Practical 01

- Aim:- Demonstrate the use of different accessing mode, different attribute and read method.

Step I:- Create a file, object using open method and use the write accessing method followed up by writting some contents onto the file and then closing the file.

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

Step III:- Now use the file object for finding the name of file, the file mode in which it is opened whether the file is still open or close and finally the output of the program attribute.

Step IV:- Open file object in read mode, display the update open in rt mode with parameter & display the attr

Step V:- Now open file object in append mode and use write method in read mode to display append output

Step VI:- Use seek method with the apparent open file object in read mode

Step VII:- Use the for conditional statement.

```
file_obj=open("abc.txt","w") #file open(write mode)
file_obj=write("Computer Sci subject"+"/n")
file_obj=write("DBMS/n Python/n DS&/n")
file_obj:close()
```

⇒ The object of read method Computer Sci subject
DBMS
Python
DS

```
#readlines
file_obj=open("abc.txt","r")
str3=file_obj.readlines()
print("The object readlines method",str3)
file_obj.close()
```

⇒ The output of readline method : Computer Science subject
DBMS
Python
DS

file attributes

```
a. fileobj.name
print ("Name of file (name attribute): ", a)
>>> Name of file (name attribute): abc
b= fileobj closed
print (closed) attribute", b)
>>> (closed) attribute : True

c = file obj.mode
print ("file mode", e)
>>> file mode e

d : file obj softspace, d
print ("softspace", d)
>>> softspace 0
```

r+ mode

```
file= open ("abc.txt", r+)
str1= obj read(r)
print ("Output of str1", str1)
file.obj.close ()
```

>>> Output of str1 'SIMIT'

read mode

```
fileobj=open ("abc.txt", r)
str2 = fileobj .read()
print ("Output of read:
       ", str2)
```

>>> Output of read mode:
SIMIT

Append Mode:-

```
file obj.open ("abc.txt", "a")
fileobj .write ("Data Structure")
fileobj.close ()
fileobject = open ("abc.txt", "r")
str3 = fileobj .read ()
print ("Output of append mode", str3)
file object.close ()
```

>>> Output of append mode : SIMIT, Data Structure.

Seek ()

```
. fileobj = open ("abc.txt", "r")
str 4 = fileobj.seek(0, 0)
str 8 = fileobj .seek (0, 1)
print ("The beginning of the line is :", str 8)
```

```
class odd:
    def __iter__(self):
        self.num = 8
        return self
    def next(self):
        if (self.num <=10):
            num = self.num
            self.num += 2
            return num
        else:
            raise StopIteration
```

```
>>> 4 = next()
>>> 2 = iter()
>>> 2 = next()
>>> 1/2 . next()
>>> 2/5 . next()
>>> 2/7 . next()
>>> 2/11 next()
```

Practical 22

1) Demonstrate the use of iteration

Theory: In python, iterate is an object which implements iterator class which has 2 methods namely --iter()-- and --next()-- type, set of the set represent the iterable object.

1) Write a program using iterable object for displaying the odd number in range 1 to 10.

Algorithm:

Step I: Define a iter() with arguement and initial the value & return the value.

Step II: Define the next() with an arguement and compare the upper limit by using a conditioned statement.

Step III: Now create an object of the given class and pass the object in the iter method.

=

(Q2) Write a program using an iter for calculating the correct powers of a given no. for by iter method that value calculated should be 1, 2, 2², 2³, 2⁴

>>>

Algorithm:

>>>

Step 01: Define iter() method with arguement initialize and return the value.

>>> Step 02: Now define next() with an arguement and compare upper limit with an arguement.

Step 3: Now create object of the given class and pass the object to the iter method

>>>

>>>


```
class power:
def __iter__(self):
    self=0
    return self
def next(self):
    if (self.p<=10):
        num = self.p
        self.p += 1
        p = 2**num
        print("2**.self.p-1 = ", p)
        return p
    else:
        raise StopIteration
```

```
>>> p >> processor()
>>> x = iter()
>>> next()
    2**0 = 1
>>> n.next()
    2**3 = 8
```

05

```
class fact:
    def __iter(self):
        self.f = 1
        return self
    def next(self):
        if (self.f <= 10):
            num = self.f
            self.f += 1
            fac = 1
            for i in range(1, num+1):
                fact = fact * i
            print(self.f - 1, "! =", fact)
        else:
            raise StopIteration
```

>>>
>>>
>>>
>>>
>>>
>>>

1) Write a program for factorial of number in 1 to 10 range

Algorithms:

: Define an iter() method with argument, initialize the value and return the value

: The __iter__() method with an argument and compare the upper limit using a conditional statement

: Now create an object of the other class and pass it as an argument in the iter() method.

