PRACTICAL - 1

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

Step 1: Create a file object using open method & use the write access mode followed by writing some contents onto the file & then closing the file.

Step 2: Now open the read mode & then use read(), readline() & readlines() & store the output in variable & 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 & finally the output of the Softspace attribute.

Step 4: Now open the fileobject in write mode. Some another content close subsequently. Then again open the fileobject in 'w+' mode that is the update mode & write contents.

```
fileobj = open("abc.txt", "w")
fileobj.write('python is indented language')
fileobj.close()
fileobj = open("abc.txt", "r")
a = fileobj.readline()
print(a)
```

Output:

>>> python is indented language.

```
fileobj.close()
c = fileobj.name
b = fileobj.closed
c = fileobj.mode
d = fileobj.softspace
print(e)
print(b)
print(c)
print(d)
```

output:

```
>>> abc.txt
>>> True
>>> r
>>> 0
```

```
# read()
str1 = fileobj.read()
print("The output of read method:", str1)
fileobj.close()
```

>>> ("The outputs of read method : ','computer science

Step 5: Open file object in read mode disp. the update written contents cl close open again in 'r+' mode with parameter passed cl display the output Subsequently.

Step 6: Now open filobject is append mode open write method write contents. close the filobject again open the filobj in read mode cl display the 'appending' output.

Step 7: Open the filobj in read mode. declare the variable cl perform filobj dot tell method cl store the output consequ in variable.

Step 8: Use the seek method with the argument with opening the file obj in read mode cl closing Subsequently.

Step 9: open filobj with read mode also use the readlines method cl store the output consequently in cl print the same for counting the length use for condition Statement cl display the length.

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

output:

```
>>>('The output of readline method: ', 'computer science')
```

```
# readlines()
str3 = fileobj.readlines().
print(*str3).
fileobj.close().
>>> ['computer science \n', 'DBMS \n', 'Python \n', 'OS \n']
```

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

```
b = fileobj.closed.
print("(close) attribute: ", b).
>>> (close) attributes: ', 'True').
```

```
c = fileobj.mode.
print("file mode ", c).
>>>('file mode is 'r').
```

```
d = fileobj.softspace
print("softspace", d).
>>>(softspace: ', 0).
```

```
# ~~write~~ r+ mode.
fileobj = open("abc.txt","r+
s1 = fileobj.read(6).
print(s1).
fileobj.close().
>>> ('computer science')
```

```
# append
fileobj = open("abc.txt","a")
fileobj.write(" data structure")
fileobj.close()
fileobj = open("abc.txt","r")
s3 = fileobj.read()
print("s3")
fileobj.close
>>> computer scienue data structure

# tell()
fileobj = open("abc.txt","r")
pos = fileobj.tell()
print("tell(): ", pos)
fileobj.close
>>> ("tell(): ', 01)

# seek.
fileobj.seek(0,0)
print fileobj.readline()
>>> None.

fileobj.seek(0,1)
print fileobj.readline()
>>> None (ython is indented language).

fileobj.seek(0,2)
print fileobj.readline()
>>> (thon is indented language).

# length of different lines exists.
fileobj = open("abc.txt","r")
stat = fileobj.readlines()
print(" stat)
for line in stat:
    print(len(line))
fileobj.close()
>>> (["python is indented language"] 24
```

31/1/19

PRACTICAL - 02.

AIM: Iterators.

021

Step 1: Create a tuple with elements that we need to iterate using the iter() next method. The number of time we use the iter() next Iterating element in the tuple Display the same.

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

Step 3: Define a function name Square with a parameter which will obtain output of Square value of the given number. In Similar fashion declared circle to get the value. raised 3 () 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 'lambda' i-e annonymous function () print the Same.

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

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

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

Step 9: Now use the next() & 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.

iter() & next()

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

```
>>> banana
>>> Apple
>>> orange
```

for loop.

```
mytuple1 = ("Gucci", "ZARA", "Puma")
for x in range mytuple1:
    print(x)
```

```
>>> Gucci
    ZARA
    Puma
```

Square & cube.

```
def square(x):
    Y = x * x
    return Y
def cube(x):
    Z = x * x * x
    return Z
function1 = [square, cube]
for r in range(5):
    value = list(map(lambda x: x(r), function1
    print(value)
```

[1, 1]
[2, 4]
[3, 8]
[4, 27]
[5, 64]

```
# map().
listnum = [0, 4, 5, 7, 9, 11, 15, 19, 20, 21, 25].
listnum = list(map( lambda x : x%.5, listnum)).
print(listnum)
def even(x):
    if (x%.2 == 0):
        return even
    else:
        return odd
list(map(even, listnum)).
```

>>> [0, 4, 0, 2, 4, 2, 0, 4, 0, 1, 0].

```
# 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 = odd().
= iter(myobj)
input("enter number: ")
```

step 12: Define a function with if condition, if exactly equal to one return else on parameters.

step 13: Enter a input from the declare an empty list else append method for appending the input value print the output variable.

```
for i in mylist:
    if (i<x):
      print(i)
```

024

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

```
def fact(n):
    if (n==1):
        return 1.
    else:
        return (n* fact(n-1))
= int (input("enter Number:  "))
ist = [ ]
list.append(n).
= map(fact, list)
print("the factorial is : ", Y)
```

```
>>> The f
    Enter Number: 5
('The factorial is : ', [120]).
```

I/O Error:

```
try:
fo = open ("Rizwana", "r").
fo - write ("my name is Rizwana").
except:
    print ("Error").
else:
    print ("operation succesfull").
>>> Error.
```

* value Error:

```
try:
  x = int(input("Enter statement")).
except value error:
  print ("error").
try:
  a = open ("abc.txt", "w").
  a.write ("Mars").
except I.O Error:
  print ("I live on Mars").
else:
  print ("operation succesfull").
```

```
>>> Enter: 11.
    Succesful
>>> Enter : abc.
    I live on Mars.
```

Type Error, Zero Division Error.

```
try: Multiline Exception.
  print (~~incompatible~~ 110).
  print ('10' + 23).
except (Type Error, Zero Division Error).
    print ("invalid").
```

```
>>> 1.0
>>> 1010
>>> Invalid.
```

PRACTICAL - 3.

