1.	Demonstrate the use of different file accessing mode differences attributes read method	25	26/11/19	
2.	Iterators	29	3/12/19	
3.	Exceptions	31	17/12/19	
4.	Regular expressions	35	24/12/19	
5(A)	Gui Components	39	7/01/20	
5(B)	Gui Components - Radio Button Scrollbar	41	14/1/20	
5(C)	GUI Components - Route, message box	45	21/1/20	
5(D)	GUI - Image	49	28/1/20	
5(E)	GUI - Spinbox, Pane Window & Canvas	51	4/2/20	
(6)	Database Connectivity	53	18/2/20	

PRACTICAL-1

OBJECTIVE:- Demonstrate the use of different file accessing modes difference attributes read method

Step 1:- Create a file object using open method and use the write access mode followed by writing some Contents onto the file and then closing the file

Step 2:- Now open the file in read mode and then use read(), readline() and readlines() and finally display the Contents of Variable.

Step 3:- Now use the fileobject for finding the name of the file, the file mode in which is opened whether the file is still open or close and finally the output of the Softspace attribute


```
fileobj = open("abc.txt", "w") # File open(write mode)
fileobj.write("Computer Science Subjects"+"\n")
fileobj.write("DBMS \n Python \n DS \n")  # file write
fileobj.close() # file close

file obj = open("abc.txt", "r") # read mode
  # read()
    str1 = fileobj.read()
    Print ("The output of read method :", str1)
    file obj.close()
>>> ('The outputs of read method :', 'Computer Science Subjects
    \n DBMS \n Python \n DS \n')
  # readline()
    fileobj = open("abc.txt", "r")
      str2 = fileobj.readline()
      Print ("The output of readline method :", str2)
      file obj.close()
    >>> ('The output of readline method :', 'Computer Science
        Subjects \n DBMS \n Python \n DS \n')
      # readline()
      fileobj = open("abc.txt", "r")
        str2 = fileobj.readline()
        Print ("The output of readline method :", str2)
        file obj.close()
      >>> ('The output of readline method :', 'Computer Science Subject \n')
        # readline()
          fileobj = open("abc.txt", "r")
            str3 = fileobj.readline()
            Print ("The output of readline method :", str3)
            file obj.close()
>>>('The output of readline method :', ['Computer Science Subject \n', 'DBMS \n', 'Python \n
                                                                        DS \n'])
  # File attributes
    a = fileobj.name
    Print ("name of file (name attribute):", a)
    >>> ('name of file (name attribute)', 'abc.txt')
      b = fileobj.closed
      Print ("(close) attribute :", b)
      >>> ('(close) attribute :', 'True')
```

```
c = file obj . mode
Print ("file mode"; c)
>>> ('file mode', 'r')
d = file obj . Softspace
Print ("Softspace"; d)
>>> ('Softspace : ', 0)
```

※ W+ mode

```
fileobj = open ("abc.txt", "w+")
file obj . write ("Ashish Sir")
fileobj . close()
```

※ r+ mode

```
file obj = open ("abc.txt", "r+")
s1 = file obj . read (6)
Print ("output of r+"; s1)
file obj . close()
>>> ('output of r+', 'Ashish Sir')
```

※ append mode

```
file obj = open ("abc.txt", "a")
file obj . write ("data Structure")
file obj . close()
file obj = open ("abc.txt", "r")
s3 = file obj . read()
Print ("output of append mode:"; s3)
file obj . close()
>>> ('output of append mode : ', 'Ashish Sir'
    data Structure
```

※ write mode

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

※ read mode

```
fileobj = open ("abc.txt", "r")
s = fileobj . read()
Print ("output of read
 mode !"; s)
>>> ('output of read mode !',
    'Ashish Sir')
```


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

Step 5 :- open fileobj in read mode display the update written contents and close open again in 'r+' mode with parameter passed and display the output subsequently

Step 6 :- Now open file obj in append mode, open write method write contents close the fileobj again open the fileobj in read mode and display the appending output

Step 7: open the fileobj in read mode. declare a variable and Perform fileobject dot tell method and Store the output Consequently in Variable.

