
else:
    print("successful")
>>> enter: 11
 Successful
>>> enter: lol
 Arithmatic Error

# TypeError, ZeroDivisionError
try:
    print(1/0)
    print('10' + 23)
except(TypeError, ZeroDivisionError):
    print("Invalid")
>>> Invalid
```

17/12/19

PRACTICAL - 3

Aim : Exceptions

Algorithm:

Step 1: Open a fileobject in read mode and use the write access mode followed by writing some contents into the file

Step 2: Except IOError and print suitable message if not print appropriate message

Step 3: Accept input from the user in try block. If it has ValueError display the the message accordingly.

Step 4: Print statements with TypeError or ZeroDivisionError in try block. Now except TypeError and ZeroDivisionError together and display suitable message. If there is no error print suitable message.

Step 5: Initialize a variable with an integer. Then accept a value from user.

Step 6: Print result of division of two variables in try block. Now except TypeError and print appropriate message.

Step 7: Else except zeroDivisionError and display a suitable message.

= TypeError and
= ZeroDivisionError

```
a,b = 1,
b = input("Enter: ")
try:
    print(a/b)
except TypeError:
    print("Incompatible Value ")
except ZeroDivisionError:
    print("Denominator is zero ")
```

```
>>> Enter: 2
  0.5
>>> Enter: 0
 Denominator is zero
>>> Enter: c
 Incompatible value
```

```
# split method
import re
seq = 'hello123, howdy789, 456howru'
pattern = r'\d+'
output = re.split(pattern, seq)
print(output)
['123', '789', '456']
import re
s = 'hello123, howdy789, 456howru'
pattern = r'\d+'
o = re.split(pattern, s)
print(o)
['hello', 'howdy', 'howru']
# sub method
import re
s = 'abc def ghi'
pat = r'\s+'
r = ''
output = re.sub(pattern, r, s)
print(output)
abcdefghi
# findall method
import re
seq = 'abc@edu.com, xyz@google.com'
p = '[\w\.-]+[\w\.-]+'
result = re.findall(p, seq)
print(result)
['abc', 'edu.com', 'xyz', 'google.com']
```

24/12/19

PRACTICAL - 4

Aim: Regular Expression

Algorithm:

Step 1: Import re module in python environment
Step 2: Initialize a variable to store a string Use appropriate pattern to split the numbers. Now use the split() method with the pattern and string as its arguments.
Step 3: Print the result.
Step 4: Again import re module. Initialize a variable with string of your choice An appropriate pattern should be used for seperating the characters. Use split() method with pattern and string as its arguments and then display the output.
Step 5: Import re module from the library. A sequence of your choice can be initialized. A pattern both beginning with raw string in formed. Then initialize a variable with blankspace Use sub() method with three arguments pattern, sequence and the variable initialized with blankspace. Print the output

Step 6: Import re module from the library. Initialize a sequence with desired string. The pattern to be used includes [] with \w\.-
Use the concatenation operator

Step 7: \w in the pattern represents a match with any alphanumeric character. Use findall() method with pattern and sequence as arguments and display the result.

Step 8: Import re module. Initialize a sequence. Use findall() method with raw string and list of vowels i.e pattern along with the sequence. Print the output.

Step 9: Import re module. Initialize a string with list of males and females. Form a pattern beginning with raw string. Use findall() method with two arguments i.e pattern and sequence.

Step 10: Print the output. Initialize a variable for ~~cout~~ counting males and another to count females. Use for loop to check condition if it is 'ms.' than count of female will be incremented else count of male will be incremented.

Step 11: Display the result with exact count of males and females.

```
import re
s='thakur college of sci and com'
p=re.findall(r'[aeiouAEIOU]\w+',s)
print(p)
['akur','ollege','of','and','om']
```

```
import re
s='mH.a,ms.d,ms.t,mH.g,ms.f'
p=r'[/ms/mH/]+'
o=re.findall(p,s)
print(o)
m=0
f=0
for v in o:
    if v=='ms':
        m=m+1
    else:
        f=f+1
print("No. of males is: ",fm)
print("No. of females is: ",f)
```

```
['mH','ms','ms','mH','ms']
No. of males is : 2
No. of females is : 3
```

```
#seach method
import re
s='python is an interesting language'
o=re.seach('\Apython',s)
print(o)
v=o.group()
print(v)
<-sre.SRE_Match object ; span=(0,6), match='python'>
python
```

match method

```
import re
l=['8345431351', '9123456781', '7852147963',
    '5124789634']
for v in l:
    if re.match(r'[8-9]{1}[0-9]{9}', v or
        len(v)==10):
        print("cell number matches")
    else:
        print("cell number not matching")
```