Practical No. 3

Aim: Demonstrating the use of exception handling

Theory: An exception is an event which during execution of a program which changes the normal flow of the program. An exception must be handled immediately, it terminates the program.

→ Algorithm:

Step 01: Define a function that accepts the age of the student through standard input.

Step 02: Use the 'try except' conditional statement to check and raise valueerror if necessary

Step 03: Use the while loop to check boolean expression if it holds true or not, run the loop until proper input is given


```
def acc_age():
    age = int(input("Enter your age:"))
    if (age > 30 or age < 16):
        raise ValueError
    else:
        print("Your age is", age)
valid = False
while not valid:
    try:
        age = acc_age()
        valid = True
    except ValueError
        print("Your age is not in range")
```

Output:

```
>>> Enter your age: 15
    Your age is not in range
>>> Enter your age: 18
    Your age is 18
```

```
def divide(a,b):
    ans=a/b
    return ans
while True:
    try:
        a=int(input("Enter the 1st no."))
        b=int(input("Enter the 2nd no."))
        ans=divide(a,b)
        print(ans)
        break
    exception zero divide error:
        print("Error!")
```

Output:

```
>>>Enter the ~~first~~ 1st no. 1
>>>Enter the 2nd no. 1
    1

>>>Enter the 1st no. 1
>>>Enter the 2nd no. 0
    Error!
```

Demonstrating the use of zero division error:

Algorithms:

Step I:- Use try block and accept the input using input().

Step II:- Define a function with two parameters to divid the number entered by the user.

Step III:- Define while loop to the boolean expression hold true

Step IV:- The except with zero division error, the demonstration in zero

Practical 04:

Aim: Demonstrating the use regular expressions.

Theory: Regula expression represent the sequence of character which in mainly used for finding & replacing the given pattern in the pattern.

Algorithm:

Step I: Applying a pre-integrated string and pattern in and display the output.

Step II: It is used to match all the decimal digit ~~where it is used to~~

```
import re
S = "hello 1234 abc 567"
r = re.findall("\d+", s)
r1 = re.findall("\d+", s)
print(r)
print(r1)
```

Output:

```
>>> ['1234', '567']
>>> ['hello', 'abc']
```

```
import re
s = "Python is an important language"
r = re.search ("\nPython", s)
print(r)
if (r):
    print ("Match found")
else:
    print ("Match not found")
```

Output

```
>>>re match object .span(0,6)
          match "Python">
>>> Match found
```

```
import
string "Python is important"
r1 = re.findall ("^\w+", string)
r2 = re.findall ("+\w+", string)
print (r1)
print (r2)
```

Output: >>>Python

>>> important

) Find matching string at the beginning of the sequence.

Algorithm:

Import 're' module and apply a string

: Use search() with /A extention and string as parameter to search for a parameter string.

: Now display the output.

: Now, use the conditional statement for to check whether the string has found the word or not..

) Extracting words with and without , from a sentence:

: Import 're' module and apply a string.

: Use findall() to extract a word from a given string

: use ^ & + to extract the first word along with space and extract and without space last word

: Now display the output

3) Extract username, hostname and both.

I: Import re module and apply for string

II: Use findall() to extract username, hostname and both

III: Display the output.

```
import re
string = "abc tcsc.edu"
str1 = re.findall("\w", string)
str2 = re.findall("\w", string)
str3 = re.findall("\w", string)
print(str2)
print(str3)
print(str3)
```

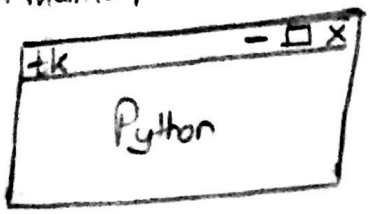
Output >>> [abc]

>>> [tcsc.edu]

>>> ['abc' 'tcsc.edu']