Step 8: Use the Seek method with the arguments with opening the file obj in read mode and closing Subsequently

Step 9: open file obj with read mode also use the readlines method and Store the output Consequently in and Print the Same for Counting the length use the for Conclution Statement and display the length


```
# tell()
fileobj = open("abc.txt", "r")
Pos = fileobj.tell()
Print ("tell(): "; Pos)
fileobj.close
>>> ('tell(): ', 0L)
# SeekI()
fileobj = open("abc.txt", "r")
St = fileobj.SeekI(0,0)
Print ("SeekI(0,0) is: ", St)
fileobj.close
>>> ('SeekI(0,0) is: ', None)
fileobj = open("abc.txt", "r")
St = fileobj.SeekI(0,1)
Print ("SeekI(0,1) is: ", St)
fileobj.close()
>>> ('SeekI(0,1) is: ', None)
fileobj = open("abc.txt", "r")
St2 = fileobj.SeekI(0,2)
Print ("SeekI(0,2) is: ", St2)
fileobj.close()
>>>(SeekI(0,2) is: ; None)
# finding length of difference lines exist
within lines
fileobj = open("abc.txt", "r")
Stat = fileobj.readlines()
Print ("output: "Stat)
for line in Stat:
        Print (len(line))
fileobj close()
>>>('output:', ['DBMS'])
1
```

last

Scanned with CamScanner

```
* iter() and next()
mytuple1 = ("banana", "orange", "apple")
myiter1 = iter(mytuple)
Print(next(myiter1))
myiter2 = iter(mytuple)
Print(next(myiter1))
myiter3 = iter(mytuple)
Print(next(myiter1))
>>> banana
    orange
    apple
* for loop
  mytuple1 = ("Kevin", "Stuart", "bob")
    for x in mytuple1:
        Print(x)
  >>> Kevin
      Stuart
      bob
* Square and Cube
    def Square(x):
        Y = x * x
        return Y
    def Cube(x):
        z = x* x** X
        return z
arr1 = [Square, cube]
```


Practical-2

OBJECTIVE : Iterators

Step 1: Create a tuple with elements that we need to iterate using the iter and next method the number of time we use the iter and next method we will get the next iterating element in the tuple. Display the same.

Step 2: The Similar output can be obtained by using for conditional 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 Value to get the Value raised 3. and return the same.

Step 4: Call the declared functions using function call.

Step5: Using for Conditioned Statement Specifying the range use the list type Casting with map method declare a 'lambda' i.e annonymus function and Print the Sums.

Step 6: Declare a list num 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 Statements do check whether the number is even and odd and return respectively

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

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

```
for x in range(5):
    Value = list(map(lambda x: x(x), func1))
    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))
#odd numbers
class odd:
    def __iter__(Self):
        Self.num=1
        return Self
    def __next__(Self):
        num.Self=num
        num.Self+=2
    def __next__(Self):
        num=Self.num
        Self.num+=2
        return num
```

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

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


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.

Output
2/11/22

Practical-3

Topic: Exceptions

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

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

Step 3: Else display the operation is Successful!

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

Step 5: In except block use ValueError and Print the same message.

Step 6: Else display the operation is Successful.

```
#IOError:
try:
    fo = open ("abc.txt", "w")
    fo = Write ("Python is an intendent language.")
except IOError:
    Print("Enter appropriate mode for file
          operation!")
else:
    Print("Operation is Successful!")
>>> operation is Successful!
# Routput:
>>> Enter appropriate mode for file
    operation!.
# 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!
>>> Enter a Statement: 123
    operation is Successful!
```

```
# Type error, Zero division error
    a:
    b = input ("Enter :")
try: Print (a/b)
except Type Error:
    Print ("Incompatible Value")
except Zero Division Error:
    Print ("Denominator is Zero so operation
        Cannot be Performed")
>>> Enter : t
    Incompatible Value
>>> Enter : 2
    0.5
>>> Enter : 0
    Denominator is Zero so operation Cannot
    be Performed.
# multiline exception
a, b: 1, 0
try:
    Print (1/0)
    Print (1/0+10) '10'
except (Type Error, Zero Division Error):
    Print ("Invalid Input!")
>>> 1.0
>>> 10/0
>>> Invalid Input!
```


Step 7 : Accept an integer Value from the user. In the try use the division method.

Step 8 : For the exceptions to be raised use the except keyword (and) i.e Type Error Print Incompatible Values.

Step 9 : Use except with error of Zero division Error, and Print the message awarding that if entered number is Zero not able to Perform operation!

Step 10 : Declare Static Variables and Values

Step 11 : For Multiline exceptions use the error types by Separating them with a Comma.

Step12: Use try block open a file in write mode and subsequently enter values in the file

Step13: Use the IOError and display appropriate message.