```
cell number matches
cell number matches
cell number not matching
cell number not matching
```

Step 12 : Import re module. Intialize a sequence with desired string. Use search() method and the arguments should be the word to searched along with the sequence.

Step 13 : Print the result. Now use the group method and then display the output.

Step 14 : Import re module from the library. Initialize a variable with some cell numbers. Use for loop and the if loop ~~loop~~ with match() method and the arguments should be the range also check the length. If it satisfies the condition print 'cell number matches' else print 'cell number not matching'.

7/11/2022

07/01/2020

PRACTICAL - 5

Aim: GUI components

Algorithm:

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

Step 2: Create an object using TK()

Step 3: Use the pack method along with object created from the text method

Step 4: Use the mainloop method for triggering of the corresponding events

Step 5: Use the tkinter library for importing features of text widget

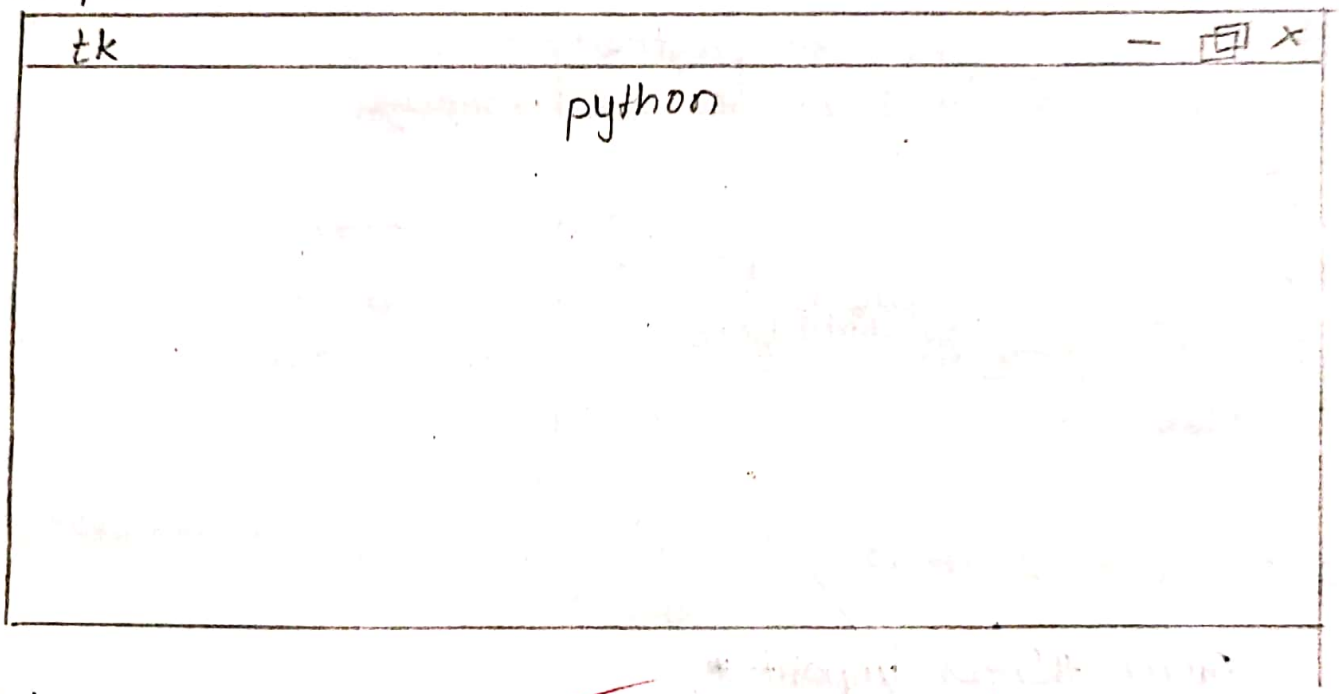
Step 6: Create a variable from text method and position it onto the parent window

Step 7: Use the pack method along with object created from the

#a Label Method

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

Output:

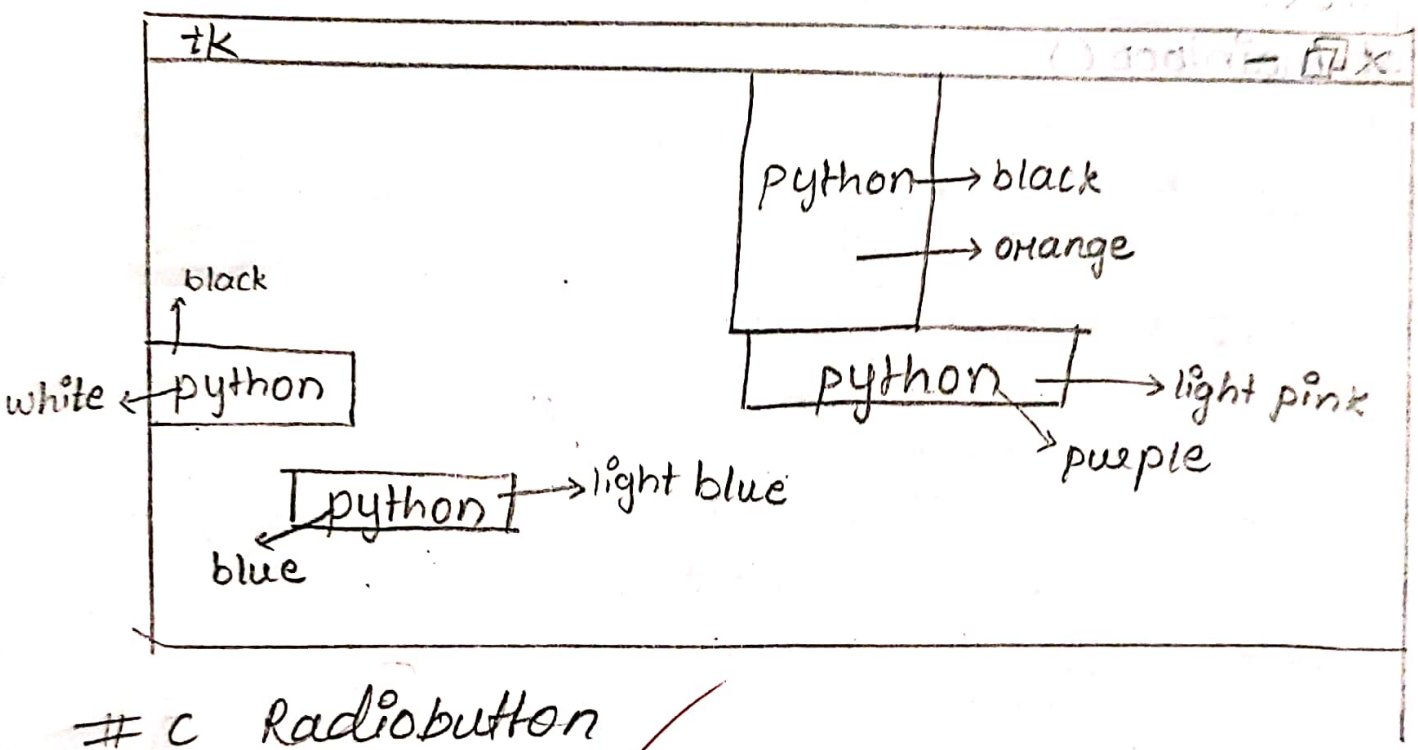


#b

