THAKUR COLLEGE OF SCIENCE & COMMERCE

NAAC Accredited with Grade "A" (3rd Cycle)

THAKUR TRUSTS

ISO 9001 : 2015 Certified

Degree College

Computer Journal

CERTIFICATE

SEMESTER II **UID No.** _____

Class FY.B.Sc.CS **Roll No.** 1753 **Year** _____

This is to certify that the work entered in this journal is the work of Mst. / Ms. MISHRA ANUSHKA ABHAY

who has worked for the year 2019-20 in the Computer Laboratory.

Teacher In-Charge

Head of Department

Date : _____

Examiner

☆☆ INDEX ☆☆

```
Fileobj = open("abc.txt","w") # file open (write mode)
fileobj.write("computer science subjects "+"\n")
fileobj.write("DBMS in python in DS \n")# file write
file obj.close() # file close

Fileobj = open("abc.txt","r") # read mode
#read()
str1 = fileobj.read()
print("The output of read method :", str1)
File obj.(closec)
>>> ('The output of read method : ''computer science
    subjects in DBMS in python 'in DS \n')

#readline()
fileobj = open("abc.txt","r")
str2 = fileobj.readline()
print("The output of readline method :", str2)
file obj.close()
>>>("The output of readline method :','computer
    science subject \n')

# readlines()
fileobj = open("abc.txt","r")
str3 = file obj.readlines()
print("The output of readlines method:", str3)
fileobj.close()
>>> ('The output of readline method :", ['computer
    science subjects \n, 'DBMS in', 'Python \n', 'DS \n']
```

Objective: Demonstrate the use of different file accessing modes, different attributes read method

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

Step 2: Now open the file in read mode and then use read(), readline(), & readlines() & ~~state~~ store the output in variable and finally display the contents of variable.

Step 3: Now we the file object for finding the name of the file, the file mode in which it is opened whether the file is still open or close & 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 object in 'w+' mode that is the update mode & write contents.

Step 5: Open fileobject 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 fileobject in append mode open write method write content close the fileobject again open the fileobject in readmode & display the append output.

Step 7: Open the fileobject in read.mode, declare a variable & perform fileobject dot tell method & store the output consequently in variable.

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

```
# File attributes
a = fileobj.name
print("name of file (name attribute):", a)
>>> ('name of file (name attribute); 'abc.txt')
b = fileobj.closed
print ("closed) attribute :", b)
>>> ('(close) attribute :', 'True')
c = fileobj.mode
print ("Filemode.", c)
>>> ('File mode', 'r')
d = file.obj.softspace
print ("softspace", d)
>>> ('softspace :", 0)
```

```
# w + mode
fileobj.open ("abc.txt", "w+")
fileobj.write ("Computer")
Fileobj.close()
```

```
# write mode
Fileobj = open ("abc.txt", "w+")
File obj.write ("DBMS")
Fileobj.close()
```

```
# r+ mode
Fileobj = open ("abc.txt", "r+")
str1 = fileobj.read(6)
print ("output of r+ ", str1)
fileobj.close()
>>> ('output of r+', 'Computer')
```

```
# read mode
file obj = open ("abc.txt", "r")
str2 = fileobj.read()
print ("output of read mode:",
       str2)
>>> ('output of read mode:',
    'Computer')
```

```
# Append mode
fileobj = open ("abc.txt", "a")
fileobj.write ("data structure")
fileobj.close()
fileobj = open ("abc.txt", "r")
str3 = fileobj.read()
print ("output of append mode :", str3)
file obj.close()
>>>('output of append mode :', 'computer', 'data structure')

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

# seek()
file obj = open ("abc.txt", "r")
str4 = fileobj.seek(0,0)
str = fileobj.read(10)
print ("The beginning of the file :", str3)

# Finding length of different lines exist within lines
fileobj = open ("abc.txt", "r")
str9 = fileobj.readlines()
print ("output :" str9)
for line in str9:
        print (len(line))
    fileobj.close()
>>> ('output :', ['college database'])
```

Step 9 : Open fileobject with read mode also use the readlines method & store the output consequently in & print the same for counting the length use the for condition statement & display the length.

Step 10 : Open file obj with readmode also apply the readmode. On the fileobj & store it in a variable use while conditional statement to check the length of the content in the variables : if the content is greater than 0, then print the content & use end syntax in print statement to add (hash) special symbol in content of file

16/1

```
# content with special character
fileobj = open ("abc.txt", "r")
content = fileobj.read(1)
while len(content) > 0:
    print(content, end = "#")
    content = file.read(1)
```

>>) H # E # L # L # O #
I # A # M # A # I
H # I # M # A # N # S # H # U
D # E # V # L # O # P # E # R

```
# iter() & next()
mytuple = ("banana", "apple", 'grapes")
my iter 1 = iter (mytuple)
pomt (next (my iter 1))
myiter2 = iter (mytpple)
print (next (my iter))
myiter3 = iter (mytuple)
print (next (myiter1))
```

>>> banana
apple
grapes

```
# For loop
mytuple 1 = ("Kelvin", "Surat", "Mumbai")
for x in multiple1:
    print (x)
```

>>> Kelwin
Surat
Mumbai

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

PRACTICAL No. 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 & 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 statement. 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 of 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 type casting with map method Declare a 'lambda' i.e annonymus function & 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 statement do check whether the number is even and odd & return respectively.

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

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

```
value = list(map(lambda x: x(r).function))
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"
list(map(even, listnum))
```

output:
```
[0, 4, 1, 2]
```

```
# odd numbers
class odd:
    def __iter__(self):
        self.num=
        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
7
9
11
13
```

```
# Odd numbers using iter method
class myclass:
    def __iter__(self):
        self.a = 1
        return self
    def __next__(self):
        if self.a<=10:
            x = self.a
            self.a += 1
            return x
        else:
            raise StopIteration

myobj = myclass()
myiter = iter(myobj)
for x in myiter
    print(x)
```

output:

```
1
2
3
4
5
6
7
8
9
10
```

Step

Step

Step

Step

Step

Step

Step 10 : Define an object of a class.