Step14: Define a function with empty list and calculate the length of the list

Step15: Define another function Y initialise or declare some elements in list and calculate the length of the same and display the same.

Step16: In try block accept input from the user and if the user enters character values raise an Error that is saying up Enter integer values.


```
# Using except Keyword
try:
    a=open("abc.txt","w")
    a.write("Python")
except IOError:
        Print ("Error")
else:
    Print Successful
def x():
    l=[]
    Print (len(l))
def Y():
    li=[2,4,4,1]
    Print (len(li))
Print (x())
Print (Y())
output: Successful
        0
        None
        None
# raise Keyword:
try:
    a:int (input("Enter a number:"))
        raise Value Error
except Value Error:
    Print ("Enter integer Value!")
>>> Enter: xyz
    Enter integer Values!
```

※ match()

```
import re
Pattern = r"FYCS"
Sequence = "FYCS represents Computer Science Stream"
if re.match (Pattern, Sequence):
    Print ("matched Pattern found!")
else:
    Print ("NOT FOUND")
```

>>> matched Pattern found!

※ numerical Values (Segregation)

```
import re
Pattern = r'\d+'
String = 'hello123, howdy789, 45howru'
output = re.findall (Pattern, String)
Print (output)
```

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

※ split()

```
import re
Pattern = r'\d+'
String = 'hello123, howdy789, 45howru'
output = re.Split (Pattern, String)
Print (output)
```

>>> ['hello', ', howdy', ', ', 'howru']

Practical - 4

Topic: Regular expression

Step 1: Import re module declare Pattern and declare Sequence use match method with declare arguments if arguments matched than Print the Same otherwise Print Pattern NOT FOUND!

Step2: Import re module, declare Pattern with literal and mita 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 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 th group() with dot operator do Search() gives memory location 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 the for 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 number are equal to 10. if Criteria matches Print number matches otherwise Print failed

```
* no-Space:
  import re
  String = 'abc def ghi'
  Pattern = r'\s+'
  replace = ''
  V1 = re.Sub(Pattern, replace, String)
  Print(V1)
>>> abcdefghi

* group()
  import re
  Sequence = 'Python is an interesting language'
  V = re.Search('\APython', Sequence)
  Print(V)
  V1 = V.group()
  Print(V1)
>>> <_Sre.SRE_Match object at 0x0281DF00>
    Python

* Verifying the given Set of Phone numbers
  import re
list1 = ['9145673210', '7865432981', '9145673210',
         '9876543201']
for Value in list1:
      if re.match(r'[8-9]{1}[0-9]{9}',
      Value or len(Value) == 10):
    Print("Criteria matched for Cell number!")
  else:
      Print("Criteria failed!")
```

>>> Criteria matched for Cell number
Criteria matched for Cell number
Criteria failed!
Criteria matched for Cell number

* Vowels

```
import re
Str1='Plant is life overall"
output=re.findall(r"b[aeiouAEIOU]\w+',Str1)
Print(output)
```

>>> ['is', 'overall']

* host & domain

```
import re
Seq='abc.tcsc@edu.com, XYZ@gmail.com'
Pattern=r'[\w\.-]+[\w\.-]'
output=re.findall 'Pattern, Seq)
Print(output)
```

>>> ['abc.tcsc', 'edu.com', 'XYZ', 'gmail.com']

* Counting of first 2 letters:

```
import re
S=|mr.q, ms.b, ms.c, mr.t'
P=r'[/ms/mr/]+
o=re.findall(P,S)
Print(o)
m=0
f=0
for V in O:
```


Step 7: import re module declare a String, use the module with findall() for finding the Vowels in the String and declare the Same.

Step 8: import re module declare the host and domain name declare Pattern for Seprating the host & domain name. use 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 Use findall() declare two Variables with initial Value as Zero use for Condition and Subsequently use the if Condition check whether Condition Satisfy add up the or else increment Value. and display the Values Subsequently.