AIM : Exceptions.

step 1 : In any try block open an fill in the open mode with write mode, write some content in the file.

step 2 : In except (IO Error) block use the error in IO error. use the appropriati message display.

Step 3 : Else display the opration. is succesful.

step 4 : In try block except an input from the user.

Step 5 : In except block use value Error & print the Same menage.

Step 6 : Else display the operation is succesful.

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. Type Error print Incompatible values.

Step 9: use except with type Error of zero divisor Error & print the message according to that if entered number is zero, then it is not able to perform operation.

Step 10: Declare static variables & values.

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

Step 12: use try block. open a title. in write mode & subsequently enter values in the file.

Step 13: use the IO Error & display appropriate message.

Step 14: Define a function with Empty list & Calculate the length of the list.

Step 15: Define another function Y. Initialize. & declare some elements in dist & Calculate the length of the same & display the same.

```
# Try Zero Division Error
a = 1.
b = input ("Enter ")
try:
    print (a/b)
except TypeError:
    print ("incompatible value")
except zero Division Error:
    print ("Denominator is Zero")
```

output:

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

```
# using except keyword:
try:
a = open("abc.txt", "w").
a.write ("python").
except IO Error:
    print ("Error!").
else:
    print ("succesfull)
def x():
    d = [].
    print (len (c)).
def Y ():
    l1 = [2, 4, 4, 1].
    print (len (l1)).
print (x())
print (Y()).
```

output:

```
>>> Succesfull.
    0
    None.
    4
    None.
```

Raise keyword:

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

output:

```
>>> Enter: xyz
Enter integer value!
```

Resplit:

```
import re
Sequence = "hello123, howay456, 789, howyll"
pattern = r'\d+'
output = re.split(pattern, Sequence)
print(output)
```

```
>>> ['hello', '123', 'howay', '456', '789', 'howyll']
```

Find all:

```
import re
Sequence = ('hello123, howay456, 789 Hii')
pattern = r'\d+'
output = re.findall(pattern, Sequence)
print(output)
```

```
>>> ['hello', 'howdy', 'Hii']
```

Removing wide spaces:

```
import re
S = 'ab  c  d  ef'
pattern = r'\s+'
r = ' '
output = re.sub(pattern, r, S)
print(output)
```

```
>>> abcdef
```

PRACTICAL-4.

AIM: Regular expression:

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 & 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 separating the charachters. use the Split () method with pattern & String as its arguments & then display the output.

Step 5:
Import re module from the library. A sequence of your choice can be initialized a pattern beginning with blank space.
use Sub() method with three arguments pattern Sequence. & the variable initialized

with blank space 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 charachter use find all() method with pattern & Sequence as arguments & display the result.

Step 8:
Import remodule Initialize a Sequence use find all() method with raw String & list of vowels. i.e pattern along with the Sequence print the output:

Step 9:
Import re module Initialize a Sequence with desired String use search() method & the arguments should be the word to search along with the Sequence.

Step 10:
Import the result now use the group method & then display the output.

Alpha numeric charachter

```
import re
seq = 'xyz tcsc@gmail.com, abc tcsc@gmail.com'
pattern = '[\w\.-]+[\w\.-]+'
output = re.findall(pattern, seq)
print(output)
>>>['xyz tcsc', 'gmail.com', 'abc.tcsc', 'gmail.com']
```

Research method.

```
import.re.
s='python is an indented language'
output = re.Search('\Apython', s)
print(output).
v = output.group().
print(v).
>>> : Src. SRE - match object at 0x 030F9413r3python.
```

Number start from 8 or 9.

```
import.re.
i=['9930400224','9819386123','5930813355'].
for v in i:
    if re.match[r'[8-9]{1}[0-9]{9}v or len(v)==10
        print("call no matches").
    else:
        print("call no does not match"].
>>> call number matches.
call number matches.
call values does not match.
```

```
# String starts with vowel:
import re
s = 'cats are very lazy'
p = re.findall(r'\b[aeiou]\w+', s)
print(p)
```

```
>>> ['ats', 'are', 'ery', 'azy']
```

```
#import re
s = 'mr.a, ms.c, mr.L, ms.D, ms.U)
p = r'[/ms/mr/]+1
o = re.findall(p, s)
print(o)
m = 0
f = 0
mr v in o:
    if v == 'ms'
        m = m+1
    else:
        f = m+1
print("No of malis", m)
print("No of females", f)
```

```
>>> ['mr', 'ms', 'ms', 'mr', 'ms'].
No of malis = 2
No of females = 3
```

Step 11:
Import the re module from the library Initialize a variable with some call number use for loop of the . if loop with match() method of the arguments should be the range also check the range.

Step 12:
if it satisfies the condition print "call number" matches else print " cell no does not match".

step 13:
Import re module Initialize a string with list of males & females, from a pattern beginning with raw string use find all() method with two arguments i.e pattern & sequence.

Step 14:
Print the output Initialize a variable for counting males & females. use for loop to check condition if it is 'ms'. than count of female will be incremented else count of male will be incremen

Step 15:
Display the result with exact count of males & females.

PRACTICAL - 5(A)

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 & use the text method.

Step 4 : use the mainloop() for trigerring of the corresponding above mention events.

#2 :

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

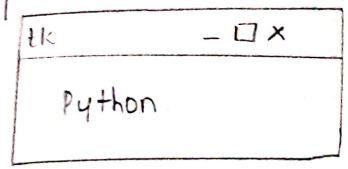
Step 2 : Create a variable from the Text method & position it on the parent window.

Step 3 : use the pack() along with the object created from the Text() & use the parameter.

