# PRACTICAL - 1

OBJECTIVE : 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 than 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

```
fileobj = open("abc.txt", "w") # file open (write mode)
fileobj.write("computer science subjects" + "\n")21
fileobj.write("DBMS \n PYTHON \n 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 outputs of read method:', 'computer science
    subjects \n DBMS \n PYTHON \n 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 science
    subjects \n')

# readlines()
fileobj = open("abc.txt", "r")
str3 = fileobj.readlines()
print("The output of readlines method: ", str3)
fileobj.close()
>>> ('The output of readlines method:', ['computer
    science subjects \n', 'DBMS \n', 'PYTHON \n', 'DS \n'])
# file attributes
a = fileobj.name
print("name of file (name attribute): ", a)
>>> ('name of file (name attribute)', 'abc.txt')
b = fileobj.closed
print("(close) attribute : ", b)
>>> ('(close) attribute :', 'True')
```

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

```
# w+ mode
fileobj = open ("abc.txt", "w+")
fileobj.write ("loukik sir")
fileobj.close()
```

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

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

```
# r+ mode
fileobj = open ("abc.txt", "r+")
s1 = fileobj.read(6)
print ("output of r+", s1)
fileobj.close()
>>> ('output of r+', 'loukik')
```

```
# append mode
fileobj = open ("abc.txt", "a")
fileobj.write ("data structure")
fileobj.close()
fileobj = open ("abc.txt", "r")
s3 = fileobj.read()
print ("output of append mode:", s3)
fileobj.close()
>>> ('output of append mode:', 'loukik sir data structure')
```

step4 : Now open the fileobj in write mode write some another content close subsequently. Then again open the file obj in 'w+' mode that is the update mode and write contents

step 5 : open fileobj in read mode display the update written contents 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 contents close the fileobj again open the fileobj in read mode and display the 'appending' output.

step 7: open the fileobj in read mode, declare a variable and perform fileobject dot tellmethod 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 readlines method and store the output consequently in and print the same for counting the length use the 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)
file obj = open("abc.txt", "r")
st= fileobj.seek(0,1)
print("seek(0,1) is : ", st)
fileobj.close()
>>> ('seek(0,1) is :', None)
fileobj = open("abc.txt", "r")
st2 = fileobj.seek(0,2)
print("seek(0,2) is : ", st2)
fileobj.close()
>>> ('seek(0,2) is :', None)
# finding length of different lines exist
  within lines
fileobj = open("abc.txt", "r")
stat = fileobj.readlines()
print("output : " stat)
for line in stat:
        print(len(line))
file obj.close()
>>>('output : ', ['Noukik sir data structure ']
```

31/2/17

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

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

```
# for loop
mytuple1 = ("Kevin", "stuart", "bob")
for x in mytuple1:
    print(x)
```

```
>>> Kevin
    stuart
    bob
```

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

# PRACTICAL - 2

OBJECTIVE : Iterators

step 1 : create a 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 we will get the next iterating element in the tuple. Display the same.

step 2 : The similar output can be obtained by using for conditional statment. An iterable 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 fashion declare cube to get the value raised 3. and return the same.

step 4 : call the declared function using function call.

step 5: using for conditional statement specifying the range use the list typecasting with map method declare a 'lamda' i.e annonymus function. and print the same.

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

step 7: define a function even 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 initialise 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(lamluda x: x% 5, listnum))
print(listnum)
def even(x):
    if (x%.2 ==0):
        return "EVEN"
    else:
        return "ODD".
list(map(even, listnum)) >>> [0,4,5,7,9,11, 13,15,20,
                               19,25]
# 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
```

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

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

# factorial using map function:

```
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: 4
('The factorial is: ', [24])
```

step 10 : Define an object of a class.

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

10/12/18

step 12 : Define a function with if condition if exactly equal to one return else one parameter

step 13 : Enter a input from the declare an empty list. use append method. for appending the input value.
call the map using two parameter print the factorial for input variable.

# PRACTICAL-3

TOPIC: Exceptions

step 1: In the try block open an file in the open mode with write mode. write some contents in file

step 2: In except (IOError) block use the error in IOError use the appropriate message to display.

step 3: Else display the operation is successful!

step 4: In try block accept an input from the user

step 5: In except block use valueerror and print the same message.

step 6: Else display the operation is successful!

# IOError:

```
try:
    fo = open ("abc.txt", "w")
    fo.write ("python is an intendent language")
except IOError:
    print (" Enter appropriate mode for file operation!")
else:
    print (" operation is successful!")
```

#R

>>> operation is successful!

#R output:

>>> Enter appropriate mode for file operation!

# ValueError:

```
try:
    x = int (input ("Enter a statement: "))
except ValueError:
    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 = 1
b = input ("Enter: ")
try:
    print (a/b)
except TypeError:
    print ("Incompatible value")
except ZeroDivisionError:
    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 ('10'+10)
except (TypeError, ZeroDivisionError):
    print ("Invalid Input!")
```

```
>>> 1.0
>>> 1010
>>> Invalid Input!
```

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

step 8 : For the exception to be raised use the except keyword (and) i.e TypeError print Incompatible values.

step 9 : Use except with type 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 errors types by separating them with a comma.

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

step 13: use the IOError 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 ("python")
except IOError:
        print ("Error!")
else:
    print successful
def x():
    l=[]
    print (len (l))
def y():
    li = [2,4,4,1]
    print (len(li))
print(x())
print (y())
```

output: successful
0
None
4
None

# raise keyword:

```
try:
    a= int (input ("Enter a number: "))
    raise valueError
except valueError:
    print ("Enter integer value!")
```

>>> Enter: XYZ
Enter integer values!

# 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 (segreation)

```
import re
pattern = r'\d+'
string = 'hello123, howdy789, 45howru'
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']

# PRACTICAL - 4

TOPIC : Regular expression.

step 1 : 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!

step 2 : Import re module declare pattern with literal and meta charater. Declare string value. use the findall () with arguments and print the same.

step 3 : Import re module declare pattern with meta charater use the split ( ) and print the output.

step 4: import re module declare string and accordingly declare pattern replace the ~~no~~ blank space with no-space. use sub() with 3 arguments. and print the string without spaces.

step 5: import re module declare a sequence use search method for finding subsequently use the group() with dot operator as search() gives memory location using group() it will show up the matched string.

step 6: import re module ~~gi~~ 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 ~~rest~~ next number are in range of 0 to 9 and check whether the entered numbers are equal to 10. if criteria matches print can number matches otherwise print failed.

```
# 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 interesting language'
v = re.search('\Apython', 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', '7865432981',
         '9876543201']
for value in list1:
    if re.match(r'[8-9]{1}[0-9]{9}',
     value or len(value)==10):
        print("criteria matched for cell number!")
    else:
        print("criteria failed!")
```

```
>>> criteria matched for cell number
    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:
```

step 7: import re module declare a string use the module with findall() for finding the vowels in the string and declare the same.

step 8: import re module declare the host and domain name declare pattern for seperating the host & domain name. use the finddll() and print the output respectively.

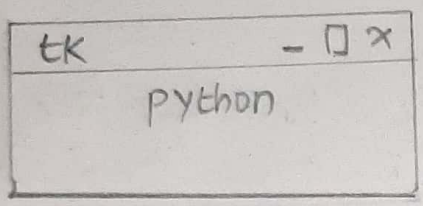
step 9: import re module enter a string use pattern to display only two elements of the particular string. use findall() declare two variables with initial value as zero use for condition and subsequently use the if condition check whether condition satisfy add up the or else increment value. and display the values subsequently.

```
if (v=='ms'):
    f=f+1
else:
    m=m+1
print ("NO. of males is:", m)
print ("NO of females is:", f)
>>> ['mr', 'ms', 'ms', 'mr']
('NO. of males is:', 2)
('NO. of females is:', 2)
```

# creation of parent window

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

output:




#2: label, attributes

```
from Tkinter import *
root TK()
l= label (root, text="python")
l.pack ()
l1= Label (root, text="CS1", bg="grey",
      fg="black", font="10")
l1.pack (side = LEFT, padx=20)
l2 = Label (root, text = "cs", bg = "light blue",
      fg - "black", font = "20")
l2.pack(side = LEFT, pady="30")
l3 = Label (root, text = "CS1", bg= yellow",
      fg ="black", font ="10")
l3.pack(side = TOP, ipadx =40)
```

# PRACTICAL - 5

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 importing the features of the text widget.

step 2: create a variable from the text method and position it on the parent window.