7/1/22

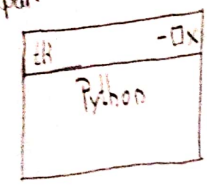
```
if(V=='ms'):
    f=f+1
else:
    m=m+1
Print ("No. of males is :", m)
Print ("No. of females is :", f)
```

```
>>>['mr', 'ms', 'ms', 'mr']
('No. of males is :', 2)
('No. of females is :', 2)
```

Creation of Parent Window

```
from TKinter import *
root = TK()
l = Label(root, text="Python")
l.Pack()
root.mainloop()
```

output:



#2

```
from TKinter import *
root TK()
l = Label(root, text="Python")
l.Pack()
l1 = Label(root, text="CS", bg="gray",
    fg="black", font="10")
l1.Pack(side=LEFT, Padx=20)
l2 = Label(root, text="CS", bg="light blue",
    fg="black", font="20")
l2.Pack(side=LEFT, Pady="30")
l3 = Label(root, text="CS1", bg=Yellow",
    fg="black", font="10")
l3.Pack(side=TOP, iPadx=40)
```


Practical-5

Topic: Gui components

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

Step 2: Create an object using the TK().

Step 3: Create a Variable using the Widget label and use the text method.

Step 4: Use the mainloop() for triggering of the Corresponding above mention events

#2:

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

Step 2: Create a Variable from the text method and Position it on the Parent Window.

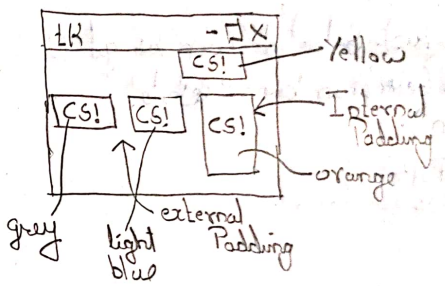
Step 3: Use the Pack() along with the object created from the text() and use the Parameter
1) Side = LEFT, Padx = 20
2) Side = LEFT, Pady = 30
3) Side = TOP, ipadx = 40
4) Side = TOP, ipady = 50

Step 4: Use the main loop() 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 then 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 :



#1:

```
# Radio button
    from tkinter import *
    root = TK()
root.geometry("500 x 500")
def Select():
    Selection = "You just Selected" + Str(Var.get())
    t1 = Label(Text = Selection, big = "Verbuto",
        fg = "green")
    t1.Pack(Side = Top)
Var = StringVar()
l1 = Listbox()
l1.insert(1, "List 1")
l2.insert(2, "List 2")
l1.insert(anchor = N)
t1 = Radio Button(root, text: "option 1", Variable
    = Var, Value = "option 1", Command = Select)
t1.Pack(anchor = N)
t2 = Radio Button (root, text: "option 2", Variable =
Var, Value = "option 2", Command = Select)
t2 = Pack(anchor = N)
root.mainloop()
```


Practical - 5 (B)

Aim : GUI Components

Step 1 : Import the relevant method 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 the user about the given Selection made from multiple option available

Step 4 : Now define the Parentwindow and define the option with Control variable.

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 Parentwindow object, text Variable which will take the Values option no 1, 2, 3.. Variable argument, Corresponding Value &

Step 6: trigger the function declared

Step 7: Now call the Pack() for radio object so treated and specify the argument using anchor attribute

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

#2:

Step1: Import relevant methods from the tkinter library

Step2: Create a Parent object corresponding to the Parent window

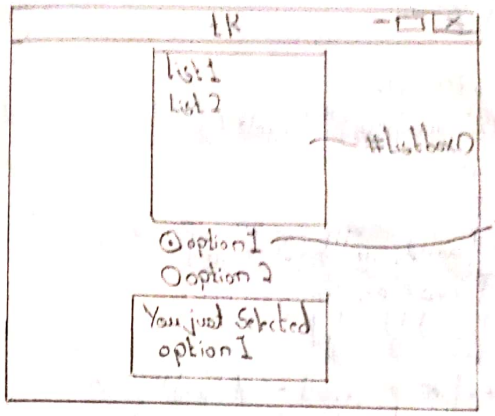
Step3: Use the geometry() for laying of the window

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

Step 5: Use the Pack() along with the Scrollbar object with Side and fill attribute