```
from tkinter import *
root = Tk()
l = Label(root, text = "python", bg = "black", fg = "white")
l.pack(side = LEFT, pady = 30)
l1 = Label(root, text = "python", bg = "orange", fg = "black")
l1.pack(side = TOP, ipady = 50)
l2 = Label(root, text = "python", bg = "light blue", fg = "blue")
l2.pack(side = LEFT, padx = 20)
l3 = Labe(root, text = "python", bg = "light pink", fg = "purple")
```

```
l3.pack(side=TOP, ipadx=40)
root.mainloop()
```

Output:



c Radiobutton

```
from tkinter import *
root= Tk()
def sel():
    selection = "you selected the option " + str(var.get())
    l = Label.config(root, text = "selection", justify=LEFT)
    l.pack(anchor=s)
    v = IntVar()
R1 = Radiobutton(text = "option 1", variable = "v",
    value = "1", command = sel)
R1.pack()
R2 = Radiobutton(text = "option 2", variable = "v",
    value = "2", command = sel)
```

text method 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 8: Use the mainloop method for triggering of the corresponding events.

Step 9: Now repeat above steps with label method which takes the following arguments 1. Name of parent window. 2. Text attribute which defines the string 3. The background colour 4. The foreground colour and then use pack method with the relevant padding attributes.

Step 10: Use the tkinter module to import the relevant method

Step 11: Define a function which tells the user about the given selection made from the multiple options available.

Step 12: Use the config method with label object, call the variable as an argument within the method.

Step 13: Now define the parent window and option using the control variable.

Step 14: Now create an object from Radiobutton method which will take arguments positioning on parent window, define the text variable, define the variable argumen and trigger given function

Step 15: Now call the pack method for radio object so created and specify the argument as an anchor attribute

Step 16: Now define the label object from corresponding method subsequently use pack method and make use of mainloop method

Step 17: Import relevant methods from tkinter library.

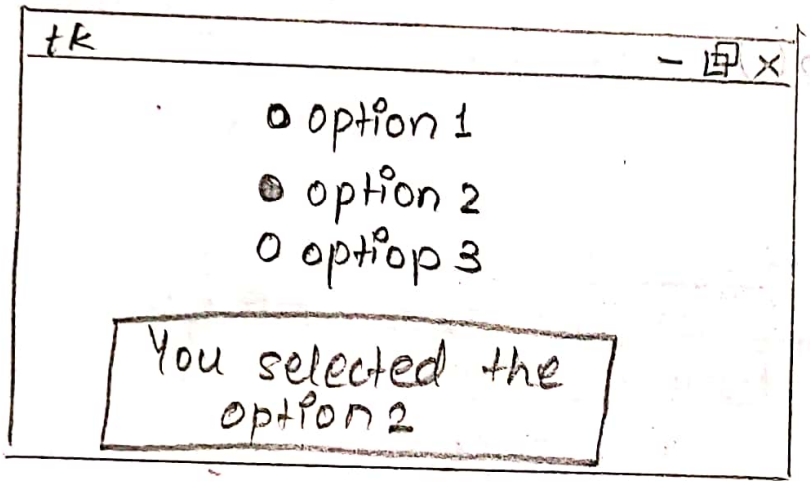
Step 18: Define object corresponding to parent window along with its size

Step 19: Now define the frame object placing it onto parent window. Create another frame object and put it onto parent window on its left side

Step 20: Similarly define the right frame and subsequently define the button

