```
fileobj = open("Mytext.txt","w") # file open (write mode)
fileobj.write("Subjects in science stream" + "\n")
fileobj.write("Physics \n Chemistry \n Maths \n")# file write
fileobj.close()# file close

#readmode
fileobj = open("Mytext.txt","r")
Str1 = fileobj.read() # read()
print("The output of read method:", Str1)
fileobj.close()
>>>(The output of read method: 'Subjects in Science stream.
    physics
    chemistry
    Maths )
fileobj = open("Mytext.txt","r") #readline()
Str2 = fileobj.readline()
print("The output of readline method:", Str2)
fileobj.close()
>>> The output of readline method: Subjects in Science stream

#readlines()
fileobj = open("Mytext.txt", "r")
Str3 = fileobj.readlines()
print("The output of readlines method:", Str3)
fileobj.close()
>>The output of readlines method: ['Subjects in science stream
'physics \n', 'Chemistry \n', 'maths \n']
#file attributes
a = fileobj.name
print("Name of file :",a)
>>> Name of file: mytext.txt
b = fileobj.closed
print("(closed attribute", b)
>>> close attribute True
```


Practical 1

Aim: Demonstrate the use of different file accessing mode different attribute 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 variables.

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 file obj in 'w' mode that is the update mode and write contents.

Step5 : Open file obj in read mode display the update written contents and close open again in 'r+' mode with parameters 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.


```
c = fileobj.mode
print ("filemode", c)
>>> file mode c

d = fileobj.softspace
print ("softspace", d)
>>>(Softspace ,0)

# w+ mode
fileobj= open("Mytext.txt","w+")
fileobj .write ("Statistics")
fileobj.close()

# r+ mode
fileobj = open ("Mytext.txt","r+")
S1 = fileobj.read (6)
print ("output of r+:", S1)
fileobj.close()
>>> output of r+: Statistics

# append mode
fileobj = open ("Mytext.txt","a")
fileobj .write ("Biology")
fileobj.close()
fileobj = open ("Mytext.txt","r")
S 3 = fileobj.read()
print ("output of append mode:",S3)
fileobj.close()
>>> output of append mode: 'Statistics Biology'
```

```
#write mode
fileobj = open("mytext","w")
fileobj.write ("Psyphology")
fileobj.close()

#read mode
fileobj =open("Mytext.txt","r")
S = fileobj.read()
print ("output of read mode:",S
>>>output of read mode 'Statistics
```

```
#tell()
fileobj = open("Mytext.txt","r")
pos = fileobj.tell()
print("tell():",pos)
fileobj.close
>>> tell(): 0
#Seek()
fileobj =open("Mytext.txt",r")
St = fileobj.seek(0,0)
print("seek(0,0) is:",St)
fileobj.close()
>>> seek(0,0) is:0
fileobj= open("Mytext.txt","r")
St1 = fileobj.seek(0,1)
print("seek(0,1) is:",St1)
fileobj.close()
>>> Seek(0,1) is :0
file obj = open("Mytext.txt","r")
St2 = fileobj.seek(0,2)
print("seek(0,2) is:",St2)
fileobj.close()
>>>Seek(0,2) is : 57
# finding length of different lines exist within lines
fileobj = open("Mytext.txt","r")
Stat = fileobj.readlines()
print("output:",Stat)
for line in Stat:
        print(len(line))

fileobj.close()
>>> output:['Statistics', 'Biology']
```


Step 7 : open the fileobj in read mode, declare a variable and perform fileobject dot tell() and store the output consequently in variable.

Step8 : use the seek method with the arguments with opening the file obj in read mode and closing subquently.

Step9 : Open file obj with read mode also the the readlines method and store the output consequently in and print the same for counting the length use the for condition statement and display the length.

3/11/17

Practical no. 2.

Aim:- Iterators.

Step1: 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 iterating element in the tuple display the same.

Step2: The similar output can be obtained by using for conditional statement. An iterable variable is to be declared in for loop which will iterate.

Step3: 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.