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

Step 12 : Define a iter method with a argument and initialize it to a first value.

Step 13 : For extracting the next element from the container. Use the next method with an argument and compare no. of elements reserved in a container.

Step 14 : Now create an object from the given class and pass the object as an argument to the iter method

Step 15 : Now using the conditional statement display all the value

Step 16 : Define a list variable with a given set of numbers.

Step 17 : Define an empty list which will contain output use the for conditional statement & subsequently use append() method. for finding square value & finally print statement for the output.

```
#without using map()
list1 = [1,2,3,4,5]
empty = []
for i in list1:
    empty.append(i**2)
print(empty)
```

Output:

```
[1]
[1,4]
[1,4,9]
[1,4,9,16]
[1,4,9,16,25]
```

Program :

```
try :
    fo = open ("Anushka", "w")
    fo.write ("Computer science")
except IOError:
    print ("Computer Science")
else :
    print ("Operation successful")
```

=>) Operation successful

Practical No. 3.

Algorithm :

Step 1 : Use the try block to define the normal course of action. For eg. : Define the file .obj and open the file in the write mode and write some content onto the file.

Step 2 : Use the except block with the I/O error as an environment error & conve the appropriate message to user else display the message that the operation is carried out successfully.

⟹ Program for demonstrating the use of value error in the given program statement

Algorithm :

Step 1 : Accept the value from the user and if it is a valid value display the entered value and terminate the condition by using break statement.

Step 2 : Define the except with the value error as a keyword and display the appropriate message.

step 3 : You can define the multiple exception using the except statement for finding the different category of errors

program :

```
try :
  a = int(input ("Enter the number"))
except.valueError :
  print ("Arithmatic error")
else :
  print ("successfull")
try :
  fo = open ("abc.txt ", "r")
  fo.write ("hello, how are you?.")
except IOError :
  print ("Environment error")
else :
  print ("successfull")
```

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 value (segregation)

```
import re
pattern = r'\d+'
string = 'Hello123, howdy789, 45hours'
output = re.findall(pattern, string)
print(output)
```

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

split()

```
import re
pattern = r'\d+'
string = 'Hello123,Vicky475'
Output = re.split(pattern, string)
print(output)
```

>>> ['Hello', 'Vicky']

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 character. Declare string value. Use the findall () with arguments and print the same.

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

Step 4 : import re modula 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 as search gives memory coration using group () it will show up the matched string.

Step 6: import re module declare list with numbers. Use the conditional statement here we have used up for ~~fo~~ condition statement. Use if condition for checking first number is either 8 or 9 and next number are in range of 0 to 9 and check whether the entered numbers are equal to 10, if criteria ~~are~~ matches print cell number matches otherwise print failed.

Step 7: import re module declare a string use the module with findall() ~~and print~~ 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 separating the host & domain name. Use ~~are~~ the findall() and print the output respectively.

Step 9: import re module. enter a string use pattern to display only two elements of the particular string

#4 no space or sub()

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

>>> abcdefghi

#5

```
import re
string = "Python is an interpreted language"
v = re.search('\A Python', string)
Print(v)
print(v.group())
```

>>> <_sre.SRE Match object at 0x0281DF04>
Python

#6 Verifying the given set of cell No.

```
import re
list = ["8007266554", "7779997551", "8208236431"]
for value in list:
    if re.match(r'[8-9]{1}[0-9]{9}', value
        len(value) == 10):
        print("Cell Number found")
    else:
        print("Match Failed")
```

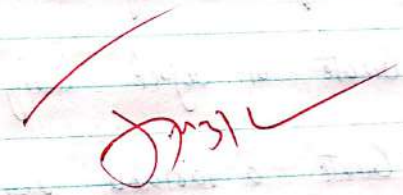
>>> Cell Number found
Match failed
Cell Number found

```
# vowel
import re
str1 = 'Hello Python is a programming language'
output = re.findall(r'\b[aeiou AEIOU]\w+', str1)
print(output)
>>> ['is', 'a']
```

```
# host domain
import re
seq = 'abc123.tscs@edureka.com .pycs@edu.com'
pattern = r'[\w]:-]+[\0]--]'
output = re.findall(pattern, seq)
print(output)
>>> ['abc123.tscs', 'edureka.com', 'pycs', 'edu.com']
```

```
# count of first 2 letters
import re
s = 'mr.a, ms.b, ms.c, mr.t'
p = r'[/m,/mr/]+
o = re.findall(p, s)
print(o)
m = 0
f = 0
for v in o:
    if (v == 'ms'):
        f = f+1
    else:
        m = m+1
print("No. of Males:", m)
print("No. of females", f)
>>> ['Mr', 'Ms', 'Ms', 'Mr']
['No. of Males': 2]
['No. of females': 2]
```

Use findall() declare two variables with initial value as zero use for condition and subsequently use the if condition check whether condition satisfy ~~check whether~~ add up the or else increment value. And display the values subsequently.



Practical No: 5

Topic : GUI Components (pack method)

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 : Creat a variable from the text method and position it on the parent windaw.

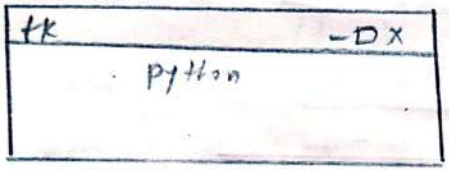
step 3 : Use the pack() along with the object created from the text() and use the parameter.