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 -"20").
l1.pack (side = LEFT, padx = 20).
l2 = label (root, text ="cs", bg ="light blue", fg ="black", font="30")
l3 = label (root, text ="cs", bg ="yellow", fg ="black", font="10").
l2.pack (side = LEFT, pady="30").
l3.pack (side = TOP, ipadx = 40).
l4 = label (root, text = "cs", bg = "orange", fg = "black",
           font= "10").
l4.pack(side = TOP, ipady = 50)
root.mainloop().
```

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) & then use the pack() with a relevant padding attributes.

14/07.

PRACTICAL 5-B

AIM: GUI components.

#1:

Step 1: Import the relevant methods from the tkinter library create an object with the parent window.

Step 2: Use the parent window object along with the geometry() declaring specific pixel size of the parent window.

Step 3: Now define a function which tells user about the given selection mode from multiple option available.

Step 4: Now define the parentwindow & define the option with control variable.

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

Step 6: Create an obj from radio button which will take following arguments parentwin object, text variable which will ~~be~~ take the values opt no 1, 2, 3, ... variable arguments, corresponding value & triger the function declared.

o Button.

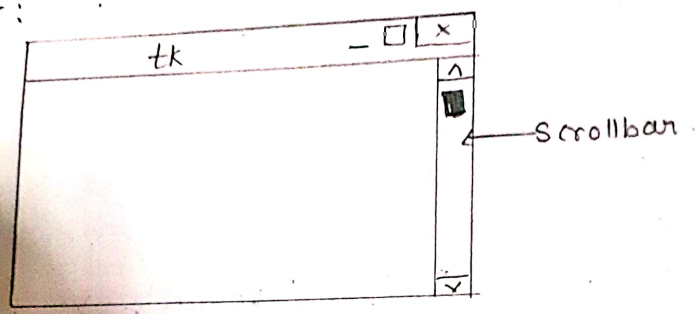
```
1 tkinter import *
t = Tk().
t.geometry ("500 x 500").
 select():
Selection = " You just selected" + str(var.get())
t1 = label(text = Selection , bg = "white", fg = "green").
t1.pack(side = TOP).
= StringVar()
: listbox()
insert (1, "list1")
insert (2, "list2")
pack (anchor = N).
= RadioButton(root, text = "option 1", variable = Var, value =
    " option 1", command = select)
pack (anchor = N)
= RadioButton (root, text = "option 2", variable = var, value =
    "option2", command = Select)
pack (anchor = N).
ot.mainloop()
```

ut:

Scrollbar()

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

output:



3:

```
# using frame widget.
from tkinter import *
window = Tk().
window.geometry("680 x 500")
Label(window, text = "number: ").pack().
frame = Frame(window).
frame.pack().
ListNodes = Listbox(frame, width = 20, height = 20, font = ("Ti
            New Roman", 10)).
ListNodes.pack(side = "left", fill = "y").
Scollbar = Scrollbar(frame, orient = "vertical").
Scrollbar.config(command = ListNodes.yview).
Scrollbar.pack(side = "right", fill = "y")
```

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

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

Step 1: Import relevant methods from the tkinter library.

Step 2: Create a parent object corresponding to th powerthewindow.

Step 3: use the geometry() for laying of the wind

Step 4: Create an object & use the Scrollbar.

Step 5: use the pack() along with Scrollbar ob with side & fill attribute.

Step 6: use the mainloop() with the parent object.

#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 by any other suitable pixel value.

Step 4: Use the label widget along with the parent object created & subsequently use the pack method.

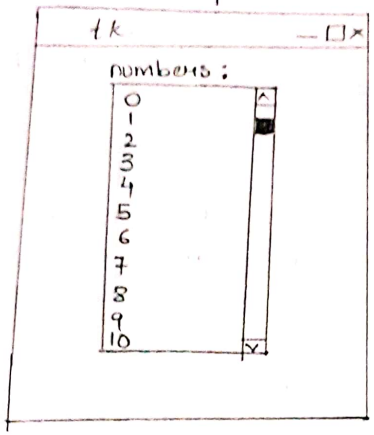
Step 5: use the frame widget along with the parent object created & use the pack method.

Step 6: use the listbox method along with the attributes like width, height font to 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() & use pack().

Step 8: Trigger the mainloop().

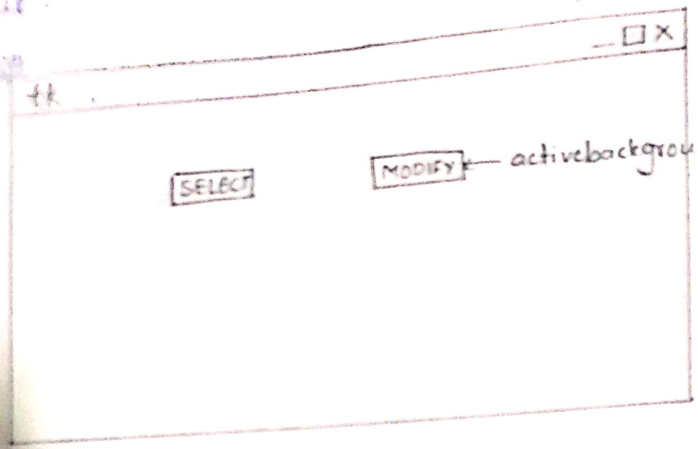
```
for x in range(100):
    listNodeo.insert(END, str(x))
window.mainloop().
```



#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").
b1.pack(side = "LEFT", padx = 20)
b2.pack(side = "right", padx = 30).
root.mainloop().
```

output :



#4:

Step 1: Import relevant methods from tkinter library.

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

Step 3: Now define the frame object from the method & place it on the parent window.

Step 4: Create another frame object termed as the left frame & put it on the parent window on its LEFT side.

Steps: Similarly define the RIGHT frame & subsequntly define the button object placed onto the given frame with the attributi as text, active background & foreground.

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

Step 7: Similarly Create the button object corresponding to the MODIFY operation put it onto the frame object on side = "right".

PRACTICAL - 5(c).

AIM: GUI components.

Step 1: Import the relevant method from tkinter library.

Step 2: Import tkinter messagebox.

Step 3: Define a parent window object alo with the parent window

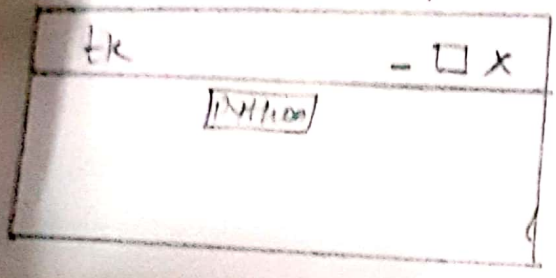
Step 4: Define a function which will use tkmessagebox with showinfo method along with info method window attribute.