Step 6: Use the mainloop with the Parent object

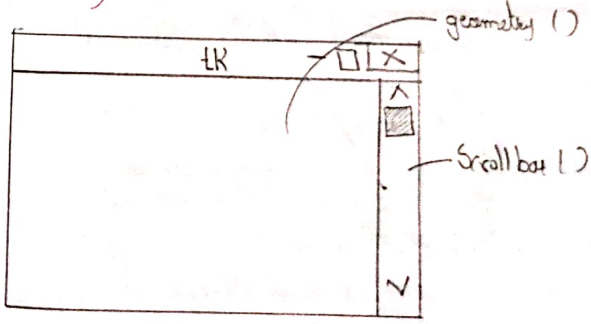
output



#2:

```
Scroll bar()
from tkinter import *
root = Tk()
root.geometry("500x500")
S = Scrollbar()
S.Pack(Side="right", fill="y")
root.mainloop()
```

output

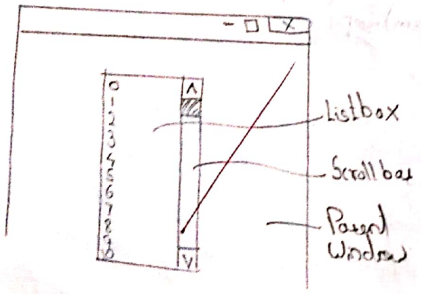


3:

#Using frame widget

```
from tkinter import *
window=Tk()
window.geometry("680x500")
Label(Window, text="numbers:").Pack()
frame=Frame(window)
frame.Pack()
list nodes=Listbox(frame, width=20, height=20
        font=("Times New Roman", 10))
list nodes.Pack(side="left", fill='y')
Scrollbar=Scrollbar(frame, orient="Vertical")
Scrollbar.config(command=listnodes.Yview)
Scrollbar.Pack.config(side="right", fill="y")
for x in range(100):
    listNodes.insert(END, str(x))
Window.mainloop()
```

output:



#3:

Step1: Import the relevant libraries from the tkinter method

Step2: Create an Corresponding object of the Parent Window

Step3: Use the geometry manager with Pixel Size (680 x 500) or any other Suitable Pixel Value

Step 4: Use the label widget along with the Parent object Created and Subsequently Use the Pack() method.

Step 5: Use the frame widget along with the Parent object Created and Use the Pack() method.

Step 6: Use the listbox method along with the attributes like width, height font Do Create a listbox method's object use Pack() for the same.

Step 7: Use the Scrollbar() with an object use the attribute of Vertical. then Configure the same with object Created from the Scrollbar() and use Pack()

Step 8: Trigger the event using mainloop.

#4:

Step1: Import relevant methods from tkinter library

Step 2: Define the object corresponding the 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 the Parent Window

Step 4: Create another frame object termed as the left frame and Put it on the Parent Window on its LEFT side.

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

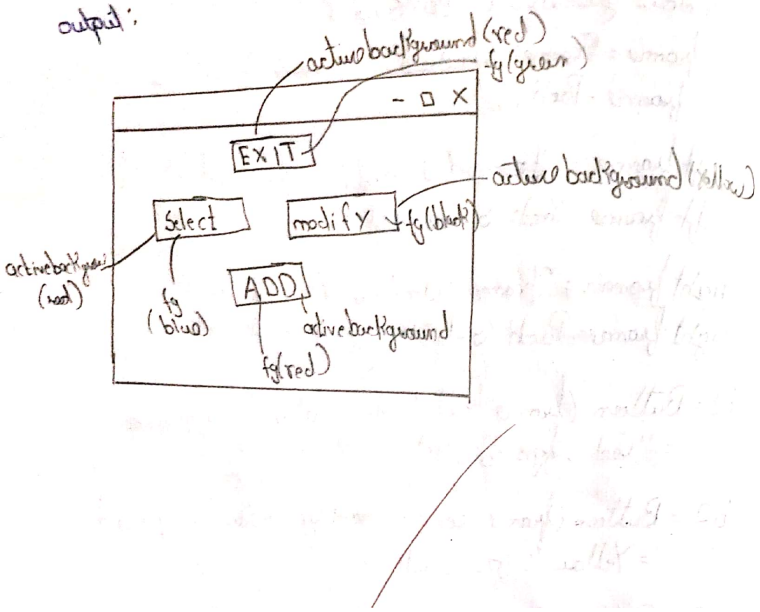
Step 6: Now use the Pack() along with the side attribute

Step 7: Similarly Create the button object Corresponding to the MODIFY operation Put it into frame object on side = "right"

#4:

```
from tkinter import *
Window = Tk()
Window.geometry("680x500")
    frame = Frame(Window)
    frame.Pack()

    left frame = Frame(Window)
    left frame.Pack(side = "left")

    right frame = Frame(Window)
    right frame.Pack(side = "right")

b1 = Button(frame, text = "Select", active background
    = "red", fg = "blue")

b2 = Button(frame, text = "modify", active background
    = "yellow", fg = "black")

b3 = Button(frame, text - "ADD", active background
    = blue", fg = "red")

b4 = Button(frame, bat = "EXIT", active background
    = "red", fg = "green")
b1.Pack(side = "LEFT", Padx = 20)
b2. Pack(side = "right", Padx = 30)
b3. Pack(side = "bottom", Pady = 20)
b4. Pack(side = "top")
```

output:



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

Step 9: Add another button & put is on the top of frame and label it as EXIT.