1) side = LEFT, padx = 20
2) side = LEFT, pady = 30

Creation of parent window

```
from Tkinter import *
root=Tk()
l = Label (root, txt = "Python")
l.pack()
root.mainloop()
```

Output:



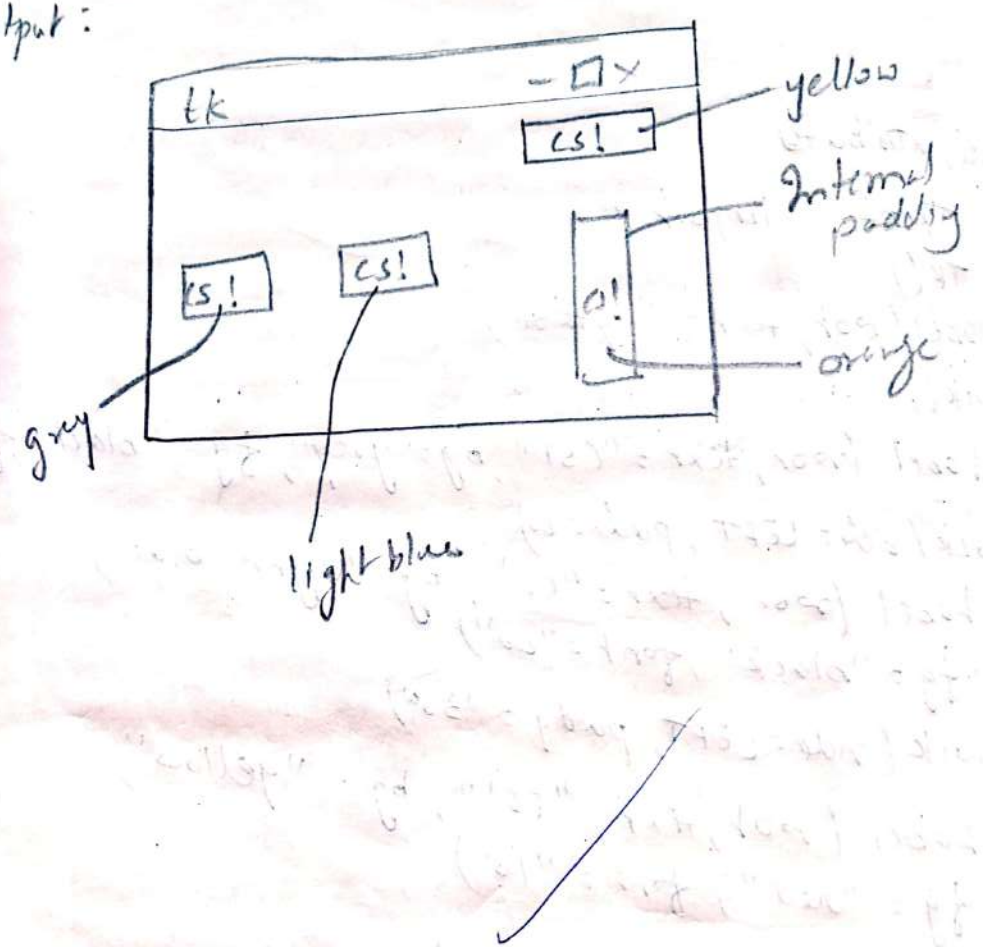
#

2 : label, attributes

```
from Tkinter import *
root Tk()
l = label(root, text = "python")
l.pack()
l1 = Label (root, text = "CS!", 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, "CS!", bg = "yellow",
     fg = "red", font = "10")
l3 = pack (side = TOP, ipadx = 40)
```

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

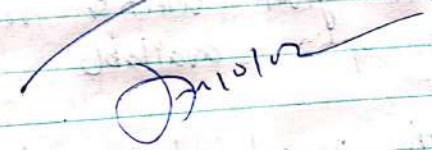
Output:



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.



PRACTICAL - ~~5(B)~~ 5(B)

AIM : Gui components - (~~pack~~ radio button method)

~~#~~ 1 :

Step 1 : Import the relevant methods from the tkinter library create an object within 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 defines a function which tells the user about the given selection mode from multiple option available.

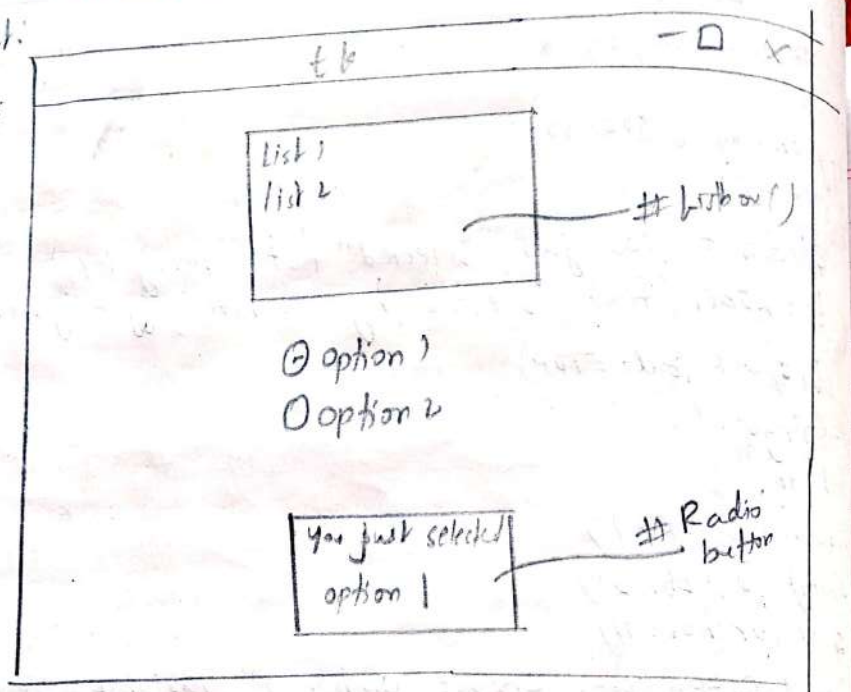
Step 4 : Now define the parentwindow and defines the option with control variable.