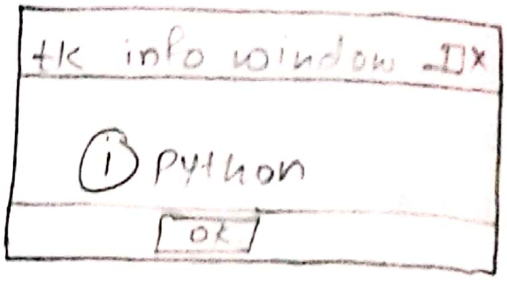
Step 5: Declare a button with parent wi ~~object~~ along with info wind attribute.

Step 6: place the button widget onto the parent window & final call mainloop() for triggerin 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().
```





Multiple window
Different button(Relief()).

```
from Tkinter import *.
root = Tk().
root.minsize(300, 300).
def main():
    root = Tk().
    root.config(bg = 'black')
    root.title("HOME")
root.minsize(300, 300).
l = label(root, text = 'SAN FRANCISCO \n places of
    Interest \n Golden gate Brigde \n lombard
    Street \n chinatown").
l.pack().
B1 = Button(root, text = "next', command = Second).
B.1.pack().
```

#2:

Step 1: Import the relevant tkinter library.

Step 2: use parentwindow object. when minsize function for window size.

Step 3: def a function declare main parent window obj. & use config(), title(), minsize(), label as well as button() & use pack() & mainloop() simultaneously

Step 4: Similarly def the function & used the attributes.

Step 5: Declare another function button along with the parent object & declare button with with attributes like FLAT, RIDGE GROOVE, RAISED, SUNKEN along with the relief widget.

Step 6: Finally called the mainloop() for event driven programming.

```
2 = Button(root , text = 'cancel', command = self.
        terminate)
2.pack():
ot mainloop().

f second():
  top = Tk()
  top.config(bg = "black").
  top.minsize(300,300).
  l = Label(top, text = "created by: RIZWANA").
  l.pack()
  B3 = Button(top, text = "BACK", command = main)
  B3.pack().
  B2 = Button(top, text = "exit", command = termin
  B2.pack().
  top.mainloop().

f button1():
top2 = Tk().
top2.geometry('300x300').
B1 = Button(top2, text = "FLAT", relief = FLAT).
B1.pack()
2 = Button(top2, text = "Groove", Relief = GROOVE)
2.pack().
3 = Button(top2, text = "Raised", Relief = RAISED).
3.pack(),
4 = Button(top2, text = "sunken", relief = SUNKEN)
4.pack.
```

```
B5 = Button (top2, text = "Bridge", Relief = RIDGE
B5.pack()
Relief
top2.mainloop().
def terminate():
quit()
B5 = Button(root, text = "Travel", command = ma
B5.pack().
B6 = Button(root, text = "Details", command = bo
B6.pack()
root.mainloop().
```

PRACTICAL - 5(D).

AIM: GUI components

Step 1: Import relevant method from tkinter library.

Step 2: Create parent window object & use the config method along with background.

Step 3: define a function finish with the messagebox widget which will use the listbox object along with insert method & insert the same & finally use the grid() with ipadx attribute

Step 4: Define a function about ~~is~~ label widget & text attribute & subsequently use the grid().

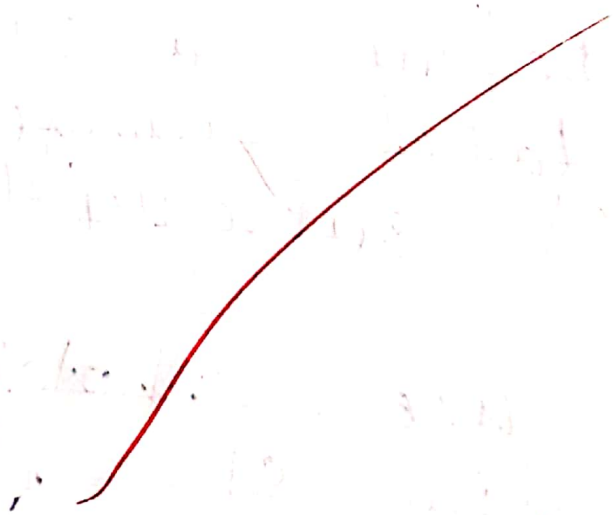
Step 5: use photoImage widget with file & filename.

Step 6: Create a frame object along with the frame() along with parent window. object with height & weigl width specified.

Step 7: Similarly create another frame object.

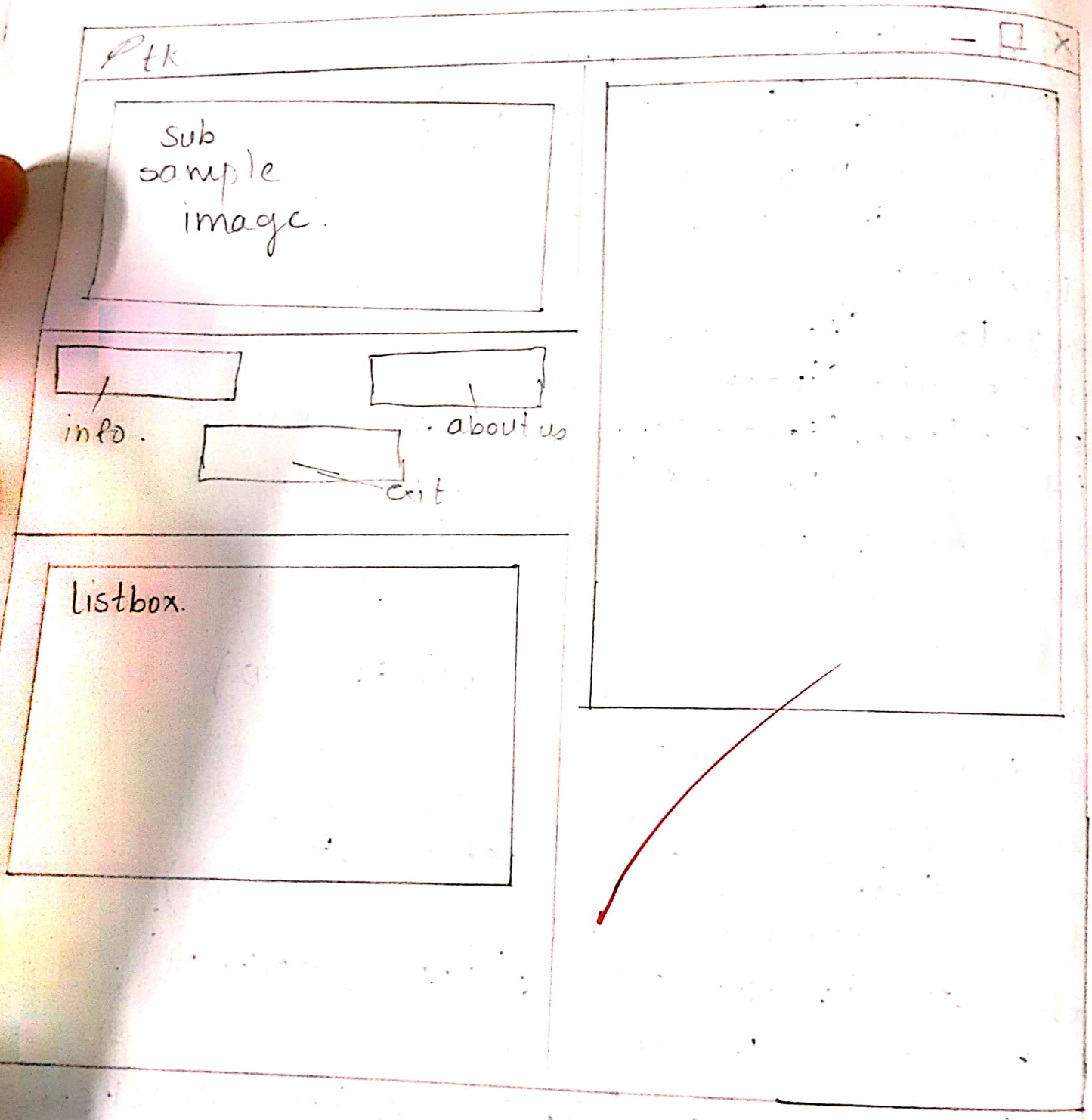
Step 8: use label widget along with frame object, relief attribute. & use grid().

Step 9: Now create button obj dealing with different section of frame.