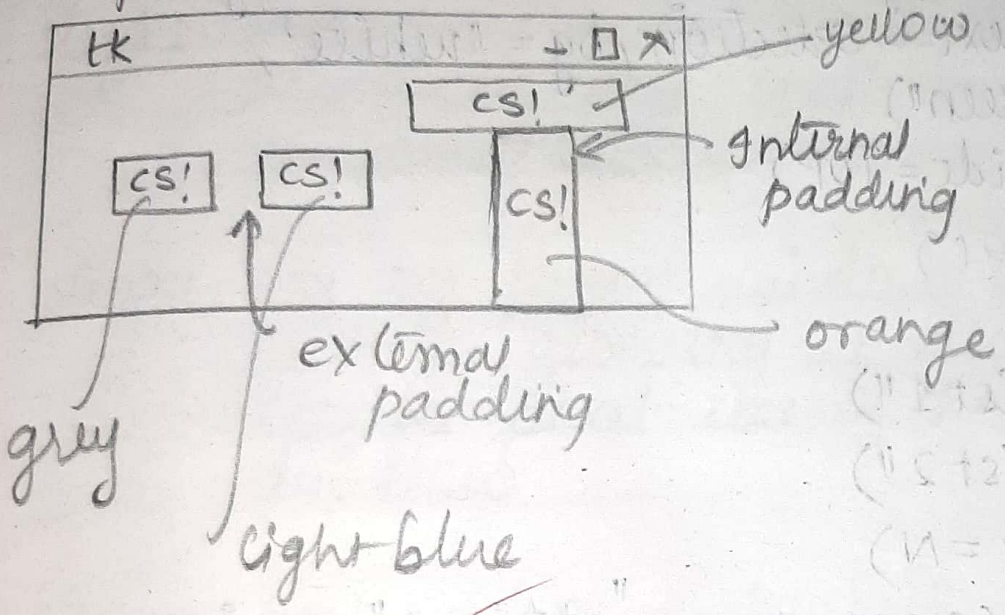
step 3: Use the pack() along with the object created from the text() and use the parameter
1) side = LEFT, padx = 20
2) side = LEFT, pady = 30
3) side = TOP, ipadx = 40
4) side = TOP, ipady = 50

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

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

```
l4= Label(root, text="CS!", bg="orange",
    fg="black", font="10")
l4.pack(side=TOP, ipady=50)
root.mainloop()
```

output:




14/5

#1:

```
# Radio button
from tkinter import *
root = TK()
root.geometry("500X500")
def select():
    selection = " you just selected "+ str(var.get())
    t1 = Label (text = selection, bg = "white",
        fg = "green")
    t1.pack(side = TOP)
var = stringvar()
l1 = Listbox()
l1.insert (1, "List 1")
l2.insert (2, "List 2")
l1.pack (anchor =N)
r1 = RadioButton (root, text = "option 1", variable
    =var, value = " option 1", command =select)
r1.pack (anchor=N)
r2 = RadioButton (root, text =" option 2", variable =
    var, value = "option 2", command = select)
r2: pack (anchor=N)
root.mainloop ()
```

# PRACTICAL - 5 (B)

AIM: Gui components

#1:

step1: Import the relevant methods from the tkinter library. create an object with the parent window.

step2: use the parent window object along with the geometry () declaring specific pixel size of the parent window.

step3: Now define a function which tells the user about the given selection made from multiple option available.

step4: Now define the parentwindow and define the option with control variable.

step5: use the listbox () and insert options on the parent window along with the pack () with specifying anchor attribute

step6: create an object from radio button which will take following arguments ( parentwindow object, text variable which will take the values option no 1, 2, 3... variable argument, corresponding value &

step6: trigger the function declared..

step7: Now call the pack() for radio object so created and specify the argument using anchor attribute.

step8: Finally make use of the mainloop() along with parent object

#2:

step1: Import relevant methods from the Tkinter library

step2: create a parent object corresponding to the parent window

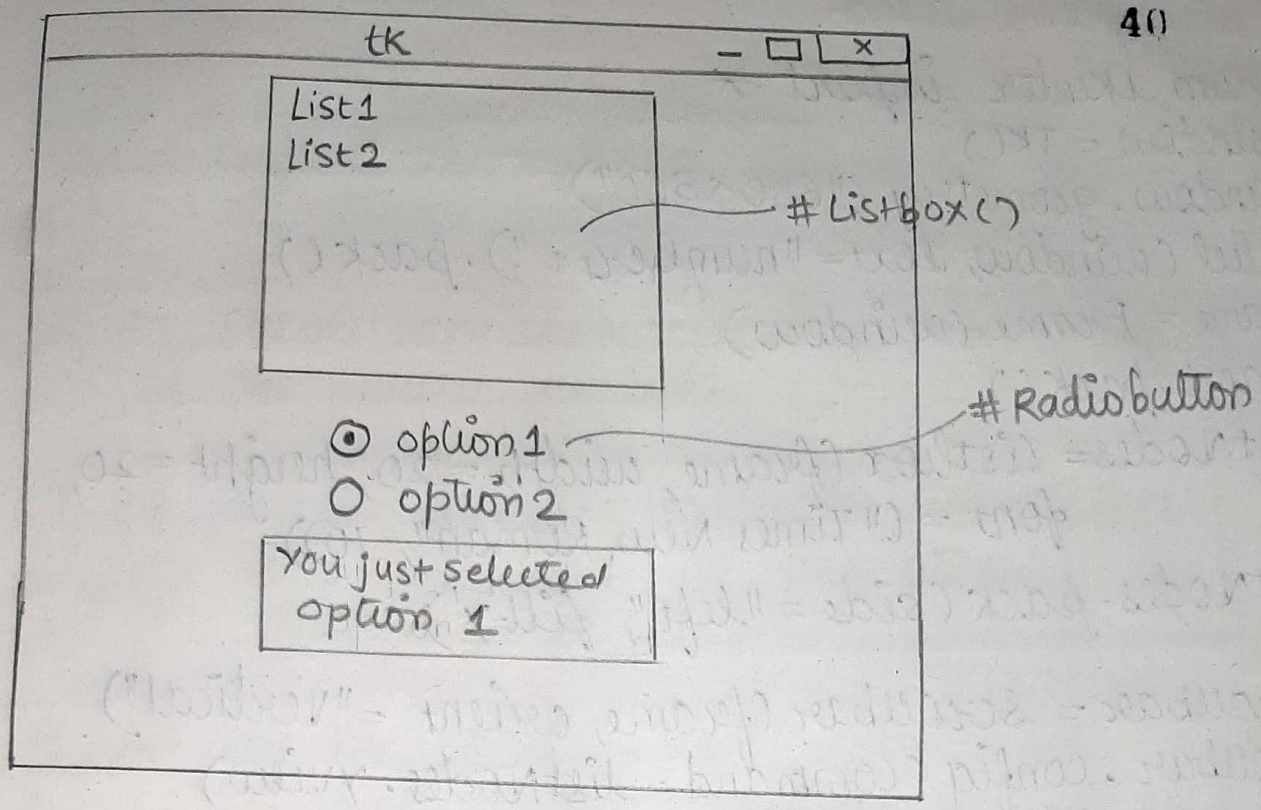
step3: use the geometry() for laying of the window

step4: create an object and use the scrollbar()

step5: Use the pack() along with the scrollbar object with side and fill attributes

step6: Use the mainloop with the parent object.

output:

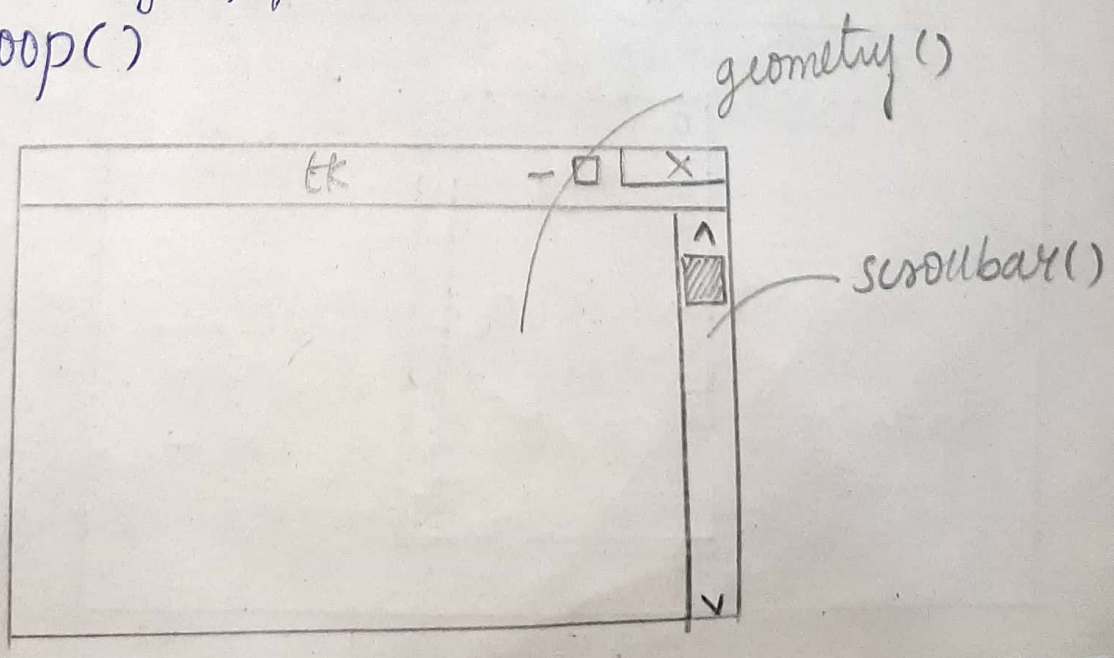



#2:

Scrollbar()

```
from tkinter import*
root = TK()
root.geometry("500x500")
s = scrollbar()
s.pack(side="right", fill="y")
root.mainloop()
```

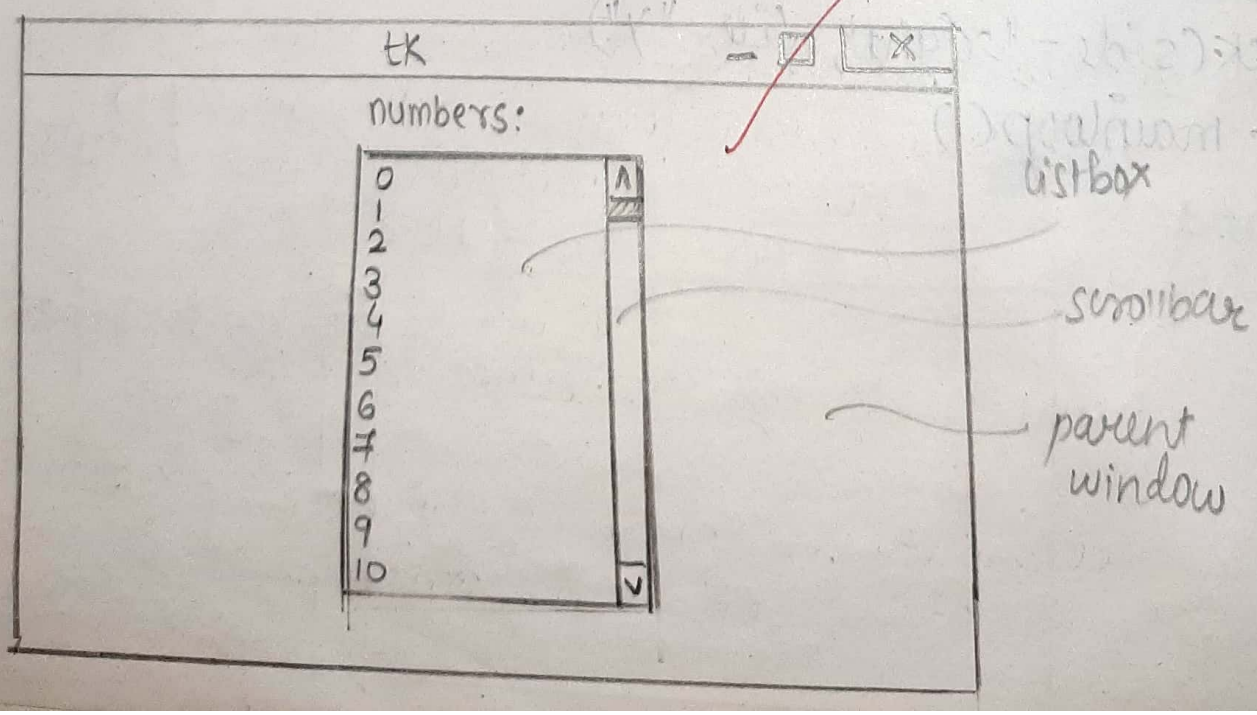
output:



#3:

```
# using frame widget
from tkinter import *
window = TK()
window.geometry ("680×500")
Label (window, text="numbers:").pack()
frame = Frame (window)
frame.pack()
listNodes= Listbox (frame, width =20, height =20,
                font = ("Times New Roman", 10))
listNodes.pack (side ="left", fill='y')

scrollbar = scrollbar (frame, orient ="vertical")
scrollbar.config (command = listNodes.yview)
scrollbar.pack (side = "right", fill = "y")
for x in range (100):
    listNodes.insert (END, str(x))
window.mainloop ()
```

Output:



# 3:

step 1: Import the relevant libraries from the tkinter method

step 2: create an corresponding object of the parent window.

step 3: use the geometry manager with pixel size (680 X 500) or any other suitable pixel value.

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 the parent object created and use the pack method.

step 6: use the listbox method along with the attributes like width, height font. Do create a listbox method's object. use pack() for the same.

step 7: use the scrollbar() with an object use the attribute of vertical. then configure the same with object created from the scrollbar() and use pack()

step 8: Trigger the events using mainloop.

#4:

step 1: Import relevant methods from tkinter library

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

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

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

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

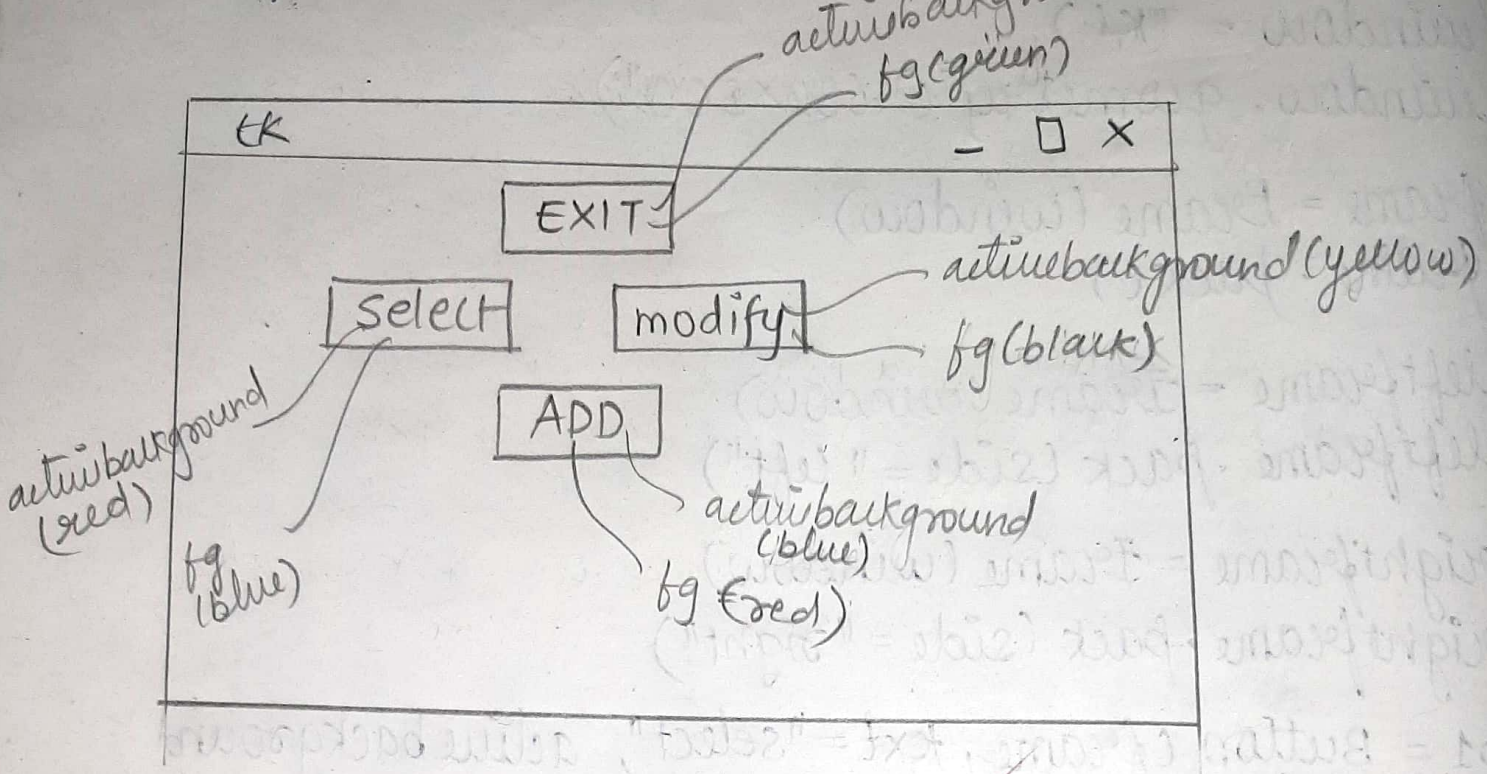
step 6: Now use the pack() along with the side attribute