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

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 = Radio Button (root, text = "option 2", variable = var,
    value = "option 2", command = select)
r2.pack (anchor = N)
root.mainloop()
```

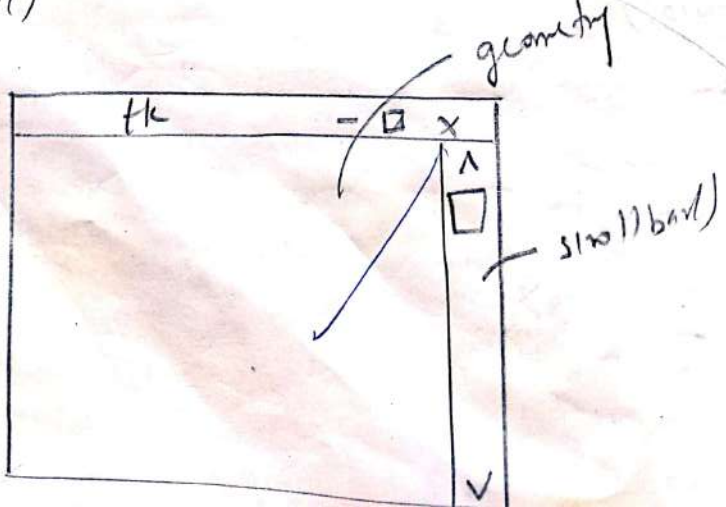
Output:



2:

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

Output:



Step 6 : Create an object from radio button# which will take following arguments & parent window object, text variable which will take the values option no : 1, 2, 3... variable arguments, corresponding value and trigger 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 window.

Step 4 : Create an object and use the scrollbar()

Step 5 : Use the pack() along with scrollbar object with side and fill attributes.

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 any other suitable pixel value.

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

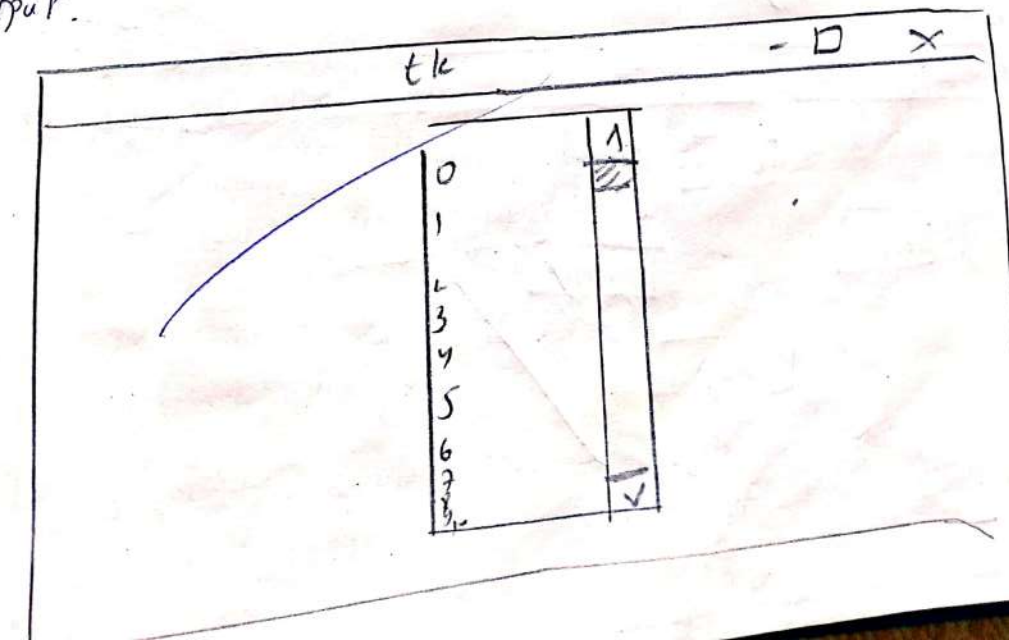
Step 5 : Use the frame widget along with the parent object created & use the pack method.

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

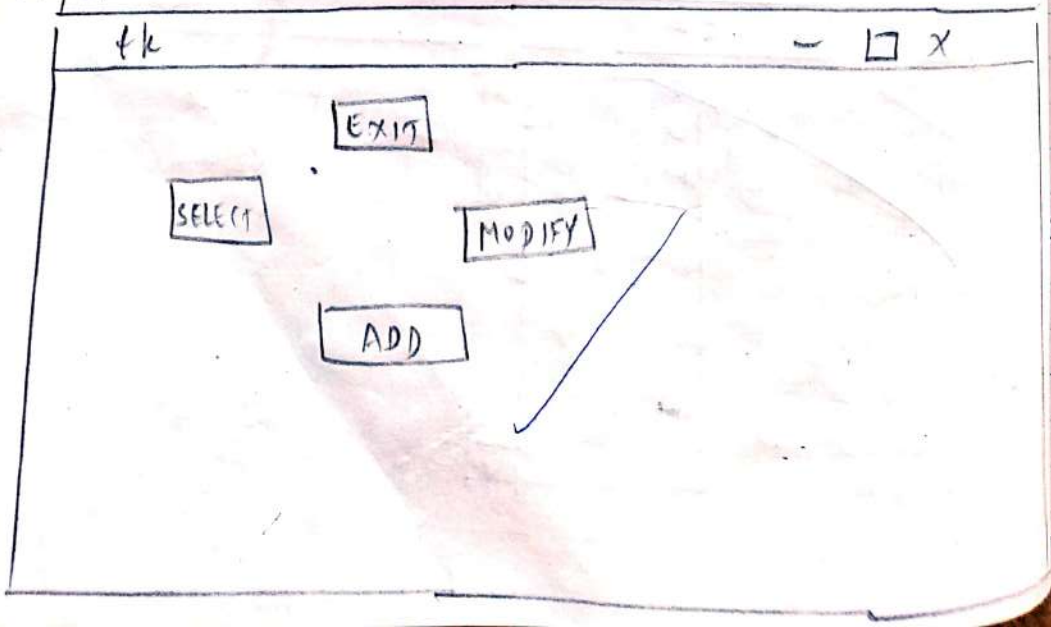
```
#Using frame widget
from tkinter import *
window = Tk()
window.geometry("680x500")
Label(window, text="numbers:").pack()
frame = Frame(window)
frame.pack()
listNodes = Listbox(frame, width=20, height=20, font=
        ("Times New Roman", 10))
listNodes.pack(side="left", fill="y")
scrollbar = scrollbar(frame, 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:



```
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="blu
b2 = Button(frame, text="modify", activebackground="yellow", fg="blu
b3 = Button(frame, text="ADD", activebackground="blue", fg="red")
b4 = Button(frame, text="EXIT", activebackground="red",
    fg="red")
b1.pack(side="LEFT", padx=20)
b2.pack(side="right", padx=30)
b3.pack(side="bottom", pady=20)
b4.pack(side="top")
```

Output:



Step 3 : 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 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 & subsequently define the button object placed onto the given frame with the attributes as text, active background & 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".