```
R2.pack()
R3 = Radiobutton (text="option 3", variable="v",
     value="3", commanid=sel)
R3.pack()
Root.mainloop()
```

Output::

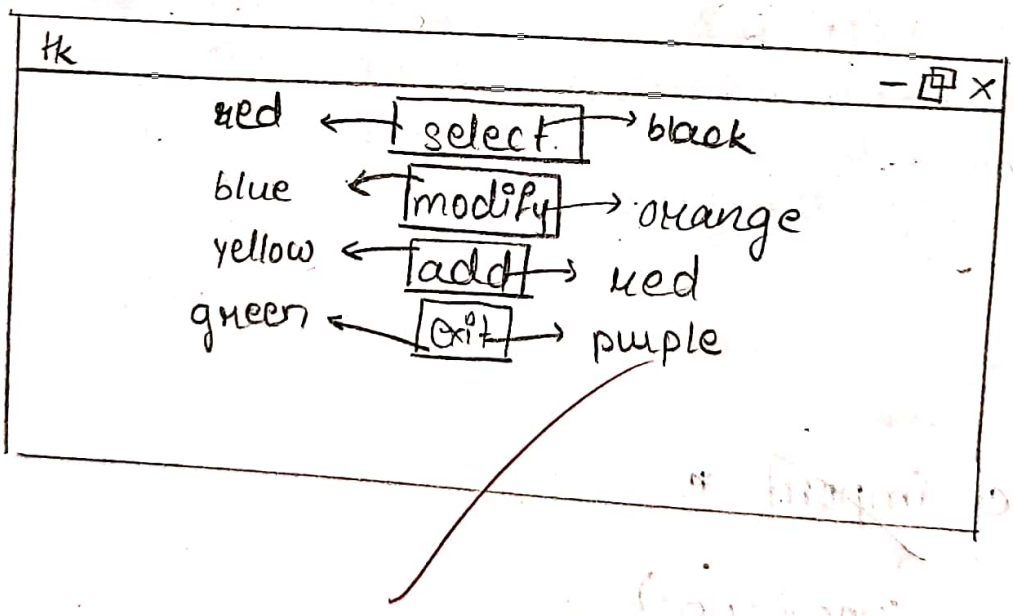


Frame object

```
from tkinter import *
top = Tk()
top.geometry ("100x200")
fram= Frame(top)
fram.pack()
leftfram = Frame (top)
leftfram.pack(side=LEFT)
b1= Button(fram, text="select", activebackground="red",
    fg=black)
b1.pack()
b2 = Button(fram, text="Modify", activebackground="blue"
    fg="orange")
```

```
b2.pack()
b3=Button(frame, text="Add", activebackground="yello
    fg="red")
b3.pack()
b4=Button(frame, text="Exit", activebackground="green
    fg="purple")
b4.pack()
top.mainloop()
```

Output:



object placed onto the given frame with the attribute as text activebackground and fg.

Step 21 : Now use pack method along with side attribute. Similarly create the button object corresponding to modify operation and put it into frame object with side='RIGHT' attribute.

Step 22 : Create another button object and place it onto the right and label the button as Add next add another button and put it onto right frame object and term it as exit.

Step 23 : Use the pack method simultaneously for all the objects and finally use the mainloop method.

2111

* Message box Method

Step 1: Import the relevant method from tkinter library

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

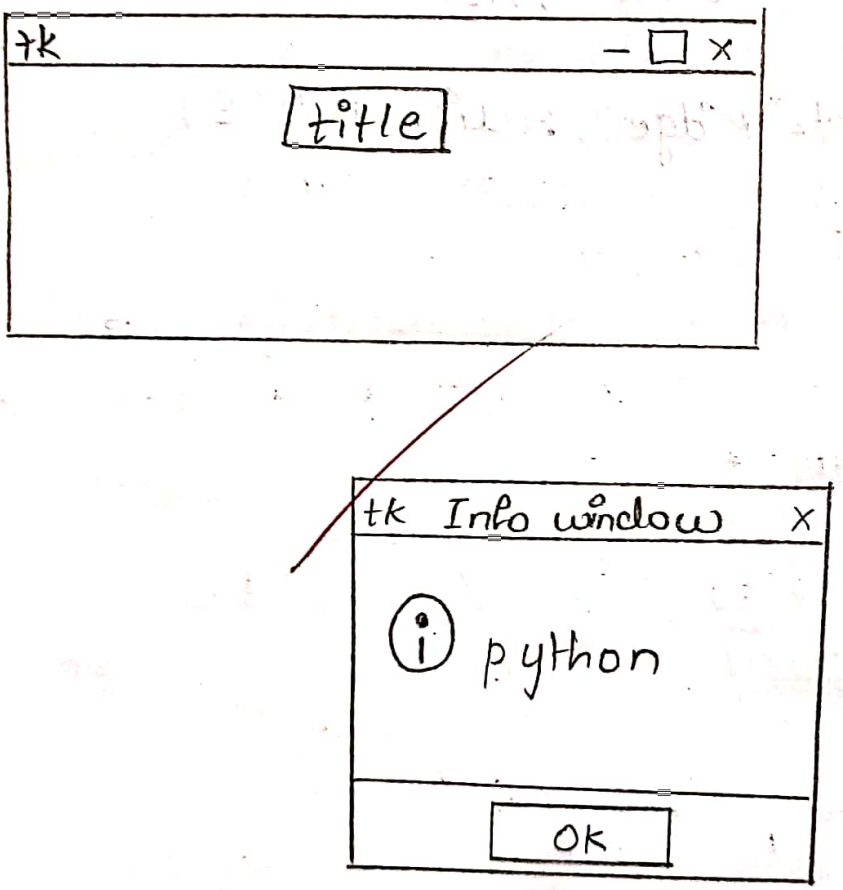
Step 3: Thus different options which are available are showinfo(), showwarning(), showerror(), askyesNo(), askquestion(), askokcancel().

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

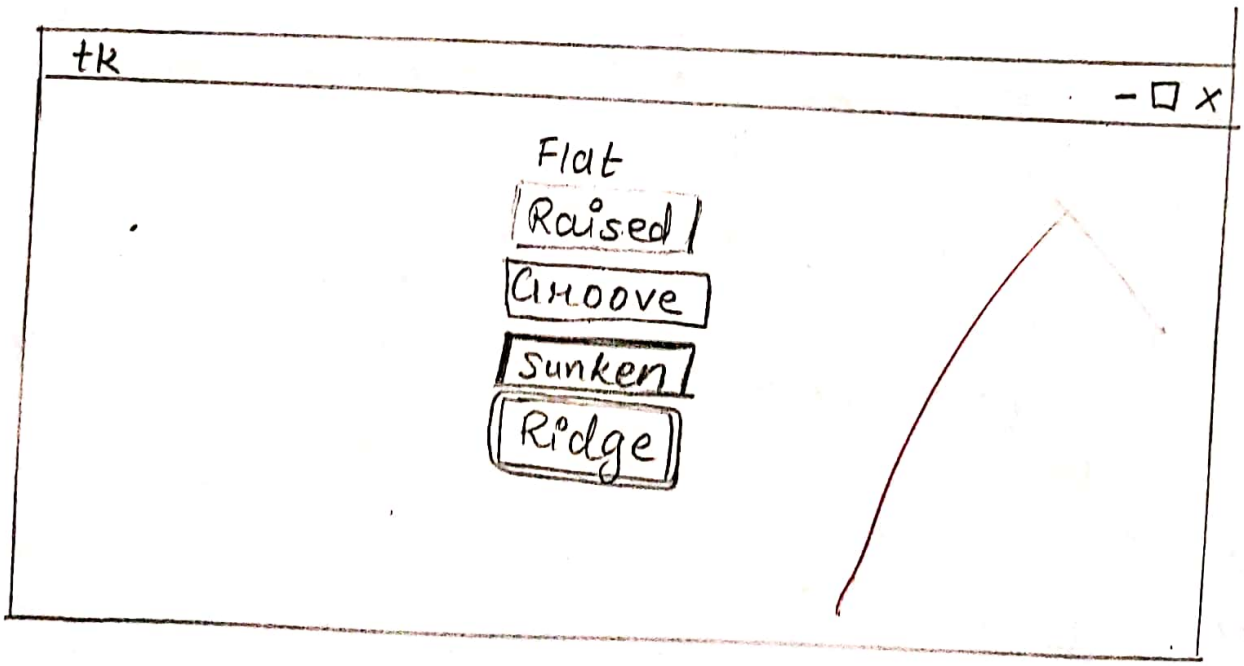
Step 5: Use the pack method to display the button widget and finally use mainloop method.

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