```
from tkinter import *
root = Tk().
root.config(bg = "pink").
def finish():
    messagebox.askokcancel("warning", this will end
                                    program"))
    quit().
def info():
    list1 = listbox().
    list1.insert(1," co.name: Apple")
    list1.insert(2," Products: iphone").
    list1.insert(3," language: swift").
    list1.insert(4,"os: IOS").
    list1.grid(ipadx = 30).
def about us():
    list2 = label(text = "about us")
    list2.grid(ipadx = 30)
    list3 = label(text = "steve jobs 2020")
    list3.grid(ipadx = 24).
P1 = photoImage(file = "download.gif").
f1 = frame(root, height = 35, width = 5).
f1.grid(row = 1, column = 0)
f2 = Frame(root, height = 35, width = 500)
f2.grid(row = 1, column = 1).
P2 = P1.subsample(5,4)
L1 = label(f1, image = P2, relief = FLAT)
1.grid(row = 1, column = 0, padx = 20, pady = 15).
2 = label(f2, image = P1, relief = SUNKEN).
.grid(padx = 25, pady = 10).
1 = button(f1, text = "Info", relief = SUNKEN,
                command = info).
.grid(row = 1, colum = 0).
```

```
b2 = Button (f1, text = "About us", relief = SUNKEN,
            command = about us)
b2.grid (row = 1, column = 2, padx = 2)
B3 = Button (f1, text = 'EXIT', relief = RAISED,
            command = finish).
B3.grid (row = 2, column = 1, padx = 15).
root.mainloop().
```



PRACTICAL - 5 (E).

AIM: GUI component

1). Creating a Spinbox.

step 1: Create an object from the tkKlibrary & subsequently create an object from Spinbox.

step 2: call the parent window along with from - to - attribute.

step 3: Make the object so created onto the parent window & trigger the corresponding events.

Panned Window.

step 1: Create an object from the paned window method & use the pack method with the attribute fill & expand.

Step 2: Create an object from the label method & put it on to the paned window with the text attribute & use the add method to embed the new object.

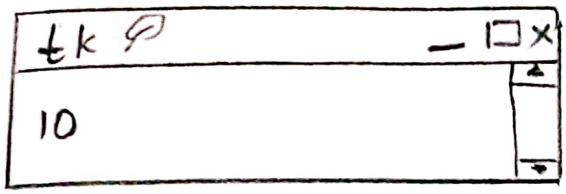
Step 3: Similarly, create a second paned window object & add it onto the first paned window with orientation specified.

Step 4: Now create another label object & place it onto the 2nd panel window object & add it onto the 2nd paned.

Step 5: trigger the mainloop method & terminate.

1) Spinbox

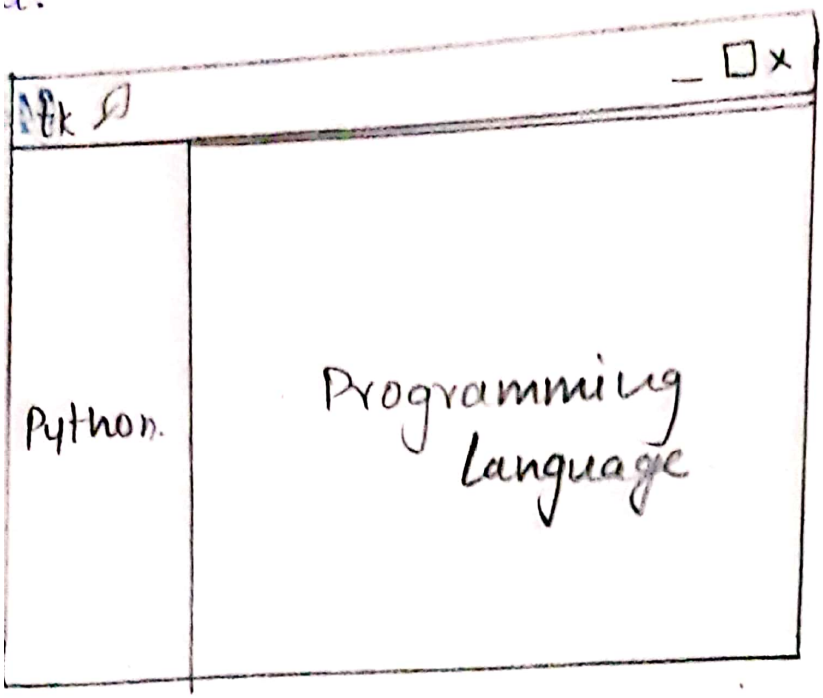
```
from tkinter import*
root = Tk()
S = Spinbox(root, from = 0, to = 10)
S.pack()
root.mainloop()
```



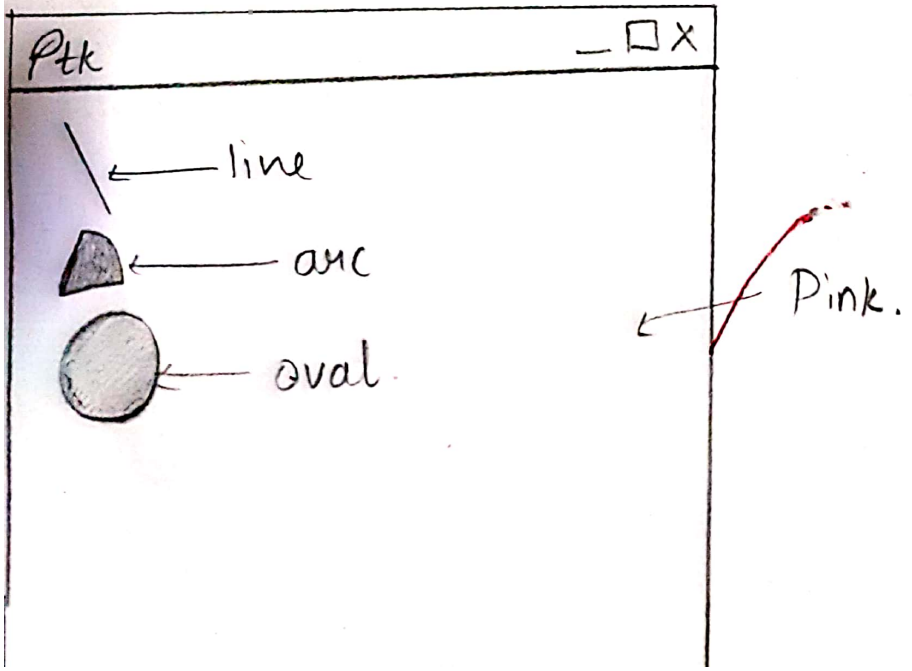
2) PannedWindow.

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

t:




```
tkinter import *
Tk()
vas (root, height = 100, width= 200, bg = "light Pink")
C. Create_arc (10,20,30,40, start=20, extent = 10, fill=
 C..Create-oval(10, 30,90,50, fill = "black").