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

#4:

```
from tkinter import *
window = TK()
window.geometry("680x500")

frame = Frame(window)
frame.pack()

leftframe = Frame(window)
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", activebackground
    ="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")
```

Output :


tk
EXIT
activebackground (red)
fg (green)
select
modify
activebackground (yellow)
fg (black)
ADD
activebackground (red)
fg (blue)
activebackground (blue)
fg (red)

step 8: create another button object & place it on the RIGHT frame & label the button as ADD.

step 9: Add another button & puts 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().

21/1

# PRACTICAL -5(C)

AIM: Gui components

step 1: Import the relevant methods from tkinter library

step 2: Import tkMessagebox.

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

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

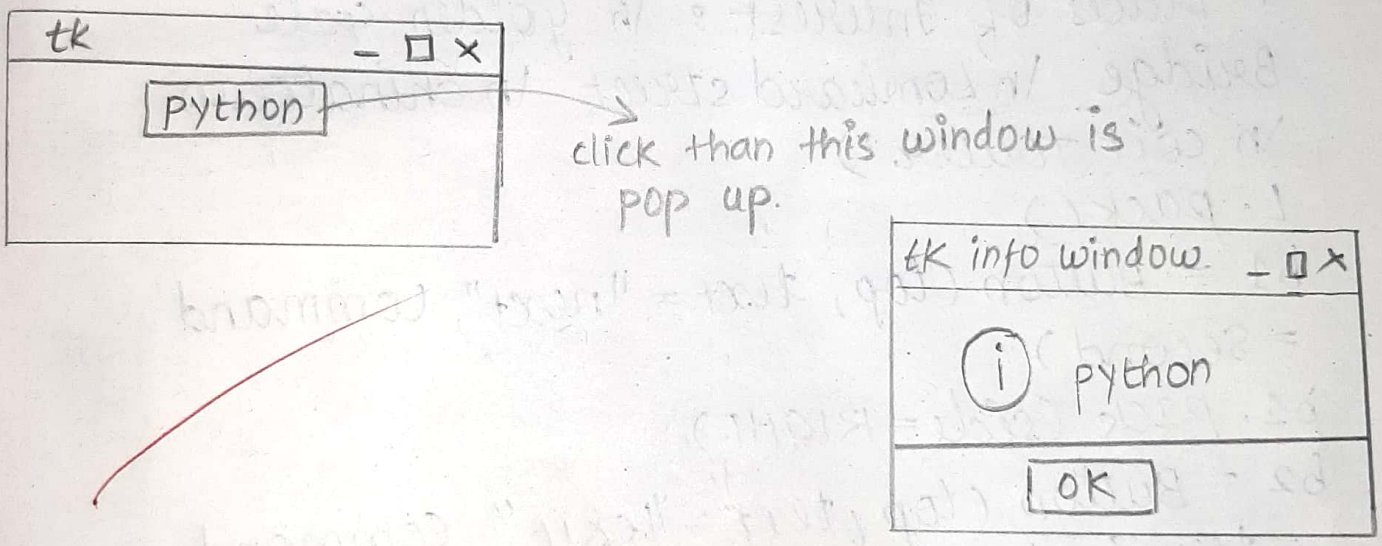
step 5: Declare a button with parent window object along with the command attribute

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

# message box

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

Output:



# Multiple window
# Different button (Relief())

```
from TKinter import *
root = TK()
root.minsize (300,300)

def main():
    top = TK()
    top.config (bg = "black")
    top.title ("HOME")
    top.minsize (300,300)
    L= Label (top, text =" SAN FRANCISCO
    \n Places of Interest : \n Golden gate
    Bridge \n Lombard street \n chinatown
    \n coit Tower ")
    L.pack()
    b1 = Button (top, text = "next", command
    = Second)
    b1.pack (side = RIGHT)
    b2 = Button (top, text = "exit", command
    = terminate)
    b2.pack( side = LEFT)
    top.mainloop()
```

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

step 2 : use parentwindow object along with minsize function for window size.

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

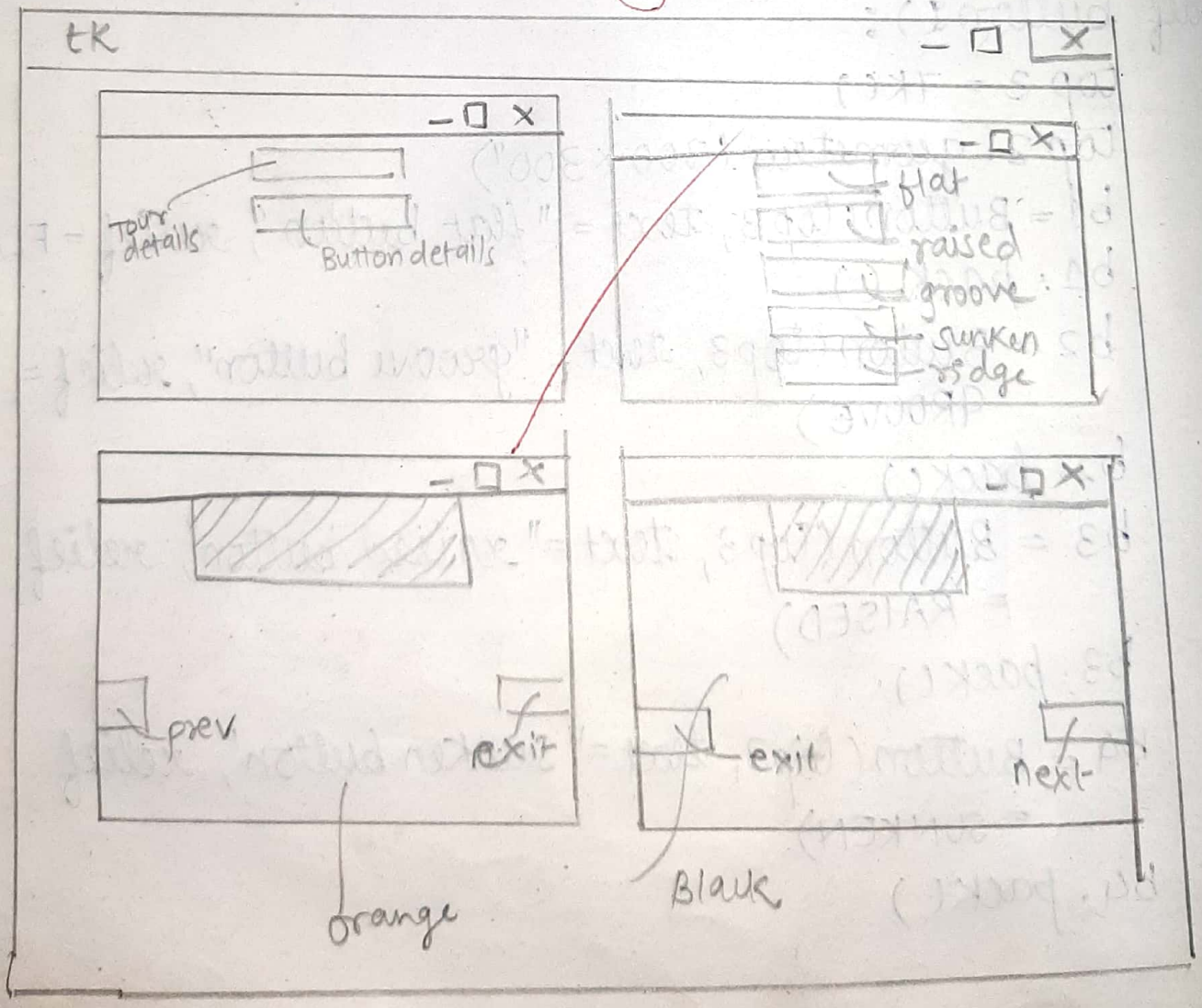
step 4 : similary define the function second and use the attributes accordingly.