Step 8 : Create another button object & place it on to the RIGHT frame & label the button as ADD.

Step 9 : Add another button & put 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().

Practical : 5 (c)

AIM : GUI components.

Step 1 : Import the relevant methods from tkinter library.

Step 2 : Import tk Messagebox.

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

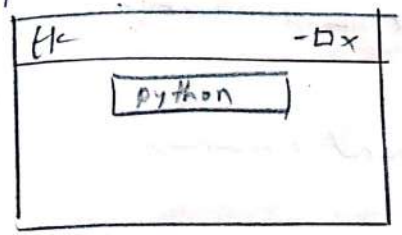
Step 4 : Define a function which will use tkmessagebox with showinfo method along with info window attribute.

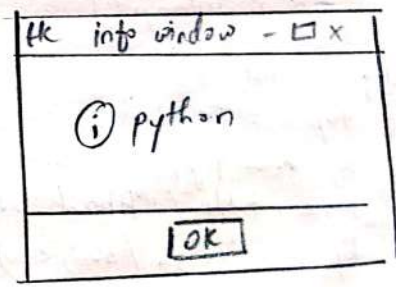
Step 5 : Declare a button with parent window object along with the command attribut.

Step 6 : Place the button widget onto the parent window & finally call mainloop() for triggering of the events called above.

```
# messagebox
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 FRANSISCO\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()

def second():
    top2 = Tk()
    top2.config(bg = "orange")
    top2.title("About Us")
    top2.minsize(300,300)
    L = Label(top2, text = "Created by: Anwhka Mishra\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()
```

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.

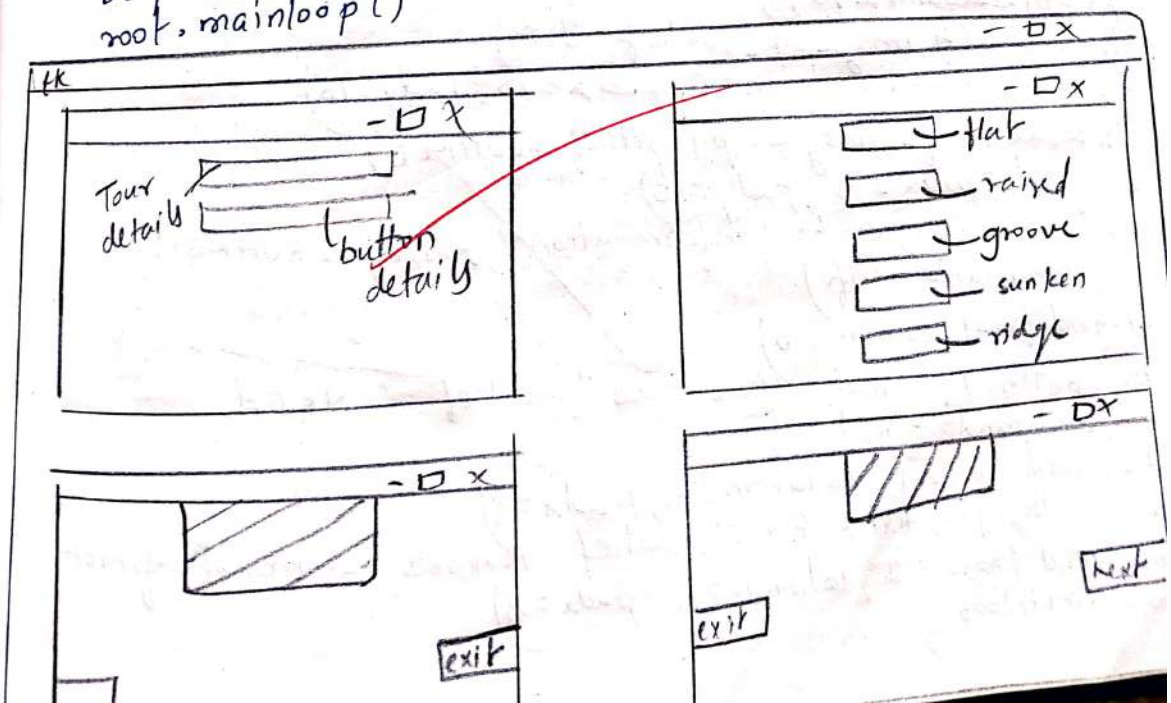
Step 3: Define a function main, declare parent window object and use config(), title(), minsize(), Label() as well as button() & use pack() & mainloop() simultaneously.

Step 4: Similarly define the function second & use the attribute accordingly.

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

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

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



```
from Jkinter import *
root=TK()
root=config(bg="grey")
def finish():
    messagebox.askokcancel("Warning", This will end program
    quit()
def info():
    list1=Listbox()
    list1.insert(1,"Co.Name:Apple")
    list1.insert(2,"Product: iphone")
    list1.insert(3,"language: swift")
    list1.insert(4,"OS: IOS")
    list1.grid(ipadx=30)
def aboutus():
    list2=Label(text="About us")
    list2.grid(ipadx=30)
    list3=Label(text="Steve Jobs theatere March 2020")
    list3.grid(ipadx=24)
p1=PhotoImage(file="cloudload.gif")
f1=Frame(root,height=35,width=5)
f1.grid(row=,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="Inform-ation",relief=SUNKEN,
    command=info)
b1.grid(row=1,colum=0)
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()
```

PRACTICAL : 5(D)

Aim : GUI components

Step 1 : Import relevant methods from the tkinter library.