= C. Create_line (10,15, 20, 30, fill = "maroon")
-().
ainloop().
```



3) Canvas.

Step 1: Create an Import the relevant tkinter library & declare the parent window.

Step 2: Create an object from canvas() & use the attributes height width, background & parent window object.

Step 3: use the method create line, create oval & create arc along with the canvas object so created & use the coordinate values.

Step 4: Similarly, use the other method & call the pack method & trigger the mainloop method.

PRACTICAL - 6

AIM: Database Connectivity.

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

step 2: use the object so created for out accessing the given website & corresponding regular name.

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

Step 4: use the close() to terminate database library.

output:

```
>>> 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.
```

output:

database not empty // match.

output:

```
import os, sqlite3
conn = sqlite3.connect("Hospital.db")
cur = conn.cursor()
cur.execute('create table dos(Name cha
    Roll no int')
cur.executi("insert into dos value('Rizwana
    1718),('TAT',1719))
conn.commit()
cur.executi('select * from dos')
print(cur.fetchall())
conn.close()
```

output:

[('Rizwana', 1718), ('TAT', 1719)].

#2:

step 1: Import corresponding library to make database connection.

step 2: Now create the connection object using sqlite-3. library & then connect() for creating new database()

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

step 4: Now with cursor object use the insert Statement for entering the values corresponding the diffurent fields.

step 5: use the commit() to complete the transaction using the connection object.

step 6: use the execute Statement along with cursor-object for accessing the value from the database using the select from where clause.

Step 7: Finally use the fetch() or fetchall() for displaying the values from the table using the cursor-object.

Step 8: Execute() & drop table Syntax for terminating the database & finally use the close().

```
from tkinter import *

def gallery():
    gal=Toplevel()
    gal.maxsize(height=600, width=1300)

    pic1=PhotoImage(file='app1.png')
    pic1.subsample(4,3)
    pic2=PhotoImage(file='app2.png')
    pic2.subsample(4,3)
    pic3=PhotoImage(file='app3.png')
    pic3.subsample(4,3)
    pic4=PhotoImage(file='app4.png')
    pic4.subsample(4,3)
    pic5=PhotoImage(file='app5.png')
    pic5.subsample(4,3)
    pic6=PhotoImage(file='app6.png')
    pic6.subsample(4,3)
    pic7=PhotoImage(file='app7.png')
    pic7.subsample(4,3)
    pic8=PhotoImage(file='app8.png')
    pic8.subsample(4,3)
    pic9=PhotoImage(file='app9.png')
    pic9.subsample(4,3)