Step4: call the declared function using function call.


```
# iter() and next()
mytuple1 = ("banana", "orange", "apple")
myiter1 = iter(mytuple)
print(next(myiter))
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]
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%5, listnum))
print(listnum)
def even(x):
    if (x%2==0):
        return "Even"
    else:
        return "odd"
print(list(map(even, listnum)))

#odd numbers
class odd:
    def __iter__(self):
        self.num = 1
        return self
    def __next__(self):
        num = self.num
        num = self.num + 2
        return num
    def __next__(self):
        if (self.num < 20):
            num = self.num
            self.num += 2
            return num
```

Output:

```
Even
Even
Even
Even
odd
odd
odd
Even
Even
odd
Even
```

Output:

```
1
3
5
7
9
11
13
15
17
19
```


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

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

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

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

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

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.


```
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
```

10/12

```
# IOError :-
try:
    fo = open ("abc.text","r")
    fo.write ("Hello world")
except IOError:
        print ("error")
else :
    print ("operation successful")
```

>>> error

```
#ZeroDivisionError:
a,b=1,0
try:
    print (a/b)
except ZeroDivisionError:
        print ("Error")
else:
    print("Successful")
```

>>> Error

```
# value Error:-
try:
x = int (input ("Enter a Statement:"))
except valueError:
    print ("Arithmetic error!")
else:
    print ("operation is successful")
```

>>> Enter a Statement : abc
Arithmetic Error!

17/12/19.

Practical no. 3.

Aim: Exception

Step 1: Use the try block to define normal course of action. Eg. Define the file object and open the file in read mode and write some content.

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

Step 3: Else display the operation is successful!

Step 4: In try block use valueerror and print the same message.

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

Step 6: Else display the operation is successful!

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 peform operation!

Step 10: Declare static variables and values -

Step 11: For multiline exception use the error types by separating them with a comma.

```
# Type Error
a = 0
b = input("Enter:")
try:
    print(a/b)
except TypeError:
    print("Incompatible value")
else:
    print("operation successful")
```

```
>>> Enter: t
    Incompatible value.
```

```
# Multiline exception
a,b = 1,0
try:
    print(1/0)
    print('10'+10)
except: (TypeError, ZeroDivisionError):
    print("Invalid Input!")
```

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

```
# 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 numbers:"))
    raise ValueError
except ValueError:
    print("Enter integer value")
```

>>> Enter : yz
Enter integer value

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 apporiate message.

Step 14: Define a function with empty list and calculate the lengths 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 enter integer values.

29/11

24/12/19

Practical no: 4.

Aim: Regular Expression

Step 1: Import re module declare pattern and declare Sequence use match method with declare arguements if arguements matched than print the same otherwise print pattern NOT FOUND!

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

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


```
# match()
  import re
  pattern = r"FYCS"
  Sequence = "FYCS represents computer Science"
  if re.match(pattern, Sequence):
        print("Match pattern found")
  else:
     print("Not found!")

>>> match pattern found

# numerical values
  import re
  pattern = r'\d+'
  String = 'hub123, howdy789, 45 howru'
  output = re.findall(pattern, String)
  print(output)
>>> ['123', '789', '45']

# Split()
  import re
  pattern = r'\d+'
  String = 'hello123, howdy789, 45 howru'
  output = re.Split(pattern, String)
  print(output)
>>> ['hello ', 'howdy', 'howru']
```

```
#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 0X02810F00)
   PYTHON

#verifying the given set of phone numbers
 import re
list1 = ['800456789', '9145673210', '7865913291',
        '903457649']

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!
```


Step 1: 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.

Step 5: import re module declare a sequence use search method for finding subsequently use the group() with dot operator.

Step 6: import re module declare list with numbers, use the conditional statement. If Criteria matches print cell number. matches otherwise print failed.