Step 10: use the Pack() simultaneously for all the objects & finally use the mainloop().

23/12

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 tk messagebox with Showinfo method along with info window attribute

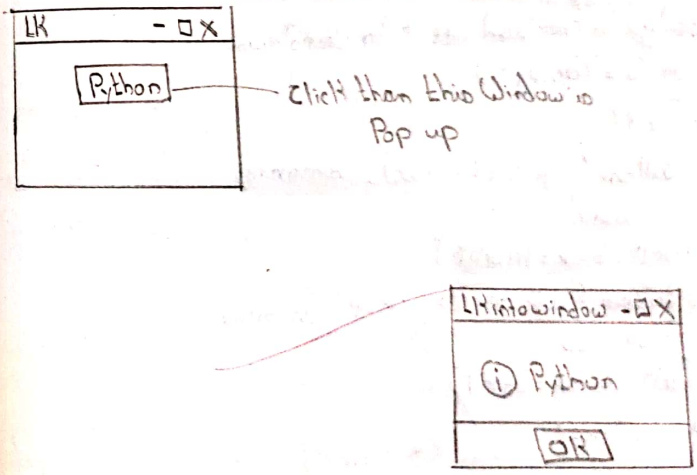
Step 5: Declare a button with Parent Window object along with the Command attribute

Step 6: Place the button widget onto the Parent Window and finally call mainloop() for triggering of the events called above

Message box

```
from Tkinter import *
import tkMessage box
root = Tk()
def function():
    tk Message box.Showinfo("info Window",
        Python")
b1 = Button(root, text = "Python", Command
        = function)
b1.Pack()
root.mainloop()
```


output:



```
#Multiple Window
#Different button(Relief())
from Tkinter import *
root = TK()
root.minsize (300,300)
def main ():
    top = tK()
    top.config (bg = "black")
    top.title ("HOME")
    top.minsize (300,300)
L=Label (top, text = "SAN FRANCISCO
 inplace of interest In Golden Gate
 Bridge In Lombard Street In Chinatown
 In Coit Tower")
L.Pack()
b1 = Button (top, text: "next", command
    = Second)
b1.Pack (Side = RIGHT)
b2 = Button (top, text = "exit", command
    = terminate)
b2.Pack (Side = LEFT)
top.mainloop()
```


Step 1: Import the relevant methods from the tKinter library along with Parent Window object declared.

Step 2: Use Parent window 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 we as button() and use Pack() & mainloop() Simultaneously

Step 4: Similarly define the function Second and use the attributes accordingly.

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

Step 6: Finally Called the mainloop() for event driven Programming

```
def Second():
    top2=TK()
    top2.config(bg="orange")
    top2.title("About US!")
    top2.minsize(300,300)
L=Label(top2,text="Created by: Nehal Tanveer In
For more details Contact to our official account")
L.Pack()
b3.=Button(top2,text='Prev',command=main)
b3.Pack(side=LEFT)
b2.=Button(top2,text="exit",command=
    terminate)
b2.Pack(side=RIGHT)
top2.mainloop()
def button1():
    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 = "subject button",
relief = RIDGE)
top3.mainloop()
def terminate():
    b5.quit()
b5 = Button (root, text = "TOUR DETAILS",
    Command = main)
b5.Pack()
b6 = Button (root, text = "BUTTON DETAILS",
    Command = button
b6.Pack()
root.mainloop()
```


Practical - 5(D)

Aim : Gui Components

Step 1: Import relevant methods from the tkinter library

Step 2 : Create parent window object and use the Config method along with background Colour attribute Specified

Step 3 : define a function finish with the message box widget which will display a message i.e a warning message and Subsequently terminate the Program.

Step 4 : define a function who use a listbox widget along with the object of the Same . use the listbox object along with insert method and insert the Same and finally use the grid() with ipadx attribute

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

Step6: Use PhotoImage widget with file and filename with gif attribute.