    Label(gal, image=pic1, width=400, height=200, bg='black').grid(row=0, column=0)
    Label(gal, image=pic2, width=400, height=200, bg='black').grid(row=0, column=1)
```

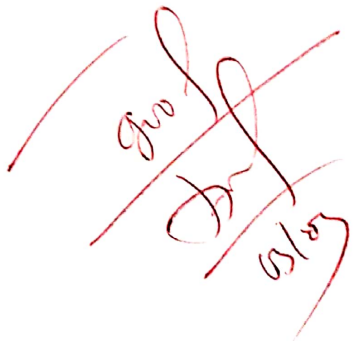
```
Label(gal, image=pic3, width=400, height=200, bg='black').grid(row=0, column=2)
Label(gal, image=pic4, width=400, height=200, bg='black').grid(row=1, column=0)
Label(gal, image=pic5, width=400, height=200, bg='black').grid(row=1, column=1)
Label(gal, image=pic6, width=400, height=200, bg='black').grid(row=1, column=2)
Label(gal, image=pic7, width=400, height=200, bg='black').grid(row=2, column=0)
Label(gal, image=pic8, width=400, height=200, bg='black').grid(row=2, column=1)
Label(gal, image=pic9, width=400, height=200, bg='black').grid(row=2, column=2)

gal.mainloop()

def ram():
  sel="4 GB RAM"
  label.config(text=sel, height=2, width=20, font='Courier 18 bold', bg='white')

def rom():
  sel="64/128/256 GB ROM"
  label.config(text=sel, height=2, width=20, font='Courier 18 bold', bg='white')

def display():
  sel="6.1 INCH"
  label.config(text=sel, height=2, width=20, font='Courier 18 bold', bg='white')

def camera():
  sel="DUAL 12 MP CAMERAS"
  label.config(text=sel, height=2, width=20, font='Courier 18 bold', bg='white')

def battery():
```

```
sel="Li-Ion 3110 mAh"
label.config(text=sel, height=2, width=20, font='Courier 18 bold', bg='white')

def colors():
  sel="BLACK,WHITE and MORE"
  label.config(text=sel, height=2, width=20, font='Courier 18 bold', bg='white')

def submit():
  rt=Tk()
  rt.title('response')
  rt.geometry("500x500")
  rt.maxsize(height=500 ,width=500)
  rt.config(bg='black')
  t=Text(rt, height=200,width=300,bg="black",fg="white")
  t.insert(END,"THANK YOU !! YOU WILL GET YOUR PRODUCT SOON.....KEEP
SHOPPING :)")
  t.pack()
  rt.mainloop


def buy():
  rot=Tk()
  rot.config(bg="black")
  rot.geometry("700x700")
  rot.title("buying")
  ll=Label(rot,text="FILL YOUR DETAILS",
fg="white",height="2",width="20",bg="black",font=("Arial
Bold",20)).grid(row=0,column=0)
  a=Label(rot,text="First Name",fg="white",height="4",bg="black").grid(row=1,column=0
```

```
=Label(rot,text="Email Id",fg="white",height="4",bg="black").grid(row=2,column=0)
=Label(rot,text="Address",fg="white",height="4",bg="black").grid(row=3,column=0)
=Label(rot,text="Contact No",fg="white",height="4",bg="black").grid(row=4,column=0)
=Label(rot,text="Quantity",fg="white",height="4",bg="black").grid(row=5,column=0)
1=Entry(rot).grid(row=1,column=1)
1=Entry(rot).grid(row=2,column=1)
1=Entry(rot).grid(row=3,column=1)
1=Entry(rot).grid(row=4,column=1)
1=Entry(rot).grid(row=5,column=1)
1=Entry(rot).grid(row=6,column=1)
Button(rot, text="SUBMIT",height=2,width=10,bg="maroon",fg="light
k",relief=RAISED,command=submit).grid(row=7,column=0)
Button(rot, text="EXIT",height=2,width=10,font="bold",bg="maroon",fg="light
k",relief=RAISED,command=rot.destroy).grid(row=7,column=1)
rot.mainloop()


ot=Tk()
ot.title('Apple')
ot.maxsize(height=750, width=1300)
ot.config(bg='white')


f=Frame(root, bg='white', height=400, width=200)
f.grid(row=0, column=0)


f=Frame(root, bg='white', height=400, width=250)
f.grid(row=0, column=1)
```

```
Label(lf, text='APPLE IPHONE 11',height=2, width=20, font='Courier 15 bold',
bg='white').grid(row=0, column=0)

photo=PhotoImage(file='apple2.png')
photo.subsample(3,4)

photo1=PhotoImage(file='apple1.png')
photo1.subsample(4,3)

photo2=PhotoImage(file='logo1.png')
photo1.subsample(4,3)

Label(lf, image=photo, bg='white', height=400, width=500).grid(row=1, column=0, padx=10,
pady=22, ipadx=3, ipady=3)
Label(rf, image=photo1, bg='white').grid(row=1, column=0, padx=10, pady=10, ipadx=3,
ipady=3)
Label(rf, image=photo2, bg='white').grid(row=0, column=0, padx=10, pady=10, ipadx=3,
ipady=3)

toolbar=LabelFrame(lf, text='SpecificationS', font='Courier 15 bold', width=200, height=100,
bg='white')
toolbar.grid(row=2, column=0, padx=15, pady=15)

Button(toolbar, text='Gallery', height=1, width=16, font='Courier 12 bold',
command=gallery).grid(row=1, column=0)
Button(toolbar, text='Display', height=1, width=16, font='Courier 12 bold',
command=display).grid(row=1, column=1)
Button(toolbar, text='RAM', height=1, width=16, font='Courier 12 bold',
command=ram).grid(row=2, column=0, padx=10, pady=10)
```

```
utton(toolbar, text='ROM', height=1, width=16, font='Courier 12 bold',
ommand=rom).grid(row=2, column=1, padx=10, pady=10)

utton(toolbar, text='Camera', height=1, width=16, font='Courier 12 bold',
ommand=camera).grid(row=3, column=0)

utton(toolbar, text='Battery', height=1, width=16, font='Courier 12 bold',
ommand=battery).grid(row=3, column=1)

utton(toolbar, text='Colors', height=1, width=16, font='Courier 12 bold',
ommand=colors).grid(row=4, column=1, padx=10, pady=10)

Button(toolbar, text='BUY NOW', height=1, width=16, font='Courier 12 bold',
command=buy).grid(row=4, column=0, padx=10, pady=10)