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

14/07

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 initial value as zero use for condition. And display the values respectively

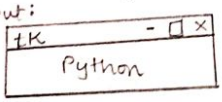

```
# vowels
import re
Str1 = 'plants are life'
output = re.findall(r'\b[aeiouAEIOU]\w+', Str1)
print(output)
>>['are']

# host and domain
import re
seq = 'abc.tcsc@edu.com, xyz@gmail.com'
pattern = r'[\w1-]+[\w1-]'
output = re.findall(pattern, seq)
print(output)
>>>['abc.tcsc', 'edu.com', 'xyz', 'gmail.com']

# country 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+,
  else:
     m = m+1
print('no. of male:",m)
print("no. of female:",f)
>>>['mr', 'ms', 'ms', 'mr']
(no of male:2)
(no of female:2)
```

```
#creation of parent window
from Tkinter import *
root =Tk()
l= label (root, text = "Python")
l.pack()
root.mainloop()
```

output:



#2:

```
from Tkinter import *
root=Tk()
l= label (root, text="Python")
l.pack()
l1 = Label(root,text = "CS!", bg="grey",
        fg="black", font="10")
l1.pack (side = LEFT, padx=20)
l2 = Label (root, text="cs", bg=lightblue",
        fg = "black", font ="20")
l2.pack (side = LEFT, pady="30")
l3= Label (root, text = "cs!", bg="yellow",
        fg="black", font="10")
l3.pack (side = TOP, ipadx="40")
```


7/1/2020.

Practical no:5.

Aim: GUI components. (Label, text)

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.

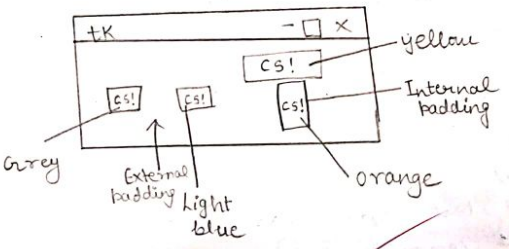
Step 3: use the pack() along with the object created from the text() and use the parameter
1) side = LEFT, padx = 20
2) side = LEFT, pady = 30
3) side = TOP, ipadx = 40
4) side = TOP, ipady = 50

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:



14/10

t. 080

```
#Radiobutton
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,"List1")
l2.insert(2,"List2")
l1.pack(anchor=N)
r1 = Radiobutton(root, text="option1", variable = var,
    value="option1", command = select)
r1.pack(anchor=N)
r2 = Radiobutton(root, text="option2", variable = var,
    value="option2", command=select)
r2.pack(anchor=N)
root.mainloop()
```

031

Practical no 5 (B)

Aim : GUI components.

1.

Step1 : Import the relevant method from the tkinter library. Create an object with the parent window.

Step2 : use the parent window object along with the geometry() declaring specific fixed 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 parent window and define the option with control available.

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

Step 6 : create an object from radio button which will take following arguments (parent window object, text variable which will take the values option 1, 2, 3, ..., variable, ~~keyboard~~ int, corresponding value and triggy the function declared.

Step 7: Now call the pack() for radio object so created and specify the arguments using anchor attribute.

Step 8: Finally make use of the mainloop() along with parent object.

2.

Step 1: Import relevant methods from the tkinter library.

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

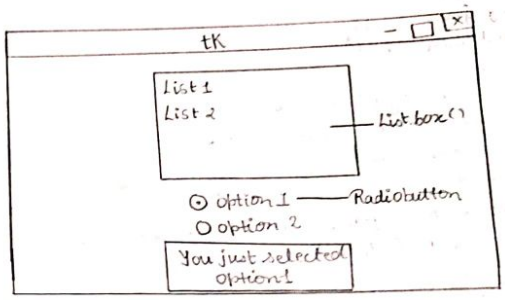
Step 3: use the geometry() for laying of the windows.

Step 4: create an object and use the Scrollbar()

Step 5: use the pack() along with the Scrollbar object with side and fill attributes

Step 6: use the mainloop with the parent object

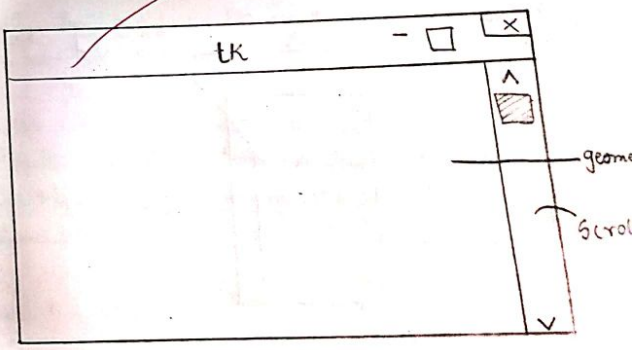
output:



2.

```
#Scrollbar.
from tkinter import*
root=Tk()
root.geometry("500x500")
S=Scrollbar()
S.pack(side="right", fill="y")
root.mainloop()
```

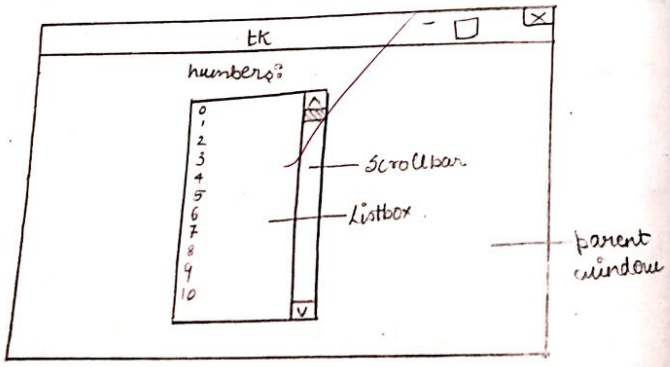
Output:



3. Listbox.

```
# using frame widget
from tkinter import*
window=TK()
window.geometry("680x500")
Label(window,text="numbers:").pack()
frame=Frame(window)
frame.pack()
listnodes=listbox(frame,width=20,height=20,
        font=("Times New Roman",10))
listnodes.pack(side="LEFT",fill='y')
scrollbar=Scrollbar(frame,orient="vertical")
scrollbar.config(command=listnodes.yview)
scrollbar.pack(side="right",fill="y")
for x in range(100):
    listnodes.insert(END,str(x))
window.mainloop()
```

output:



3.

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

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

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

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

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

Step 6: Use the listbox method along with the attributes like width, height, font. DO create a listbox() 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 term as the left frame and put it on the parent window on it LEFT side.

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

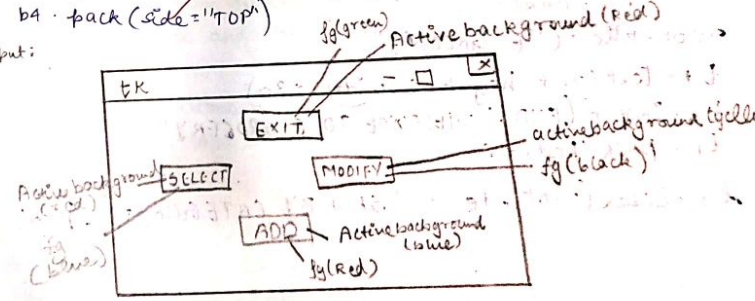
Step 6: Now use the pack() along with the side attribute.

Step 7: Similarly create a button using corresponding MODIFY operator put it into frame object on side = "RIGHT".

4. 11 Button.

```
from tkinter import *
window = Tk()
window.geometry("680 x 500")
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", pady=30)
b3.pack(side="bottom", pady=20)
b4.pack(side="TOP")
```

Output:



```
# E-Grocery.
from tkinter import *
root = TK()
def main ():
    roo = TK()
    roo.config()
    roo.title("LIST YOUR PRODUCTS!")
    l = Label (roo, text = "ENTER LIST OF PRODUCTS HERE")
    l.pack()
    e1 = Entry (roo).pack()
    b1 = Button (roo, text = "CONTINUE", command = top)
    b1.pack()
    b2 = Button (roo, text = "CANCEL", command = quit)
    b2.pack()
    roo.mainloop()
def top():
    top = TK()
    top.config()
    top.title("BUY")
    l1 = Label (top, text = "CONFIRM YOUR ORDER?!")
    l1.pack()
    b3 = Button (top, text = "CONFIRM", command = quit)
    b3.pack()
root.title = ("E-GROCERY")
t1 = Text (root, height = 2, width = 30)
t1.insert (END, "WELCOME TO E-GROCERY")
t1.pack (padx = 10)
l3 = Label (root, text = "SHOP BY CATEGORY:").pack()
```


Practical no 5 (c)

Aim: GUI components.

Step 1: Import tkinter which is the Standard python interface from python library.

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

Step 3: use the title() to give an appropriate title to parent window.

Step 4: use text() and insert() to insert or display the text that you want to show.