```
from Tkinter import *
import tkMessageBox
root = Tk()
def fun():
    tkMessageBox.showinfo("info window", "python")
b1 = Button(root, text="title", command=fun)
b1.pack()
root.mainloop()
```



```
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()
root.mainloop()
```

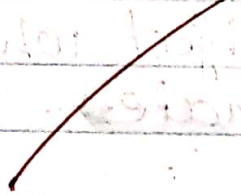


* Relief style

Step 1 : Use the button object with the following attributes
1. The parent window
2. Text attribute
3. Relief

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

Step 3: Finally use the mainloop method.



* Traversing and making use of geometry layout manager method.

Step 1: Define a function and create a object of the given window by using the three methods namely config, title and minsize

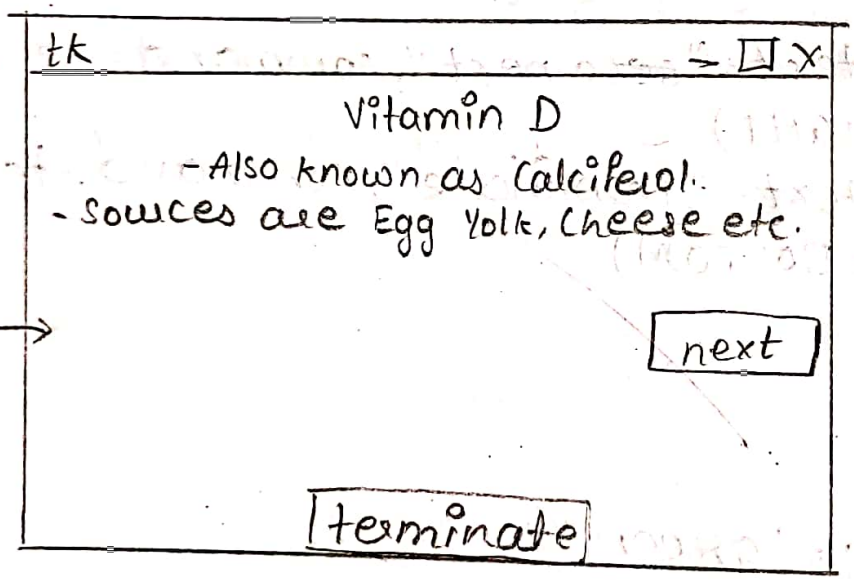
Step 2: Create a button object and use the text and command attribute for triggering the given event and used grid method along with internal and external padding specified similarly create another button object which will allow application to terminate.

Step 3: Define second function corresponding to second window with attributes config, title, minsize for the window object and define one button object which will shift the focus onto the third window.

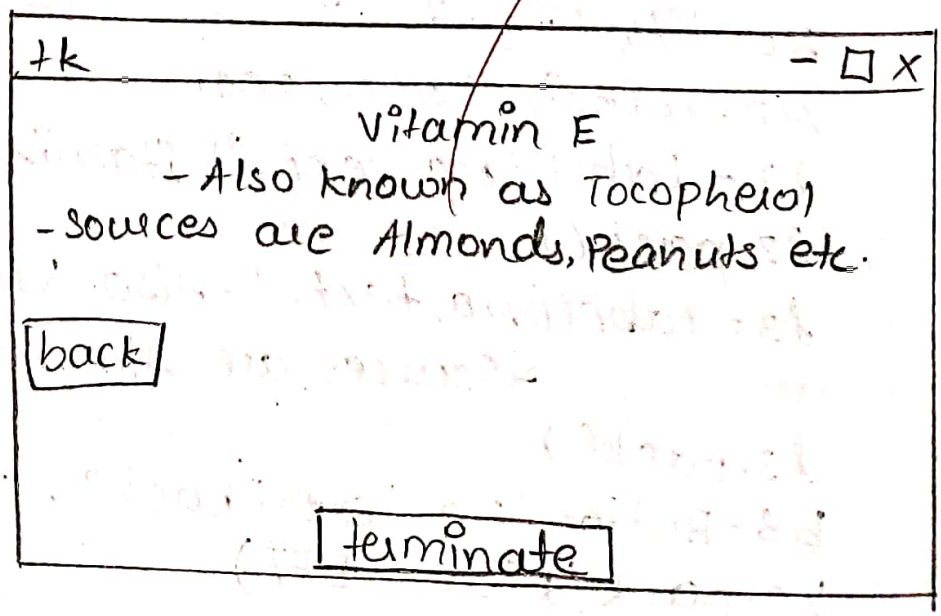
Step 4: Create third window object and in this create two button object for moving on to first window for restarting the process and second button for terminating.