Step7: Create a frame object along with the frame() along with Parent window object height & width specified and subsequently use the grid() with row & column attribute specified

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

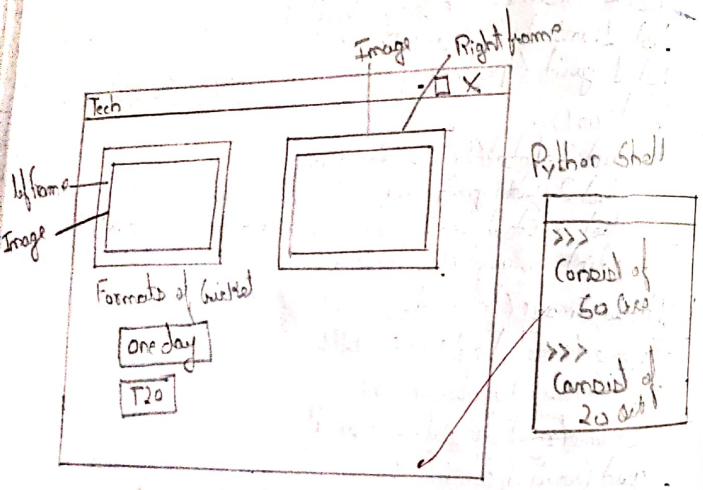
Step 9: Create another object & use the Sub sample (5,4)

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.


```
from TKinter import *
root = TK()
root = Config(bg="grey")
def finish():
    messagebox.askokcancel("Warning", "This will
    end the 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 theatre March 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=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=13)
l2 = Label(f2, image=P1, relief=SUNKEN)
l2.grid(Padx=25, Pady='a')
b1 = Button(f1, text, = "Informs action", relief=SUNKEN,
    command=info)
b1.grid(row=1, column=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(E)

Aim :- GUI Components :- Spinbox, Paned Window and Canvas

1) Spinbox :-

Step 1 :- Import relevant methods from the tkinter library.

Step 2 :- Create parent window object along TK()

Step 3 :- Create an object from Spinbox method and use attributes Parent Window object, from and to attributes

Step 4 :- Subsequently all the pack() method along with Spinbox object and Called the mainloop

2) Paned Window :-

Step 1 :- Import relevant methods from tkinter library also Create a Parent window object.

Step 2 :- Create an object along with Paned Window and Subsequently use the Pack method along with the Paned window object along with attributes fill and expand.

Step3:- Create a label object along with label method used Paned window object, orient and background Colour attributes

Step4:- Similarly create another label method use Paned Window object, orient.

Step5:- Finally use the mainloop method

3] Canvas:-

Step 1:- Import relevant methods from tkinter library declare a Parent window object.

Step 2:- Create a Canvas object along with the Canvas method with attributes such as height, width, background Colour

Step 3:- Use Create-arc() along with Canvas object declared along with start and extend Coordinates.

Step 4:- Similarly for oval and line. Use the Pack method & finally call the mainloop method.

1) from tkinter import *
master = Tk()
S = Spinbox(master, from_ = 0, to 20)
S.Pack()
master.mainloop()

2) from tkinter import *
Window = Tk()
P = Paned Window()
P.Pack(fill = BOTH, expand = 20)
l = label(P, text = "Paned Window")
P.add(L)
P1 = Paned Window(P, orient = VERTICAL, bg = "black")
P.add(P1)
L = Label(P1, text = "Sash movement")
P1.add(L)
Window.mainloop()

3) from tkinter import *
Canvas_obj = TK()
C = Canvas(root, height = 100, width = 200, bg = "orange")
Cir1 = (Create_arc (10, 20, 30, 40, Start = 10, extent = 50,
fill = "black"))
Canvas1 = (.Create_oval (20, 31, 45, 60, fill = "red"))
(line = (.Create_line (10, 15, 30, 20, fill = "blue"))
(.Pack ())
Canvas_obj.mainloop ()

Practical - 6

Aim: Database Connectivity

#1:

Step1: Import the (DBM) dbm library & use the open() for creating the database by specifying the name of the database along with the corresponding flag.

Step2: Use the object so created for accessing the given website & corresponding regular name for the website

Step3: check whether the given ur address matches with the regular name of the page is not equal to name then display the message that particular found / match or else not found / unmatched

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

#2 Step 1: Import corresponding library to make database Connection, OS & SQL Like-3

Step 2: Now Create the Connection object using SQL Like-3 Library & the Connect() for Creating new database

Step 3: Now Create cursor object using the Cursor() & from the Connection object Created

Step 4: Now use the execute() for Creating the table with the Column name & respective datatype.

Step 5: Now with Cursor object use the insert Statement for intering the Values Corresponding to different fields, Corresponding the datatype.

Step 6: Use the commit() to Complete the transaction using the Connection object

Step 7: Use the execute Statement along with Cursor object for accessing the Values from the database using the Select from where clause

1:

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

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

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

#2: import os, Sqlite 3
Conn: Sqlite 3. Connect ("employee.db")
Cur: Conn. Cursor()
Cur.execute ('Create table dos (Name char, RollNo int')
Cur.execute ("insert into dos Values ('Nehal', 1737), ('Aryan', '1723')")
Conn.Commit()
Cur.execute ('Select * from dos')
Print ((Cur.fetchall()))
Con.close()

output:

[('Nehal', 1737), ('Aryan', 1723')]