Step 5: use the label() to label the content below.

Step 6: use Checkbutton() and write the text you want to display which will return the main function.

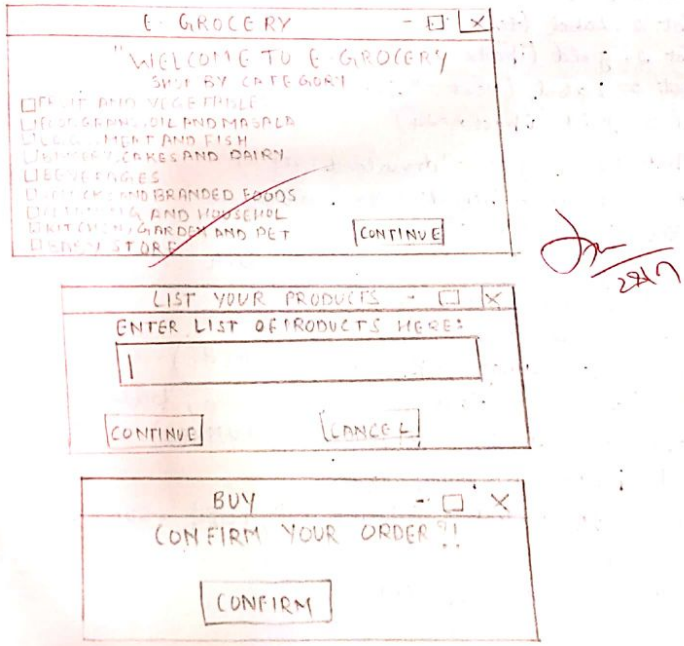
Step 7: In the main function create another window with title and label.

Step 8: create buttons using the Button() which calls the top().

Step 9: In the top function display other text using the label().

Step10: use Button() to display the corresponding client.

```
C1 = Checkbutton(root, text="FRUIT AND VEGETABLES", command=main).pack()
C2 = Checkbutton(root, text="FOODGRAINS, OIL AND MASALA", command=main).pack()
C3 = Checkbutton(root, text="EGGS, MEAT AND FISH", command=main).pack()
C4 = Checkbutton(root, text="BAKERY, CAKES AND DAIRY", command=main).pack()
C5 = Checkbutton(root, text="BEVERAGES", command=main).pack()
C6 = Checkbutton(root, text="SNANCKS AND BRANDED FOODS", command=main).pack()
C7 = Checkbutton(root, text="Cleaning AND HOUSEHOLD", command=main).pack()
C8 = Checkbutton(root, text="KITCHEN, GARDEN AND PET", command=main).pack()
C9 = Checkbutton(root, text="BABY STORE", command=main).pack()
b5 = Button(root, text="CONTINUE", command=main)
b5.pack()
root.mainloop()
```



```
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 (ipadx=20)
def aboutus():
    list 2 = Label (text="About us")
    list 2. grid (ipadx=30)
    list 3 = Label (text="Steve jobs theater march 2020")
    list 3. grid (ipadx=24)
P1 = Photo Image (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)
```


Practical no. 5 (D)

Aim: GUI Components

Step 1: Import relevant methods from the tkinter library.

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

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

Step 4: Define a function about use with label widget and text attribute and subsequently use the grid().

Step 5: use photoimage widget with file and filename with gif attribute.

Step 6: Create a frame object along with the Frame() along with parent window object height and width specified and subsequently use the grid() with row and column attribute specified.

Step 7: Create another object and use the Subsample (5, 4).