```
from tkinter import *
root = Tk()
def main():
    roo = Tk()
    roo.config(bg="pink")
    roo.title("main")
    roo.minsize(200,200)
    l = Label(roo, text="Vitamin D")
    l.pack()
    l1 = Label(roo, text="-Also known as calciferol \n
            -Sources are Egg yolk, Cheese etc")
    l1.pack()
    b1 = Button(roo, text="~~seto~~ next", command=se)
    b1.pack(side=RIGHT)
    b2 = Button(roo, text="terminate", command=ter)
    b2.pack(side=BOTTOM)
    roo.mainloop()
def se():
    ro = Tk()
    ro.config(bg="green")
    ro.title("2")
    ro.minsize(400,200)
    l2 = Label(ro, text="Vitamin E")
    l2.pack()
    l3 = Label(ro, text="-Also known as Tocopherol \n
            -Sources are Almonds, Peanuts etc.")
    l3.pack()
    b3 = Button(ro, text="back", command=main)
    b3.pack(side=LEFT)
```

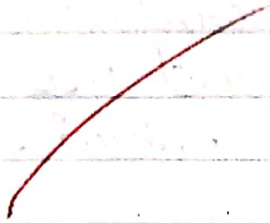
```
b4 = Button(r0, text="terminate", command=tq)
b4.pack(side=BOTTOM)
r0.mainloop()
def ter():
    qquit().

b5 = Button(root, text="Know Your Vitamins",
            command = main)
b5.pack()
root.mainloop()
```

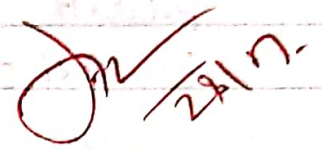


pink



Step 5: Define a function for termination and call the quit method and finally call the first function created and trigger mainloop method.





* Displaying the image

Algorithm:

Step 1: Create an object corresponding to the parent window and use the following 3 methods. • Title • Maxsize • Config

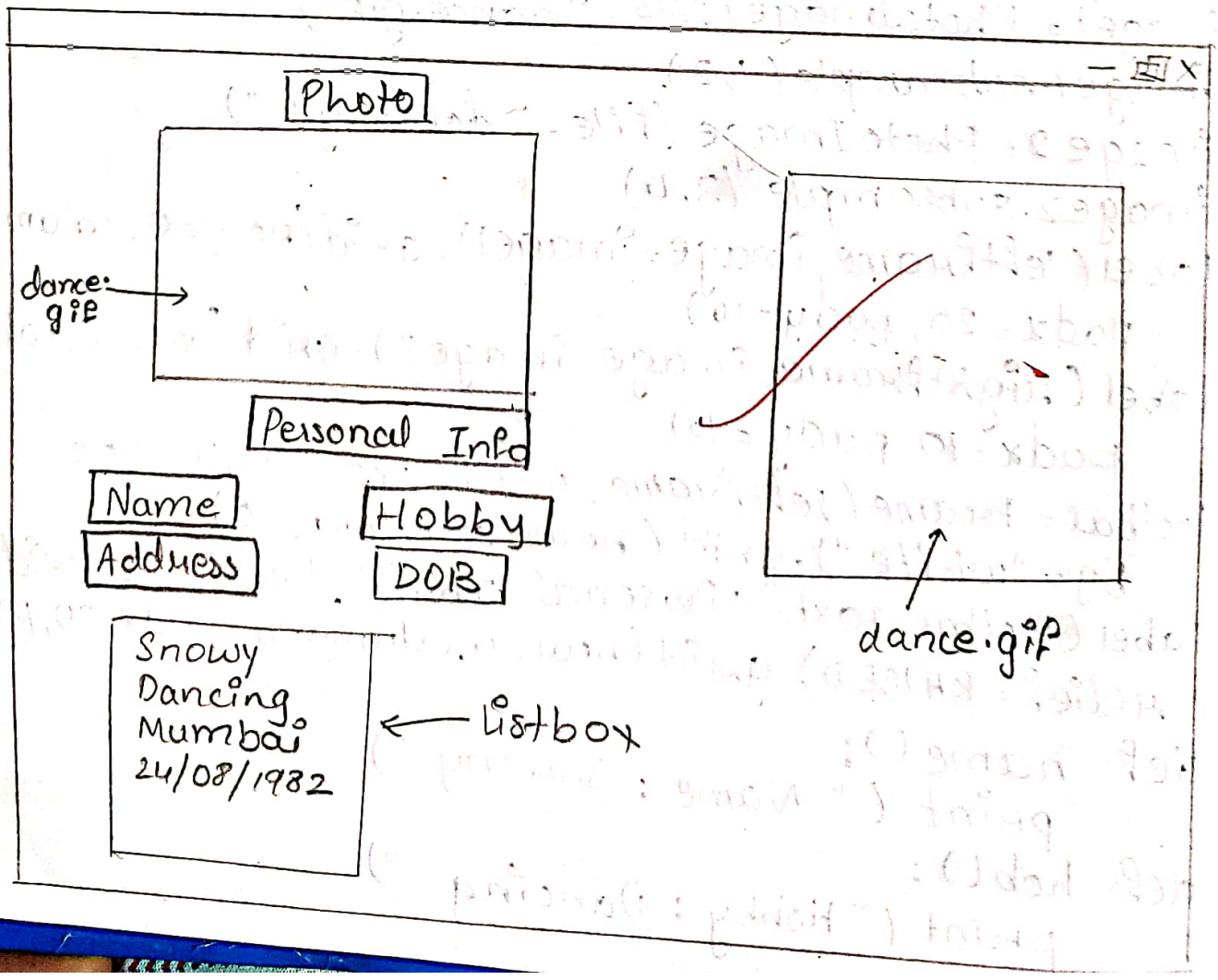
Step 2: Create a leftframe object from the frame method and place it onto the parent window with the height, width and the bg specified. Subsequently use the grid method with the row, column, padx, pady specified

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

Step 4: Create a label object from the label method and place it onto the leftframe with text attribute denoting the original image with relief attribute used as RAISED value and subsequently use grid method with row, column value specified as (0,0) with some external padding values