step 5 : Declare another function button along with parent object and declare button with attributes like ~~re~~ FLAT, RIDGE GROOVE, RAISED, 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: Tanvi Mane \n
    For more details contact to our official account")
    L.pack()
    b3 = Button(top2, text = 'prev', command = main)
    b3.pack(side = LEFT)
    b2 = Button(top2, text = "exit", command =
    terminate)
    b2.pack(side = RIGHT)
    top2.mainloop()

def button():
    top3 = TK()
    top3.geometry("300X300")
    b1 = Button(top3, text = "flat button", relief = FLAT)
    b1.pack()
    b2 = Button(top3, text = "groove button", relief =
        GROOVE)
    b2.pack()
    b3 = Button(top3, text = "raised button", relief
        = RAISED)
    b3.pack()
    b4 = Button(top3, text = "sunken button", relief
        = SUNKEN)
    b4.pack()
```

```
b5 = Button (top3, text = "ridge button",
relief = RIDGE)
top3. mainloop()
def terminate():
b5 quit()

b5 = Button (root, text = "TOUR DETAILS",
    command = main)
b5. pack()
b6 = Button (root, text = "BUTTON DETAILS",
    command = button
b6. pack()
root. mainloop()
```



# PRACTICAL - 5(D)

AIM: Gui components

step 1: Import relevant methods from the tkinter library.

step 2: create parent window object and use the config method along with background color attribute specified.

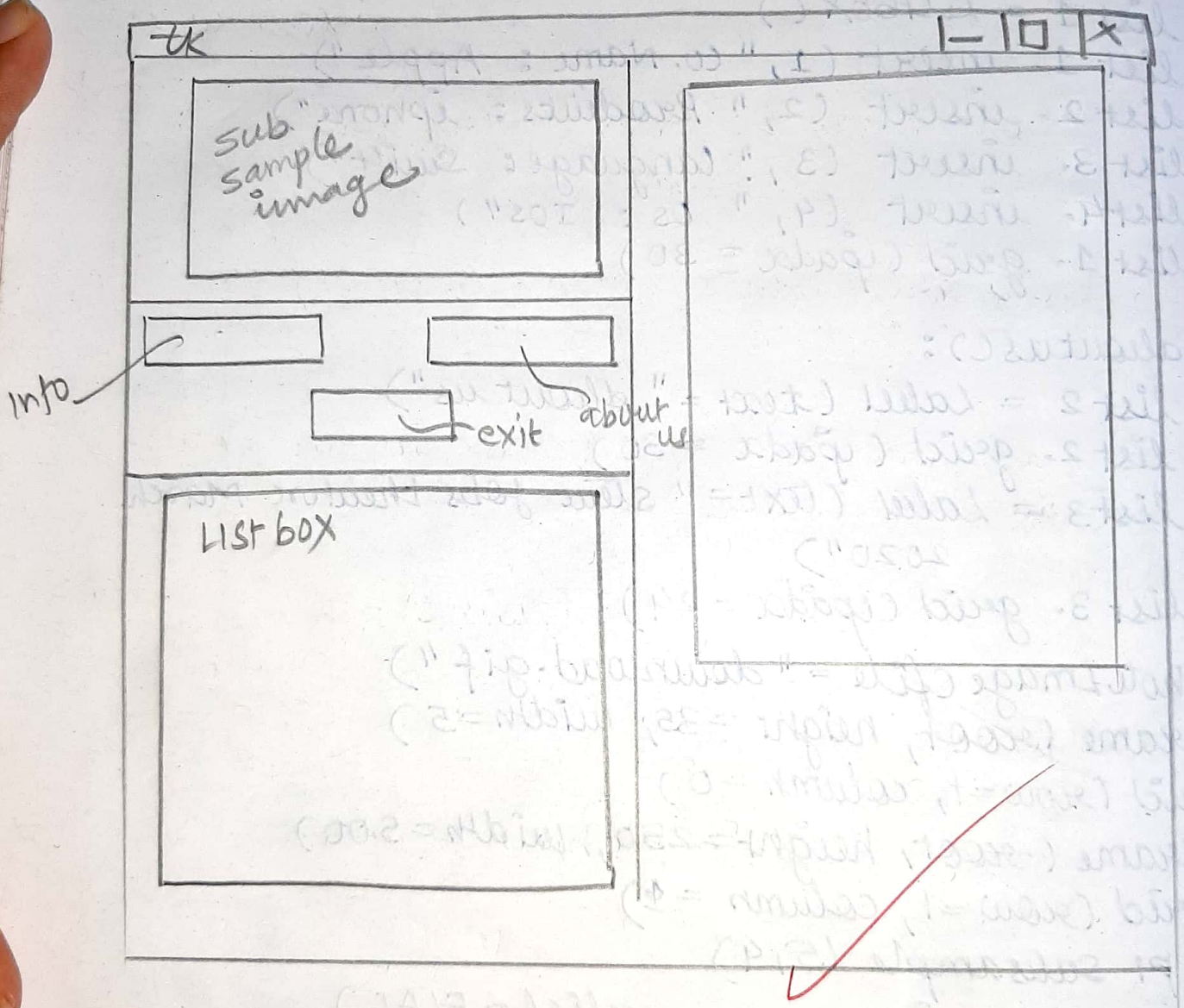
step 3: define a function finish with the messagebox widget which will display a message i.e a warning message and subsequently terminate the program.

step 4: define a function info use a listbox widget along with the object of the same. use the listbox object along with insert method and insert the same and finally use the grid() with padx attribute

step 5: define a function about us with label widget and text attribute and subsequently use the grid().