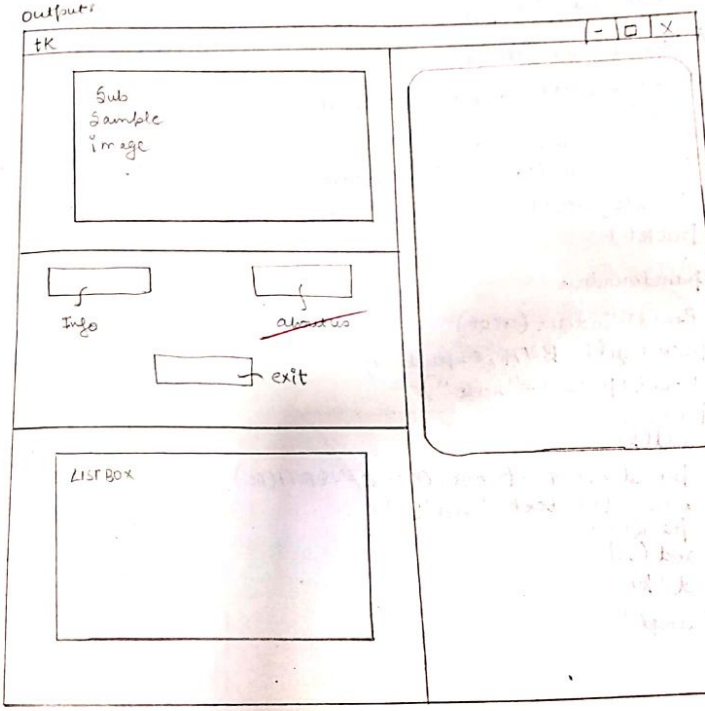
Step 8: Similarly, create another frame object as declared by step 7.

Step 9: use label widget along with the frame objects relief attribute and subsequently use the grid().

Step 10: Now create button object dealing with different section of frame.


```
b2 = Button (f1, text = "About us", relief = SUNKEN, command = about us)
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 ()
```

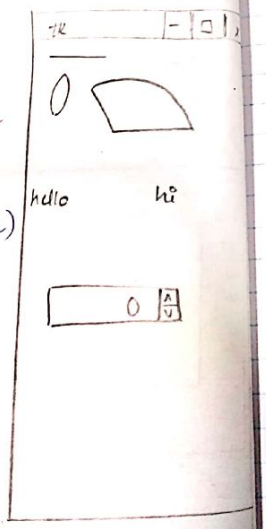
output:



```
# Spinbox
from tKinter import *
root = TK()
S1 = Spinbox(root, from_=0, to=10)
S1.pack(side=TOP, padx=200, pady=200)

# Canvas (line, arc, oval)
C = Canvas(root, height=250, width=300)
co = 10, 20, 30, 20
C.create_line(co, fill="red")
C.create_arc(10, 50, 240, 210, start=0, extent=150, fill="Yellow")
C.create_oval(20, 80, 15, 40, fill="blue")
C.pack()

# panedwindow.
p = PanedWindow(root)
p.pack(fill=BOTH, expand=10)
l = Label(p, text="hello")
l.pack()
p.add(l)
p1 = panedwindow(root, orient=VERTICAL)
l1 = Label(p1, text="hii!!")
l1.pack()
p1.add(l1)
p.add(p1)
mainloop()
```



Practical no. 5. (E)

Aim: GUI components.

Step 1: Import thinter from the python library.

Step 2: use the spinbox () to display numbers from 1 to 10.

Step 3: use canvas method to use various shapes.

Step 4: use line, arc and create ().

Step 5: use paned window to write two texts.

Practical no. 6 :

Aim: Database connectivity.

1. Import the corresponding libraries for making the database connection import OS and sqlite 3.
2. Now create the connection object using sqlite 3 library and the connect() for creating the new database.
3. Now create the cursor object using the cursor() from the connection object created in the earlier step.
4. Now use the execute() for creating the table with the column name and the respective datatype.
5. Now with the cursor obj use the insert statement entering the values corresponding to the different fields considering the datatype.
6. Use the commit() to complete the transaction using the connection object.
7. Use the execute statement for inserting the values using select.
8. Use the fetch() to display values using the cursor().
9. Using print statement display the values in a table.
10. Use the close() to terminate the database connection.

Code :-

```
import OS, Sqlite 3
connection = Sqlite 3.connect("Bank.db")
cursor1 = connection.cursor()
cursor1.execute ('create table account(username char,
                 uid int, accno int)')

cursor1.~~insert~~ execute ('insert into account values ("Raj", 125,001),
                 ("Rani", 124,002), ("Rajni", 125,003)')

connection.commit()
cursor1.execute ('Select username from account')
var= cursor1.fetchall()
print(var)
connection.close().
```

Output :-

[('Raj'), ('Rani'), ('Rajni')]