Step 2 : Creat parent window object and use the config method alog 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 & finally use the grid() with ipadx attribute.

Step 5 : Define a function about us with label widget and text attribute & subsequently use the grid().

Step 6 : Use photoimage widget with file and filename with gif attribute.

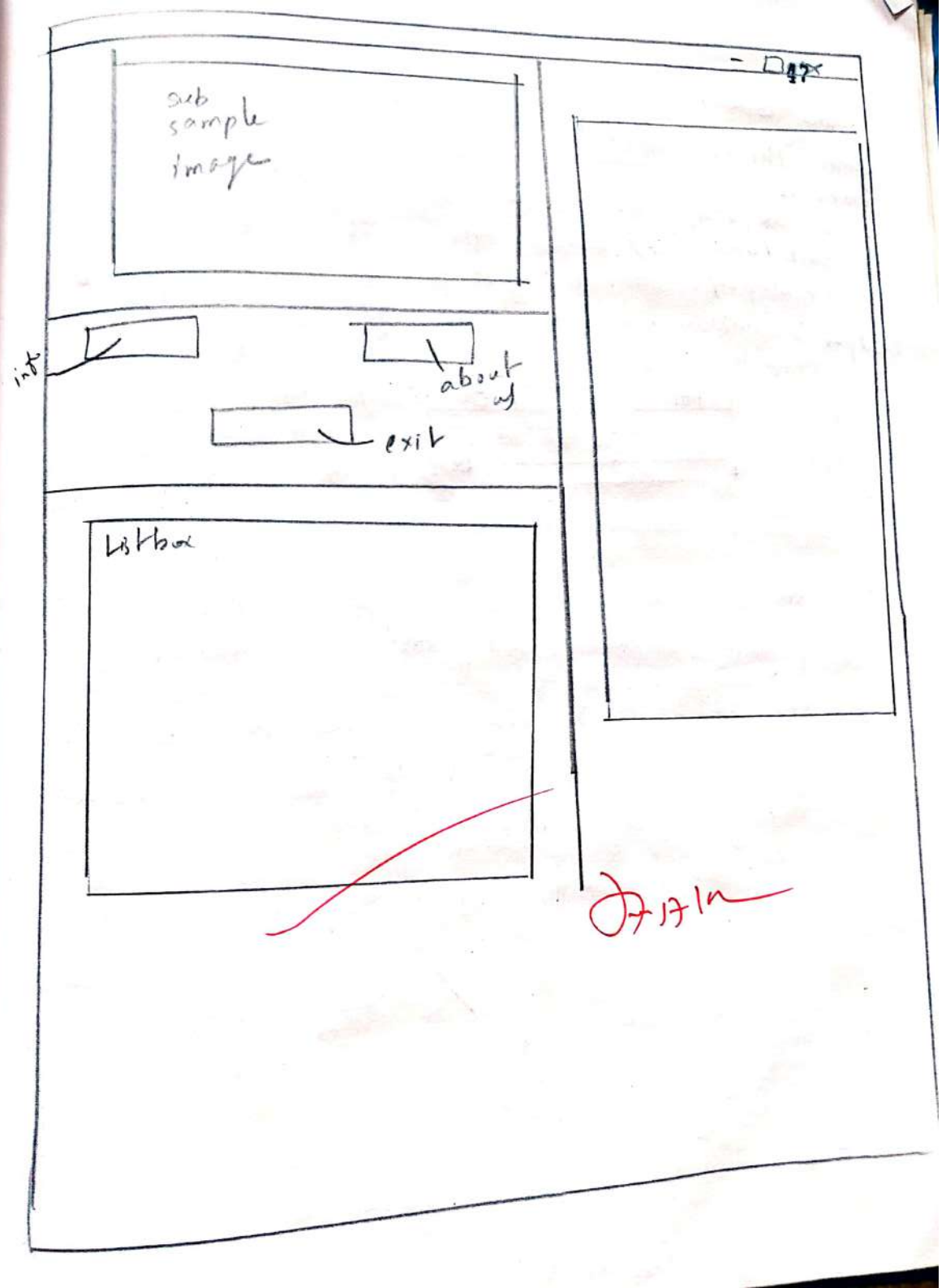
Step 7 : Create a frame object along with the Frame() along with parent window object height & width specified & subsequently use the grid() with row & column attribute specified.

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

Step 9 : Create another object & use the ~~sob~~ subsample (5, 4).

Step 10 : Use label widget along with the frame object, relief attribute and subsequently use the grid()

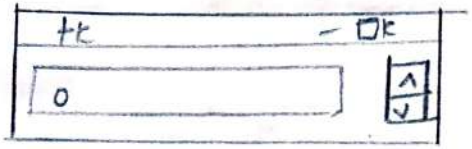
Step 11 : Now create button object dealing with different section of frame


sub
sample
image
int
about
exit
Listbox

Source code:

```
from tkinter import *
root = Tk()
s1 = Spinbox(root, from_ = 0, to_ = 10)
s1.pack(anchor = S)
root.mainloop()
```

Output:



PRACTICAL - 15(E)

Aim: To make use of spinbox widget, paned window & canvas widget.

Spinbox Widget:

Step 1: Create an object from the Tk method & subsequently create an object from the spinbox method.

Step 2: Make the object so created onto the parent window & trigger the corresponding event

Step 3: Use the pack method to provide the direction using anchor method.

Step 4: Use the mainloop method to terminate

* Panned window:

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

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

Step 3: Similarly create another label object & place it onto the 2nd ~~placed~~ panel window object & add onto the 2nd pane.

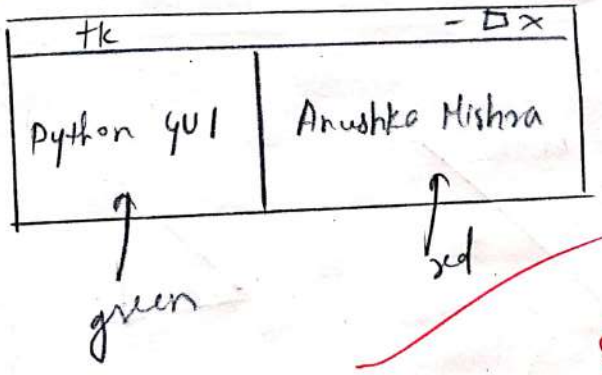
Step 4: Create a second paned window object & add it onto the 1st paned window with orientation specified.