Button(rf, text='EXIT', height=1, width=5, font='Courier 12 bold', bg='black',
fg="white",relief=FLAT, command=root.destroy).grid(row=4, column=1)


op=LabelFrame(rf, text='Specs Info', bg='white', font='Courier 15 bold', height=3, width=21)

op.grid(row=2, column=0)


Label(op, text='Info will Appear Here !', font='courier', bg='white', height=4,
width=30).grid(row=2, column=0)


label=Label(op)

label.grid(row=2, column=0)


mainloop()
```



APPLE IPHONE 11

Specifications

Gallery	Display
RAM	ROM
Camera	Battery
BUY NOW	Colors

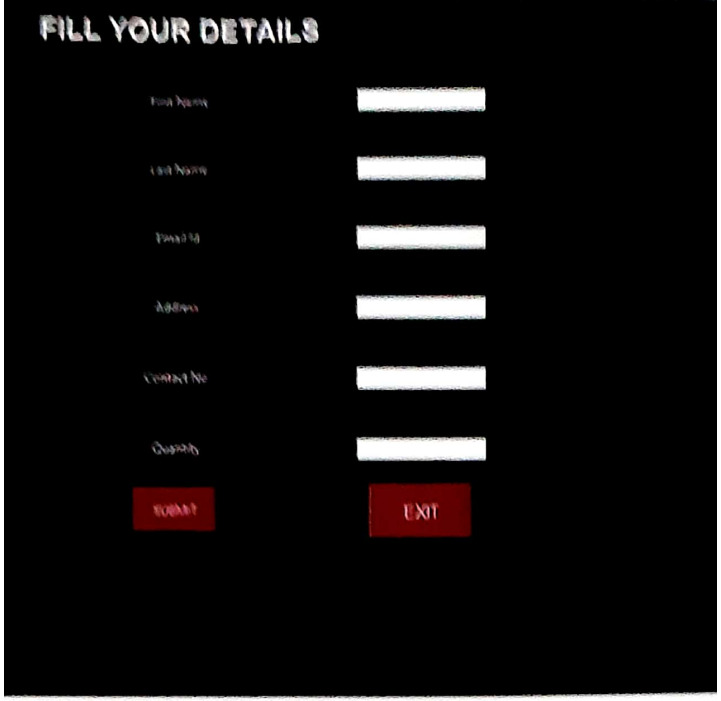
Specs Info

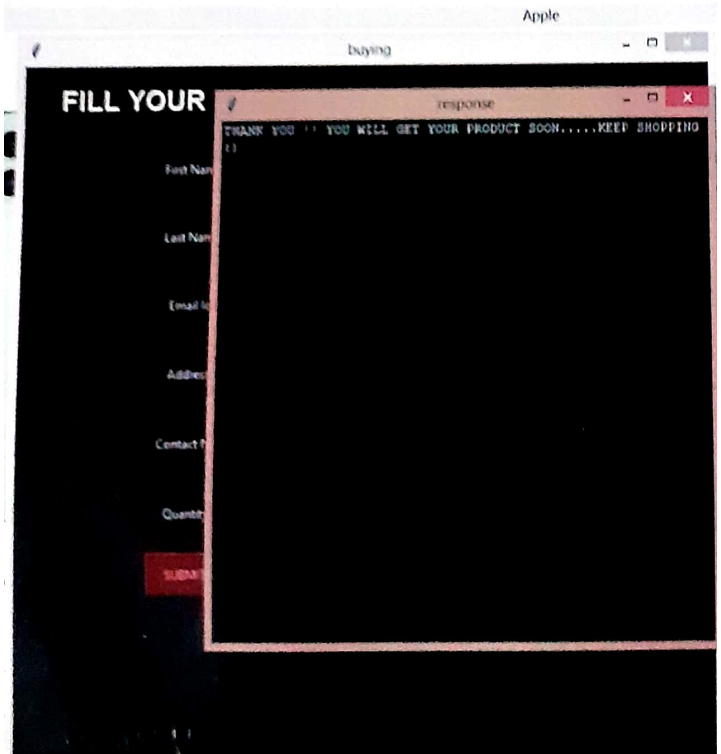
Info will Appear Here !

EXIT

Apple

APPLE IPHONE 11





```
from tkinter import*
import sqlite3

root=Tk()
root.title('DETAILS')
root.geometry("1200x1200")
root.maxsize(height=1500 ,width=1500)
root.config(bg='black')

FirstName=StringVar()
LastName=StringVar()
Email=StringVar()
gender=StringVar()
var=IntVar()

def database():
    name1=FirstName.get()
    name2=LastName.get()
    email=Email.get()
    gender=var.get()
    conn=sqlite3.connect('registration.db')
    with conn:
        cursor=conn.cursor()
    cursor.execute('create table details(First Name TEXT,Last Name TEXT,Email
TEXT,Gender TEXT )')
    cursor.execute('insert into details(First Name,Last Name,Email,Gender)
values(shaikh,Rizwana,shkriz@gmail,female),(sayed,tatheer,sydtat@gmail,female),(sa
fin,sydarf@gmail,male)',(name1,name2,email,gender))
```

```
    conn.commit()

Label(root,text="Registration form",width=20,font=("Bold",20)).place(x=90,y=53)

l1=Label(root,text="First Name",width=20,font=("Bold",10)).place(x=80,y=130)
e1=Entry(root,text=FirstName).place(x=240,y=130)

l2=Label(root,text="LastName",width=20,font=("Bold",10)).place(x=68,y=180)
e2=Entry(root,text=LastName).place(x=240,y=180)

l3=Label(root,text="Gender",width=20,font=("Bold",10)).place(x=70,y=230)
Radiobutton(root,text="Male",padx=5,variable=var,value=1).place(x=235,y=230)
Radiobutton(root,text="Female",padx=20,variable=var,value=2).place(x=290,y=230)

l4=Label(root,text="Email",width=20,font=("Bold",10)).place(x=70,y=280)
e3=Entry(root,text=Email).place(x=240,y=280)

Button(root,text="Submit",width=20,bg='brown',fg='white',command=database).place(x=180
,y=380)



root.mainloop()
```