```
from tkinter import *
import sqlite3
from tkinter import messagebox

root = Tk()
root.geometry('500x500')
root.title("Registration Form")


Fullname=StringVar()
Email=StringVar()
var = IntVar()
c=StringVar()
var1= IntVar()



def database():
  messagebox.showinfo("success.","your details are succesfully entered")
  name1=Fullname.get()
  email=Email.get()
  gender=var.get()
  country=c.get()
  prog=var1.get()
  conn = sqlite3.connect('Form.db')
  with conn:
    cursor=conn.cursor()
  cursor.execute('CREATE TABLE IF NOT EXISTS Student (Fullname TEXT,Email
TEXT,Gender TEXT,country TEXT,Programming TEXT)')
  cursor.execute('INSERT INTO Student (FullName,Email,Gender,country,Programming)
VALUES(?,?,?,?,?)',(name1,email,gender,country,prog,))
  conn.commit()



label_0 = Label(root, text="Registration form",width=20,font=("bold", 20))
label_0.place(x=90,y=53)


label_1 = Label(root, text="FullName",width=20,font=("bold", 10))
label_1.place(x=80,y=130)

entry_1 = Entry(root,textvar=Fullname)
entry_1.place(x=240,y=130)

label_2 = Label(root, text="Email",width=20,font=("bold", 10))
label_2.place(x=68,y=180)

entry_2 = Entry(root,textvar=Email)
entry_2.place(x=240,y=180)

label_3 = Label(root, text="Gender",width=20,font=("bold", 10))
label_3.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)

label_4 = Label(root, text="country",width=20,font=("bold", 10))
label_4.place(x=70,y=280)

list1 = ['Canada','India','UK','Nepal','Iceland','South Africa'];

droplist=OptionMenu(root,c, *list1)
droplist.config(width=15)
c.set('select your country')
droplist.place(x=240,y=280)

label_4 = Label(root, text="Programming",width=20,font=("bold", 10))
label_4.place(x=85,y=330)
var2= IntVar()
Checkbutton(root, text="java", variable=var1).place(x=235,y=330)

Checkbutton(root, text="python", variable=var2).place(x=290,y=330)

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

root.mainloop()
```

```
from tkinter import *

root = Tk()
root.geometry('500x500')
root.title("Registration Form")

label_0 = Label(root, text="Registration form",width=20,font=("bold", 20))
label_0.place(x=90,y=53)


label_1 = Label(root, text="FullName",width=20,font=("bold", 10))
label_1.place(x=80,y=130)

entry_1 = Entry(root)
entry_1.place(x=240,y=130)

label_2 = Label(root, text="Email",width=20,font=("bold", 10))
label_2.place(x=68,y=180)

entry_2 = Entry(root)
entry_2.place(x=240,y=180)

label_3 = Label(root, text="Gender",width=20,font=("bold", 10))
label_3.place(x=70,y=230)
var = IntVar()
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)

label_4 = Label(root, text="country",width=20,font=("bold", 10))
label_4.place(x=70,y=280)

list1 = ['Canada','India','UK','Nepal','Iceland','South Africa'];
c=StringVar()
droplist=OptionMenu(root,c, *list1)
droplist.config(width=15)
c.set('select your country')
droplist.place(x=240,y=280)

label_4 = Label(root, text="Programming",width=20,font=("bold", 10))
label_4.place(x=85,y=330)
var1 = IntVar()
Checkbutton(root, text="java", variable=var1).place(x=235,y=330)
var2 = IntVar()
Checkbutton(root, text="python", variable=var2).place(x=290,y=330)

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

root.mainloop()
```