Step 5: Now use the mainloop method to terminate

Source code :

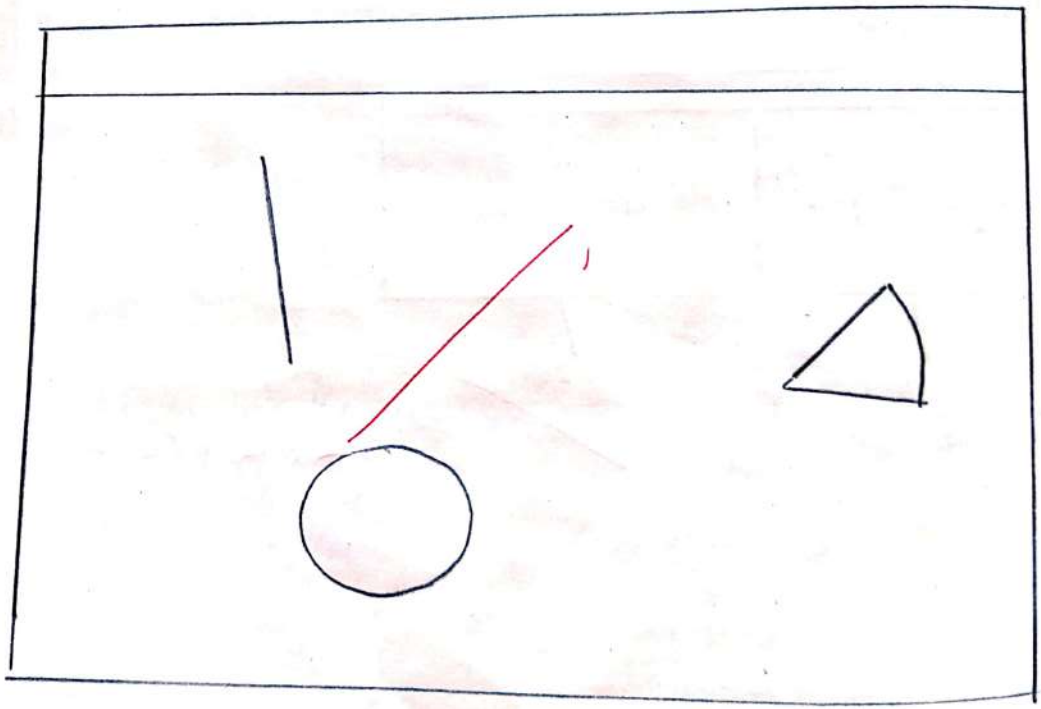
```
from tkinter import *
root=Tk()
p=panedwindow(bg="red")
p.pack(fill=BOTH, expand=10)
L1=Label(p, text="Python GUI", bg="green")
p.add(L1)
p1=panedwindow(p, orient=VERTICAL, bg="blue")
p.add(p1)
L2=Label(p1, text="Anushka Mishra", bg="red")
p1.add(L2)
root.mainloop()
```

Output :



Jan 17/22

```
from tkinter import*
root=Tk()
c1= canvas (root, height=400, width = 400, width=400,
            bg = "black")
oval = c1.create_oval(20, 140, 150, 260, fill = "white")
line = c1.create_line( 30, 40, 50, 60, fill = "blue")
arc = c1.create_arc (20, 40, 60, 80, fill = "yellow")
c1.pack()
root.mainloop()
```



Canvas widget

Step 1 : Use the tkinter method and create an object from the canvas method & use the attribute height, width, bgcolor & the parent window object.

Step 2 : Use the method create oval, create line and create arc along with the canvas object so created and use the co-ordinate value. Also use the fill attribute to assign various colours.

Step 3 : Now call the pack method & mainloop method.

24/2

PRACTICAL No. 7

AIM : To demonstrate the use of database connectivity.

#1

Step 1 : Import the dbm library and use the open method 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 and the 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 then display the message that particular found / Match or does not found / unmatched

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

7) import dbm
db = dbm.open ("database", "c")
db["name"] = "name"
if db["name"] == None:
    print ("database emty")
else:
    prit ("Match found")
db.clos

```
import os, sqlite3
con = sqlite3.connect("student db")
c = con.cursor()
c.execute("CREATE TABLE student (roll INTEGER, nam
                                  TEXT)")

c.execute("INSERT INTO student values
          (1777, 'Khushboo'), (1753, 'Anushka')")
c.execute("SELECT * from student WHERE roll="1
print(c.fetchall())
con.commit()
c.close
```

Output:

```
[(1753, 'Anushka')]
```

Algorithm :

Step 1 : Import corresponding library to make database connection, os and sqlite3

Step 2 : Now create the connection object using sqlite3 library and the connect() for creating new database.

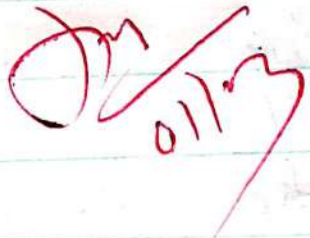
Step 3 : Now use the execute() for creating the table within the column name and respective datatype.

Step 4 : Now create cursor object using the cursor() of from the connection object created.

Step 5 : Now with the 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 select from where clause.