```
from Tkinter import *
root = TK()
root = config (bg = "grey")
def finish():
    messagebox.askOKcancel ("Warning", This will
    end the program")
    quit()

def info():
    list 1 = Listbox()
    list 1. insert (1, " Co. Name : Apple ")
    list 1. insert (2, " Products : iphone")
    list 1. insert (3, " language : Swift")
    list 1. insert (4, " OS : IOS")
    list 1. grid (padx = 30)

def aboutus():
    list 2 = Label (text = " about us")
    list 2. grid (padx = 30)
    list3 = Label (text = " steve jobs theatre March
            2020")
    list 3. grid (padx = 24)
p1 = PhotoImage (file = " download.gif ")
f1 = Frame (root, height = 35, width = 5)
f1. grid (row = 1, column = 0)
f2 = Frame (root, height = 250, width = 500)
f2. grid (row = 1, column = 1)
p2 = p1. subsample (5, 4)
l1 = Label (f1, image = p2, relief = FLAT)
l1. grid (row = 1, column = 0, padx = 20, pady = 15)
l2 = Label (f2, image = p1, relief = SUNKEN)
l2. grid (padx = 25, pady = 10)
b1 = Button (f1, text = " Information ", relief = SUNKEN,
        command = info)
b1. grid (row 1, column = 0)
```

```
b2= Button (f1, text = " About us", relief=SUNKEN,
    command = 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)
root.mainloop ()
```



step6: use photoImage widget with file and filename with gif attribute.

step7: create a frame object along with the Frame() along with parent window object height & width specified. and subsequently use the grid() with row & column attribute specified.

step8: similarly create another frame object as declard by step7.

step9: create another object & use the subsample (5/4).

step10: use label widget along with the frame object, relief attribute. and subsequently use the grid().

step11: Now create button object dealing with different section of frame.

# PRACTICAL - 5(E)

AIM: Gui components

#1: SPINBOX:

step1: Import relevant ()'s from the tkinter library.

step2: create parent window object along with TK().

step3: create an object from spinbox method and use attributes parent window object, from_ and to attributes)

step4: subsequently call the pack method along with spinbox object & called the mainloop().

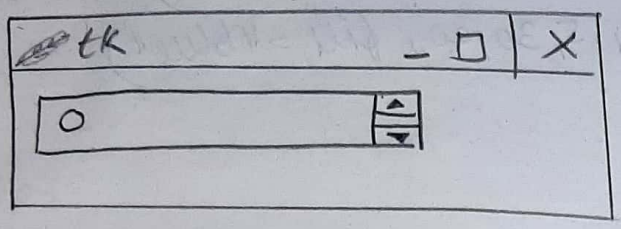
#2: PANED WINDOW

step1: Import relevant methods from the tkinter library also create a parent window object.

step2: create an object along with paned window and subsequently use the

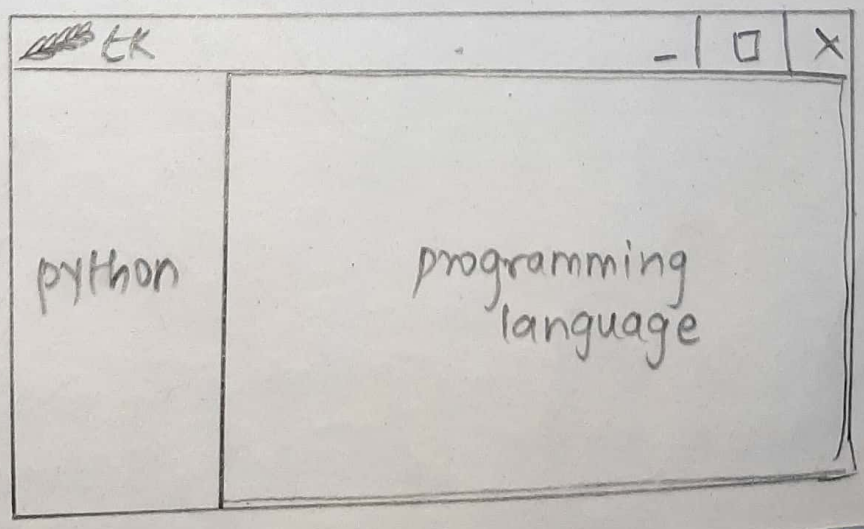
```
from Tkinter import *
master = TK()
s= Spinbox (master, from_ =0, to 10)
s. pack( )
master. mainloop ( )        # spinbox widget
```

output:



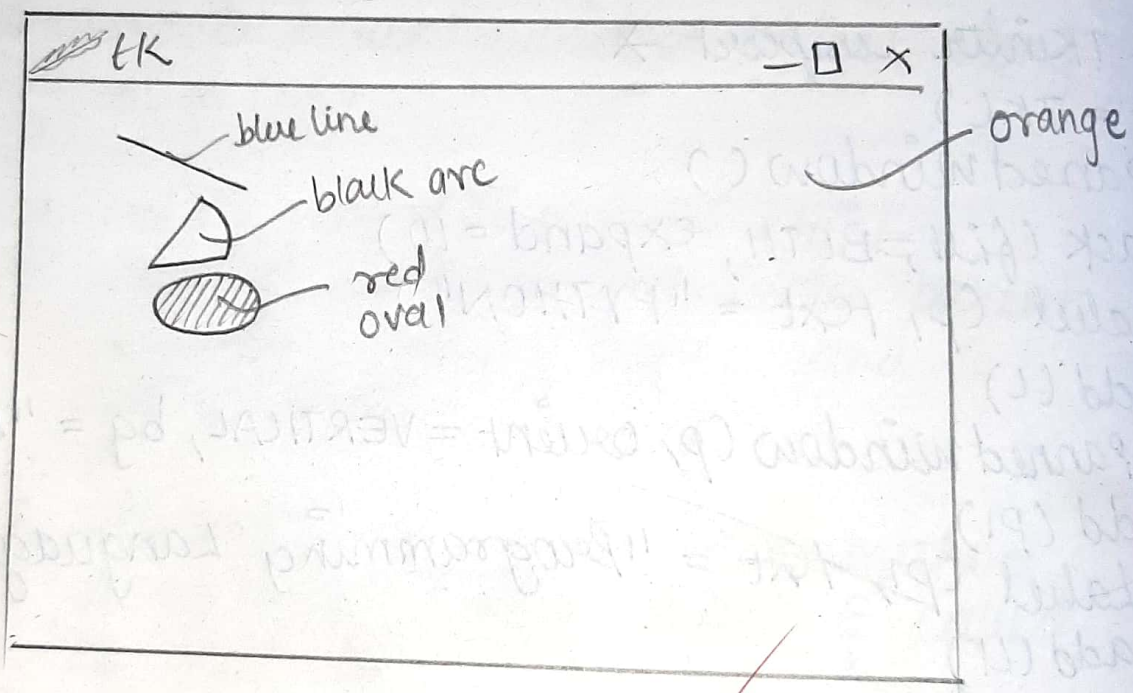

```
from Tkinter import *
root = TK( )
p= Panedwindow ( )
p. pack (fill=BOTH, expand = 10)
l= Label (p, text = "PYTHON")
p. add (L)
p1 = Panned window (p, orient =VERTICAL, bg = "black")
p. add (P1)
l1= Label (p1, text = "Programming Language")
p1. add (L1)
root. mainloop ( )
```

output:



#3

```
from Tkinter import *
root = TK()
c = canvas (root, height = 100, width = 200,
    bg = "orange")
carc = c.create_arc (10,20,30,40, start = 10,
        extent = 50, fill = "black")
coval = c.create_oval (20, 31, 45, 50, fill = "red")
cline = c.create_line (10, 15, 30, 20, fill = "blue")
c.pack()
root.mainloop()
```




canvas

pack() along with the panned window object along with attributes like fill & expand.

Step3: create an Label object along with Label() use panned window object, orient & background color.

Step4: Similarly create anothe Label method and use the add() subsequently.

Step5: Finally called the mainloop() for event trigreation.

#3: CANVAS:

Step1: Import relevant ()'s from tkinter library declare a parent window object.

Step2: create a canvas object along with the canvas() with attributes parent window object, heigh, width, background color.

Step3: use create_arc() along with canvas object declared along with start & extent & co-ordinates.

Step4: Similarly for oval & line use the pack() and called mainloop() for event driven programming.

#1:

```
>>> import dbm
>>> db = dbm.open ("database", flag='c', mode=438)
>>> db["name"] = "name"
>>> if db["name"] != None:
        print("database not empty // match!")
    else:
        print("database empty! " // NOT match")
>>> match
>>> db.close()
```

# PRACTICAL - 6

AIM: Database connectivity

#1:

step 1: Import the (DBM) dbm library & use the 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 the given website & corresponding regular name for the website

step 3: check whether the given url address matches with the regular name of the page is not equal to none than display the message that particular found / match or else not found / unmatched.

step 4: use the close() to terminate database library

#2: 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 & the connect() for creating new database.

step 3: Now create cursor object using the cursor() & from the connection object created.

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

step 5: Now with cursor_object use the insert statement for entering the values corresponding to different fields, corresponding the datatype.

step 6: use the commit() to complete the transaction using the connection object

step 7: use the execute statement along with cursor_object for accessing the values from the database using the select from where clause.

#2:
```
import os, sqlite3
conn = sqlite3.connect("employee.db")
cur = conn.cursor()
cur.execute('create table dos (Name char,
 ROLLNO int')
cur.execute("insert into dos values ('Tanvi', 1712),
 ('Tanmay', 1217)")
conn.commit()
cur.execute('select* from dos')
print (cur.fetchall())
conn.close()
```

output:

[('Tanvi', 1712), ('Tanmay', 1217')]

step 8: Finally use the fetch() or fetchall() for displaying the values from the table using the cursor_object.

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

25/12

# PRACTICAL
## PROJECT - 1

Database connectivity.

Synopsis: I made this project in order to show that how python is has platform portability with the database operations.
This project deals with database management of employee's database
Here by we have entered the data values, secondly we have displayed values updated the values & deleted & truncated values.
This project says how can ~~be~~ we do the database operations with ease with python.

**SOURCE CODE:**