```
from tkinter import *
root = Tk()
root.title("Python")
root.maxsize(1000, 900)
root.config(bg="black")
leftframe = Frame(root, bg="pink", height="400",
    width="200")
leftframe.grid(row=0, column=0)
rightframe = Frame(root, bg="light green", height="400",
        width="250")
rightframe.grid(row=0, column=0)
Label(leftframe, text="Photo", height=2, width=20).
    grid(row=0, column=0)
image1 = PhotoImage(file="dance.gif")
image1.subsample(1,2)
image2 = PhotoImage(File="dance.gif")
image2.subsample(3,4)
Label(leftframe, image=image1).grid(row=0, column=0,
    padx=20, pady=10)
Label(rightframe, image=image2).grid(row=0, column=1,
    padx=10, pady=10)
toolbar = Frame(leftframe, width=200, height=400,
    bg="white").grid(row=2, column=0)
Label(toolbar, text="Personal Info", height=2, width=20,
  relief=RAISED).grid(row=0,column=0, padx=20, pady=10)
def name():
    print("Name : Snowmy")
def hob():
    print("Hobby : Dancing")
```

```
def add():
    print("Address : Mumbai")
def dob():
    print("DOB: 24/08/1982")
Button(toolbar, text="Name", height=1, width=16, command=
    name).grid(row=1, column=0)
Button(toolbar, text="Hobby", height=1, width=16,
  command=hob).grid(row=1, column=1)
Button(toolbar, text="Address", height=1, width=16,
    command=add).grid(row=2, column=0)
Button(toolbar, text="DOB", height=1, width=16,
    command=dob).grid(row=2, column=1)
root.mainloop()
```



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

Step 6: Use the sub sample 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 left frame and placing the image after the sampling and use the grid method for the positioning in the first row.

Step 8: Create another label object positioning it onto the rightframe and specifying the image and background attribute with row and column attribute specify it as (0,0)

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

Step 10: Now define the various function for different tool bar options provided in the leftframe

Step 11: From the label method position the text on the toolbar use the relief attribute and corresponding guid value and incorporate the internal padding as well.

Step 12: Create the label method position ~~in~~ it on the toolbar with the next title as personal information and position it on same row but next column

Step 13: Now make use of mainloop method.

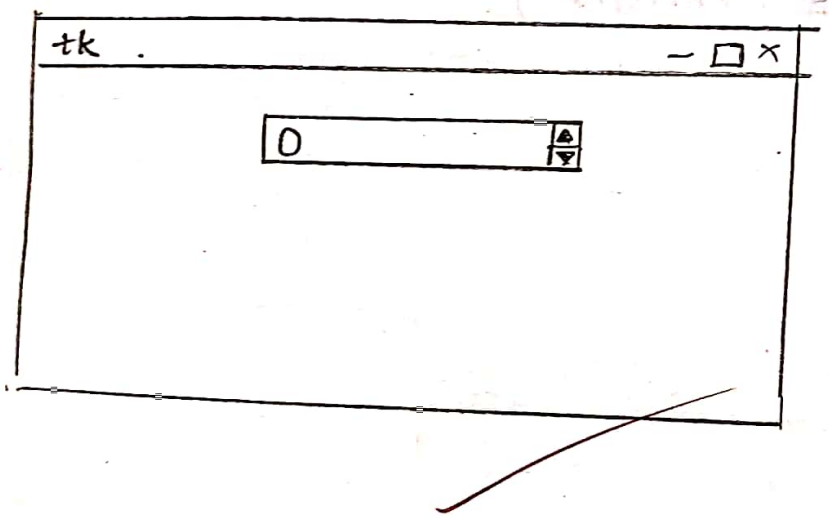
* Spinbox Widget

Step 1: Create an object from the TK() and subsequently create an object from the spinbox.

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

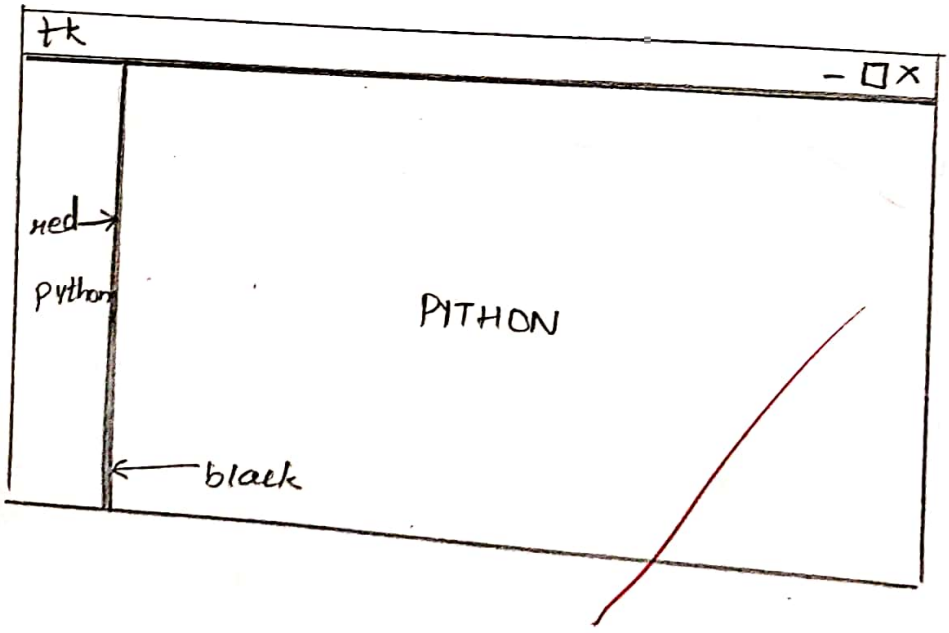
```
from tkinter import *
master = Tk()
s = spinbox (master, from_ = 0, to = 10)
s.pack ()
master.mainloop()
```

Output:



```
from tkinter import *
root = Tk()
p = PanedWindow(bg="red")
p.pack(fill=BOTH, expand=10)
l = Label(p, text="python")
p.add(l)
p1 = PanedWindow(p, orient=VERTICAL, bg="black")
p.add(p1)
l1 = Label(p1, text="PYTHON")
p1.add(l1)
root.mainloop()
```

Output:



* Paned Window Widget

Step 1: Create an object from panedwindow() and use the pack() with attribute fill and expand.

Step 2: Create an object from the label method and put it onto the panedwindow with the text attribute and use the add method to embed the new object.

Step 3: Similarly create a second panedwindow object and add it onto the first panedwindow with orientation specified.

Step 4: Now create another label object and place it onto the second panedwindow object and add it onto the second panedwindow. Trigger the mainloop

* Canvas Widget.

Step 1: Create an object from the canvas method and use the attribute height, width, bg and parent window objects.

Step 2: Use the method create line, create oval and create arc along with the canvas object so created and use the co-ordinate values

Step 3: Similarly use the other method and call the pack method and the mainloop method.

```
from tkinter import *
root = Tk()
c = Canvas(root, height=200, width=250, bg="pink")
carc = c.create_arc(10,20,30,40, start=0,
    extent=100, fill="red")
cline = c.create_line(50,60,70,80, fill="black")
coval = c.create_oval(90,100,110,120, fill="green")
c.pack()
root.mainloop()
```

```
>>> import dbm
>>> db = dbm.open("database", flag='c',
        mode=438)
>>> db["name"] = "Python"
>>> if db["name"] != None:
        print(" Database not found")
    else:
        print ("Database found ")

>>> db.close()
```

Output:

database not found

PRACTICAL-6

Aim: Database connectivity

Algorithm:

a. Step 1: Import the dbm library and use open() for creating the database by specifying the name of the database along with the corresponding flag.

Step 2: Use the object so created for accessing website and corresponding regular name for website.

Step 3: Check if the url address matches with regular name is not equal to none than display the message that particular found/matched or else not found/unmatched

Step 4: Use the close() terminate database library.

b. Step 1: Import corresponding library to make database connection, os & SQL lite-3

Step 2: Now create the connection object using SQL lite-3 library and the connect() for creating new database.

Step 3: Now create cursor object using the cursor() and from the connection object created.

Step 4: Now use the execute() for creating the table with the column name and respective datatype.

Step 5: Now with cursor-object use the insert statement for entering the values corresponding to different fields, corresponding to datatype.

Step 6: Use the commit() to complete the transaction using the connection object.

```
import os, sqlite3
connection= sqlite3.connect("emp.db")
cursor= connection.cursor()
cursor.execute('create table dos( Name CHAR,
  RollNo INT')
cursor.execute("insert into dos values
    ('ABC', 24), ('XYZ', 14)")
connection.commit()
cursor.execute('select * From dos')
print( cursor.fetchall())
 connection.close()
```

Output:

```
[('ABC', 24), ('XYZ', 14)]
```

Step 7: Use the execute statement along with cursor-object for accessing the values from the database using the select/pat From/ where clause.

Step 8: Finally use the fetch() or fetchall() for displaying the values from the table using the cursor-object.

Step 9: Execute() and drop table syntax for terminating the database & finally use the close()