GUI PROJECT

```
from tkinter import*
root = Tk()
root.geometry('500x500')
root.title("Registration Form")
L=Label(root, text = "Registration
form", width = 20, font = ("bold",20))
L.pack()
L4 = Label (root, text = "FullName", width = 20, font = ("bold", 10))
L4.pack()
E1 = Entry (root)
E1.pack()
L2 = Label (root, text = "Email", width = 20, font = ("bold", 10))
L2.pack()
E2 = Entry (root)
E2.pack()
L3 = Label (root, text = "Gender", width = 20, font = ("bold",
    10))
L3.pack()
var = IntVar()
R1 = Radiobutton (root, text = "Female", padx = 5, variable = var,
    value = 1)
R1.pack()
R2 = Radiobutton (root, text = "Fmale", padx = 20, variable = var,
    value = 2)
R2.pack()
L4 = Label (root, text = "Age:", width = 20, font = ("bold", 10))
L4.pack()
```

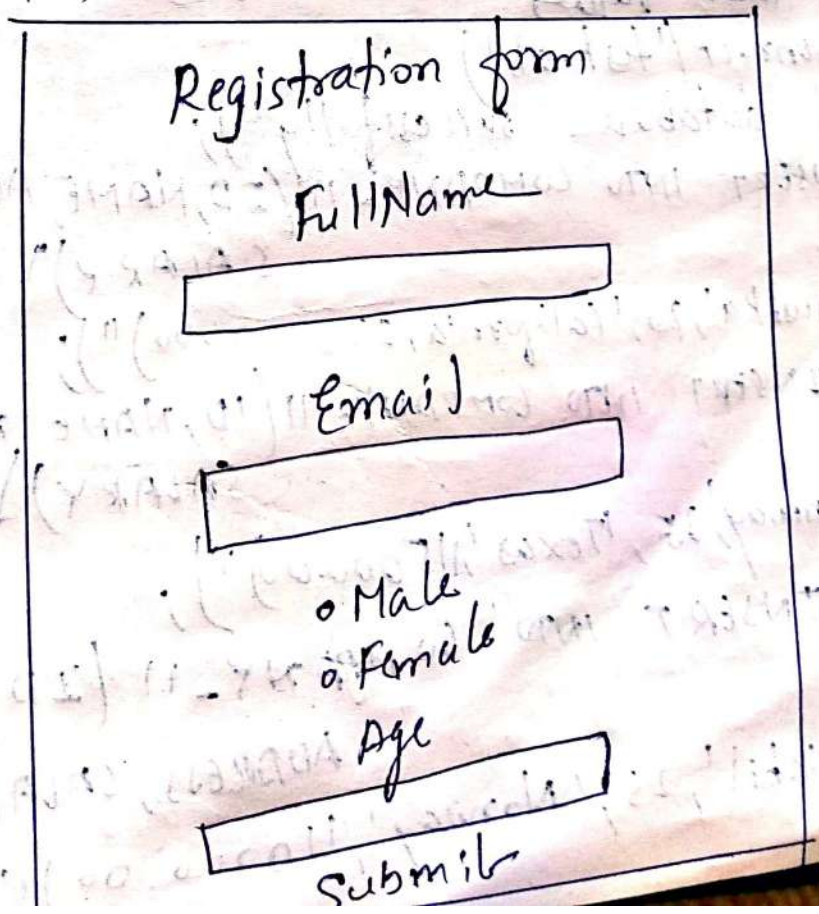
```
E2 = Entry(root)
E2.pack()
B1 = Button(root, text='Submit', width=20, bg=
  'brown', fg='white')
B1.pack()
from tkinter import*
root = Tk()
root.geometry('500x500')
root.title("Registration Form")
L = Label(root, text="Registration form", width=20,
  font=("bold", 20))
L.pack()
L1 = Label(root, text="Full Name", width=20, font=
  ("bold", 10))
L1.pack()
E1 = Entry(root)
E1.pack()
L2 = Label(root, text="Email", width=20, font:
    ("bold", 10))
L2.pack()
E2 = Entry(root)
E2.pack()
L3 = Label(root, text="Gender", width=20, font:
    ("bold", 10))
L3.pack()
var = IntVar()
R1 = Radiobutton(root, text="Male", padx=5, variable=var
value=1)
```

```
R1.pack()
R2=Radiobutton(root, text="Female", padx=20, variable=var,
  value=2)
R2.pack()
Ly=Label(root, text="Age:", width=20, font=("bold", 10))
Ly.pack()
E2=Entry(root)
E2.pack()
B1=Button(root, text='Submit', width=20, bg='brown',
   fg='white')
B1.pack()
```

it is used for displaying the registration form on the window

```
root.mainloop()
print("registration form successfully created")
root.mainloop()
```

*Output:



Database project

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 (1753,'Anwikka',20,'California',220000.00)");
conn.execute("INSERT INTO COMPANY_11 (ID,NAME,AGE,ADDRESS,
                                      SALARY)
VALUES (17 11,'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(1313,'Atharva',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 = 1753")
conn.commit
print("Total number of rows update:",
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("\n operation done succesfully");
conn.close()
```

DELETION using where clause

```
conn = sqlite3.connect('test.db')
print("\n"DELETION");
conn.execute("DELETE from COMPANY_11 where
  ID = 1713;")
conn.commit
print("Total number of rows deleted:",
conn.total-changes)
cursor = conn.executed("SELECT id, name, address,
salary from COMPANY_11")
```

```
for row in cursor:
    print "ID = ", row[0]
    print "NAME = "
    print "ADDRESS = ", row[1]
    print "SALARY = ", row[2]
    print "\n"
    print "Operation done successfully";
conn.close()
```

#TRUNCATE DDL statement

```
conn = sqlite3.connect('test.db')
cursor = conn.execute("DROP TABLE company-1");
print("\n VALUES truncated.")
conn.close
```

Output:

```
Opened database successfully
Table created successfully
Opened database successfully
Records created successfully
('ID = ', 1753)
('NAME = ', u'Anushka')
('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'Atharva')
('ADDRESS=', u'New York')
('SALARY=', 115000.0)

Operation done successfully
UPDATION OF VALUES
('Total number of rows update:', 1)
('ID=', 1713)
('Name=', u'Anushka')
('ADDRESS=', u'California')
('SALARY=', 220000.0)

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

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

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

Operation done succesfully
DELETION
('Total number of rows deleted :', 1)
ID = 1753
NAME = Anushka
ADDRESS = California
SALARY = 220000.0

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

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

Operation done successfully

VALUES truncated!