```
#connection of sqlite3 with python
import sqlite3
conn = sqlite3.connect('test.db')
print( "Opened database successfully");

#creation of table
conn.execute('''CREATE TABLE COMPANY_11
(ID INT PRIMARY KEY NOT NULL,
NAME TEXT NOT NULL,
AGE INT NOT NULL,
ADDRESS CHAR(50),
SALARY REAL);''')
print ("Table created successfully");
conn.close()

#insert into table values
conn = sqlite3.connect('test.db')
print ("Opened database successfully");
conn.execute("INSERT INTO COMPANY_11 (ID,NAME,AGE,ADDRESS,SALARY) \
VALUES (1712, 'Tanvi', 20, 'California', 220000.00 )");
conn.execute("INSERT INTO COMPANY_11 (ID,NAME,AGE,ADDRESS,SALARY) \
VALUES (1711, 'Tanmay', 25, 'Texas', 115000.00 )");
conn.execute("INSERT INTO COMPANY_11 (ID,NAME,AGE,ADDRESS,SALARY) \
VALUES (1710, 'Nikhil', 23, 'Norway', 110000.00 )");
conn.execute("INSERT INTO COMPANY_11 (ID,NAME,AGE,ADDRESS,SALARY) \
VALUES (1713, 'Atharav', 27, 'New York ', 115000.00 )");
conn.commit()
print ("Records created successfully");
conn.close()

#displaying of the values using the select clause
conn = sqlite3.connect('test.db')
cursor = conn.execute("SELECT id, name, address, salary from COMPANY_11")
for row in cursor:
   print ("ID = ", row[0])
   print ("NAME = ", row[1])
   print ("ADDRESS = ", row[2])
   print ("SALARY = ", row[3])
   print ("\nOperation done successfully");
conn.close()
```

```
#UPDATE select from where clause
conn = sqlite3.connect('test.db')
print ("UPDATION OF VALUES");
conn.execute("UPDATE COMPANY_11 set SALARY = 220000.00 where ID=1712")
conn.commit
print ("Total number of rows updated :", conn.total_changes)
cursor = conn.execute("SELECT id, name, address, salary from COMPANY_11")
for row in cursor:
    print ("ID = ", row[0])
    print ("NAME = ", row[1])
    print ("ADDRESS = ", row[2])
    print ("SALARY = ", row[3])
    print ("\nOperation done successfully");
conn.close()

#DELETION using where clause
conn = sqlite3.connect('test.db')
print ("DELETION");
conn.execute("DELETE from COMPANY_11 where ID=1713;")
conn.commit
print ("Total number of rows deleted :", conn.total_changes)
cursor = conn.execute("SELECT id, name, address, salary from COMPANY_11")
for row in cursor:
    print "ID = ", row[0]
    print "NAME = ", row[1]
    print "ADDRESS = ", row[2]
    print "SALARY = ", row[3], "\n"
    print "Operation done successfully";
conn.close()


#TRUNCATE DDL statement
conn = sqlite3.connect('test.db')
cursor = conn.execute("DROP TABLE company_11");
print("\n VALUES truncated!")
conn.close
```

**OUTPUT:**

```
Opened database successfully
Table created successfully
Opened database successfully
Records created successfully
('ID = ', 1712)
('NAME = ', u'Tanvi')
('ADDRESS = ', u'California')
('SALARY = ', 220000.0)
```

```
Operation done successfully
('ID = ', 1711)
('NAME = ', u'Tanmay')
('ADDRESS = ', u'Texas')
('SALARY = ', 115000.0)

Operation done successfully
('ID = ', 1710)
('NAME = ', u'Nikhil')
('ADDRESS = ', u'Norway')
('SALARY = ', 110000.0)

Operation done successfully
('ID = ', 1713)
('NAME = ', u'Atharav')
('ADDRESS = ', u'New York ')
('SALARY = ', 115000.0)

Operation done successfully
UPDATION OF VALUES
('Total number of rows updated :', 1)
('ID = ', 1712)
('NAME = ', u'Tanvi')
('ADDRESS = ', u'California')
('SALARY = ', 220000.0)

Operation done successfully
('ID = ', 1711)
('NAME = ', u'Tanmay')
('ADDRESS = ', u'Texas')
('SALARY = ', 115000.0)

Operation done successfully
('ID = ', 1710)
('NAME = ', u'Nikhil')
('ADDRESS = ', u'Norway')
('SALARY = ', 110000.0)

Operation done successfully
('ID = ', 1713)
('NAME = ', u'Atharav')
('ADDRESS = ', u'New York ')
('SALARY = ', 115000.0)
```

```
Operation done successfully
DELETION
('Total number of rows deleted :', 1)
ID =  1712
NAME =  Tanvi
ADDRESS =  California
SALARY =  220000.0

Operation done successfully
ID =  1711
NAME =  Tanmay
ADDRESS =  Texas
SALARY =  115000.0

Operation done successfully
ID =  1710
NAME =  Nikhil
ADDRESS =  Norway
SALARY =  110000.0

Operation done successfully

VALUES truncated!
```

# PRACTICAL

## Project-2

synopsis: GUI (graphical user Interface)
This project made is for showing up the apple products and details about the apple industry.
Different widgets are used:

a) photo Image
b) subsample
c) Label
d) spinbox
e) Drawing of windows
f) button
g) background colors
h) size
i) Listbox

Just a start of using of python widgets more to come!

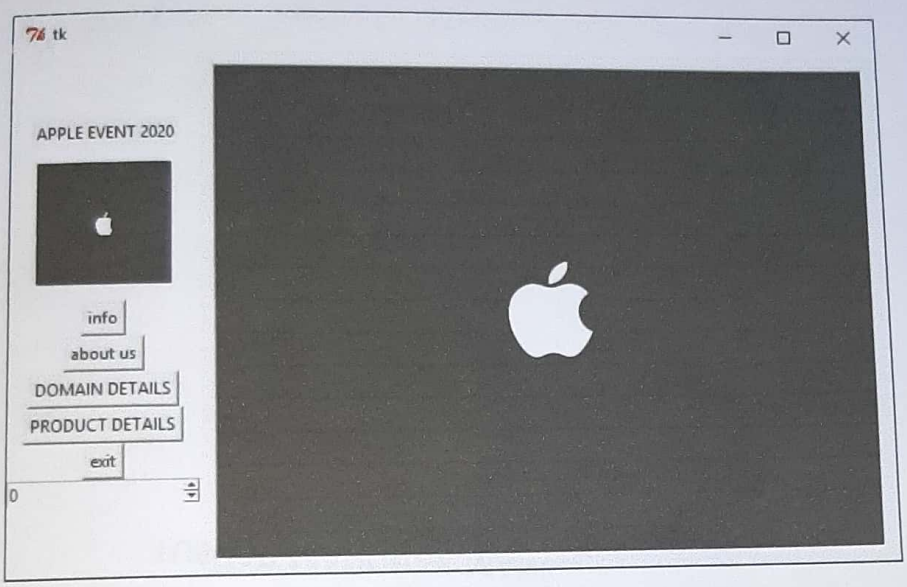
**SOURCE CODE:**

```
from Tkinter import*
root=Tk()
root.config(bg="white")

def f():
   quit()

def info():
   list1=Listbox()
   list1.insert(1,"co.name: APPLE")
   list1.insert(2,"language: SWIFT")
   list1.grid(ipadx="30")

def aboutus():
   list2=Label(text="STEVE JOBS theatere ")
   list2.grid(ipadx="30")

p1=PhotoImage(file="image.gif")
f1=Frame(root,height="250",width="500",bg="white")
f1.grid(row=0,column=0)

f2=Frame(root,height="250",width="500")
f2.grid(row=0,column=2)

p2=p1.subsample(5,4)
l1=Label(f1,image=p2,relief=FLAT)
l1.grid(row=1,column=1,padx=10,pady=10)

l2=Label(f2,image=p1,relief=SUNKEN)
l2.grid(padx=10,pady=10)

b1=Button(f1,text="info",relief=RAISED,command=info)
b1.grid(row=2,column=1)

b2=Button(f1,text="about us",relief=RAISED,command=aboutus)
b2.grid(row=3,column=1)

b3=Button(f1,text="exit",relief=RAISED,command=f)
b3.grid(row=6,column=1)

l=Label(f1,text="APPLE EVENT 2020")
l.grid(row=0,column=1)
```

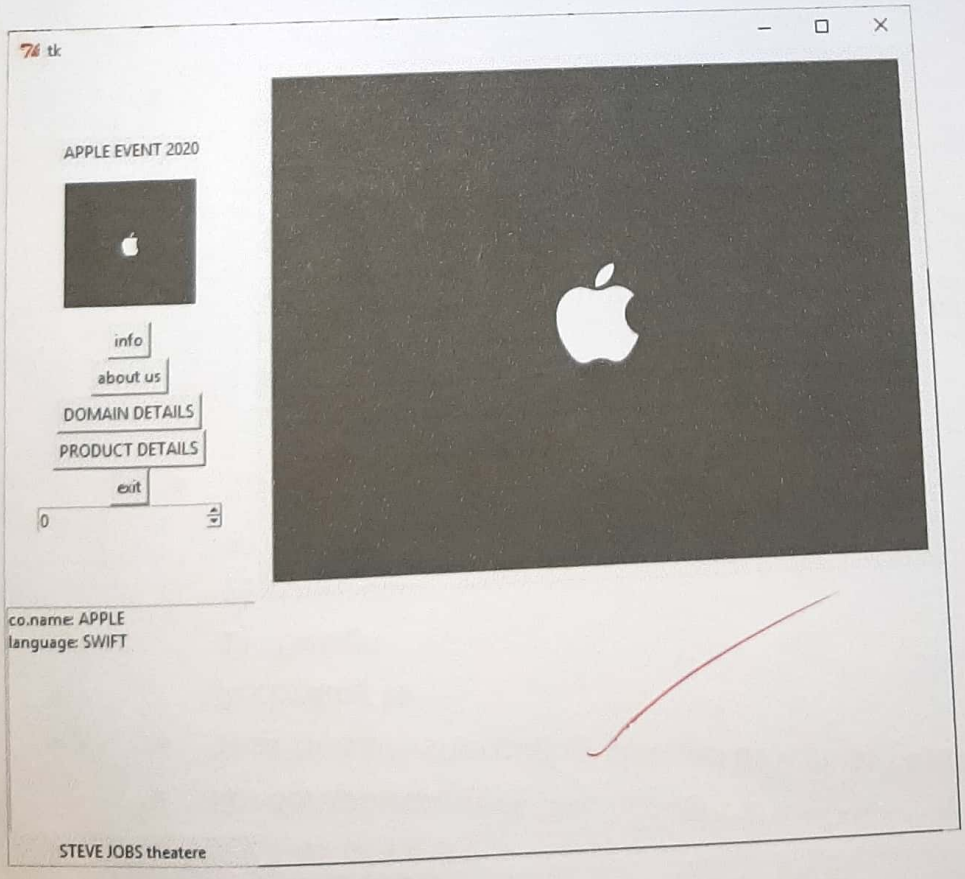
```
def main():
    top=Tk()
    top.config(bg="black")
    top.title("HOME")
    top.minsize(300,300)
    L=Label(top,text="APPLE STORE \nOUR DOMAINS: \neducation \nbusiness \nteachers \nstudents \n developer")
    L.pack()
    b1=Button(top,text="next",command=second)
    b1.pack(side=RIGHT)
    b2=Button(top,text="exit",command=terminate)
    b2.pack(side=LEFT)
    top.mainloop()

def second():
    top2=Tk()
    top2.config(bg="orange")
    top2.title("About US!")
    top2.minsize(300,300)
    def g():
        top2.withdraw()
    L=Label(top2,text="CREATED BY: Tanvi Mane \n For more details contact to our offical website")
    L.pack()
    b3=Button(top2,text="prev",command=g)
    b3.pack(side=LEFT)
    b2=Button(top2,text="exit",command=terminate)
    b2.pack(side=RIGHT)
    top2.mainloop()

def button():

    top3=Tk()
    top3.geometry("300x300")
    b1=Button(top3,text="Iphone",relief=FLAT)
    b1.pack()
    b2=Button(top3,text="Apple Watch ",relief=RAISED)
    b2.pack()
    b3=Button(top3,text="Ipad ",relief=GROOVE)
    b3.pack()
    b4=Button(top3,text="MAC",relief=SUNKEN)
    b4.pack()
    b5=Button(top3,text="Apple TV",relief=RIDGE)
    b5.pack()
    top3.mainloop()
```

```
def terminate():
    quit()

b5=Button(f1,text="DOMAIN DETAILS",command=main)
b5.grid(row=4,column=1)
b6=Button(f1,text="PRODUCT DETAILS",command=button)
b6.grid(row=5,column=1)

s=Spinbox(f1,from_=0, to=10)
s.grid(row=7,column=1)

root.mainloop()
```

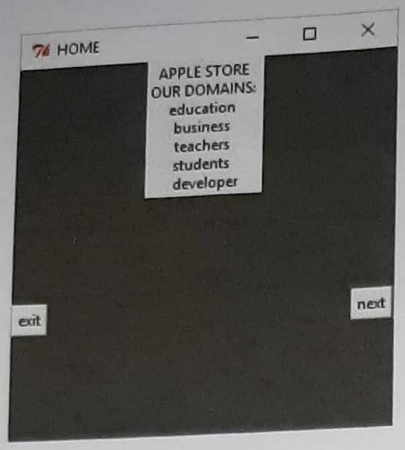
**OUTPUT:**



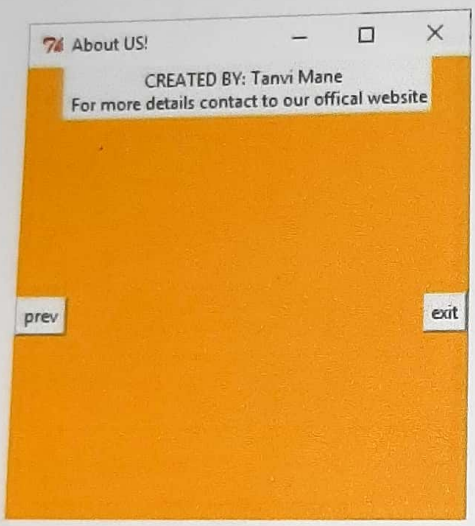

After clicking on info and about us: we get the following output

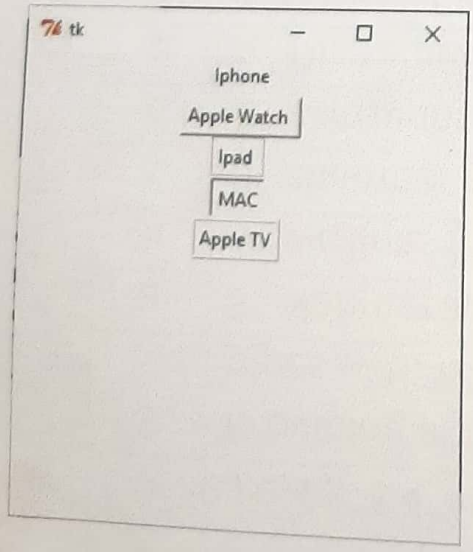


Clicking on domain button we will get the following output




Clicking on next button we will get the following output




Clicking on prev button we will get the previous window (domain details)
Clicking on the product details button we will get the following output