#Creation of Parent Window

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



#2:

```
from Tkinter import *
root = Tk()
l = Label (root, text = "Python")
l.pack()
l1 = Label (root, text = "CS!", bg = 'grey', fg = 'black',
            font = '10')
l1.pack(side=LEFT, padx=20)
l2 = Label(root, text = "CS!", bg = 'light blue', fg = 'black'
           font = 20)
l2.pack (side = LEFT, pady = 30)
l3 = Label(root, text = "CS!", bg = 'yellow', fg = 'black'
           font = 10)
l3.pack(side = TOP, ipadx = 40)
l4 = Label (root, text = "CS!", bg = 'orange', fg = 'black'
           font = 10)
```

Practical - 5

Topic: GUI components

1. Use the tkinter library for importing the features of the text widget
2. Create an object using the Tk() method.
3. Create a object usin the label widget and then use the text attribute.
4. Use the mainloop() for triggering of the corresponding above mention events

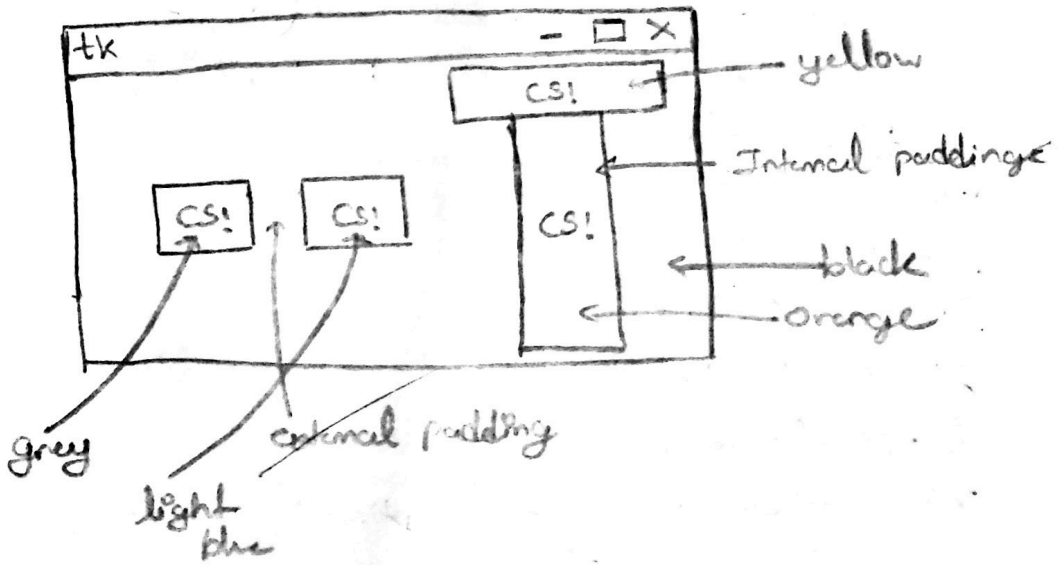
1. Use the tkinter library for importing the features of the text widget.
2. Create a object from the text method and position it on the parent window
3. Use the pack() method along with the object created from the text() method and the attributes of pack method to place the object on the parent window
4. Use the mainloop() method for triggering of the corresponding events

Step V: Now repeat the above steps with 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 attribute

```
l4.pack(side = TOP, ipady = 50)
root.mainloop()
```

→ Output:



Practical 5(b)

#1: Radio Button

```
from tkinter import *
root = Tk()
root.geometry(500x500)
def select():
    selection = "You just selected" + str(var.get())
    t1 = Label(text = "Selection", bg = "white", fg = "
    t1.pack(side = TOP)
var = stringvar()
l1 = Listbox()
l1.insert (1, "List 1")
l1.insert (2, "List 2")
l1.pack(anchor = N)
r1 = RadioButton(root, text = "Option 1", variable = va
    value = "Option 1", command = select)
r1.pack(anchor = N)
r2 = RadioButton(root, text = "Option 2", variable = va
    value = "Option 2", command = select)
r2.pack(anchor = N)
root.mainloop()
```

→ Dim:- GUI components

Ex:-

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

II: Use the parent window object with the geometry method declaring specific dimensions of the parent window.

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

IV: Now define the parent window and define the option with control variable.

V: Use the listbox() and insert options on the parent window along with the pack() method specifying its anchor attribute

VI: Create an object from the radiobutton which will take following arguements for the parent window object, text method will take the values option 1, 2, 3, ..., variable argument, corresponding values & trigger the function declared.

VII: Now use the pack() method to display the widget and specify its argument using anchor attribute

#2

Step I: Import relevant method from the tkinter library

Step II: Create a parent object corresponding to the parent window.

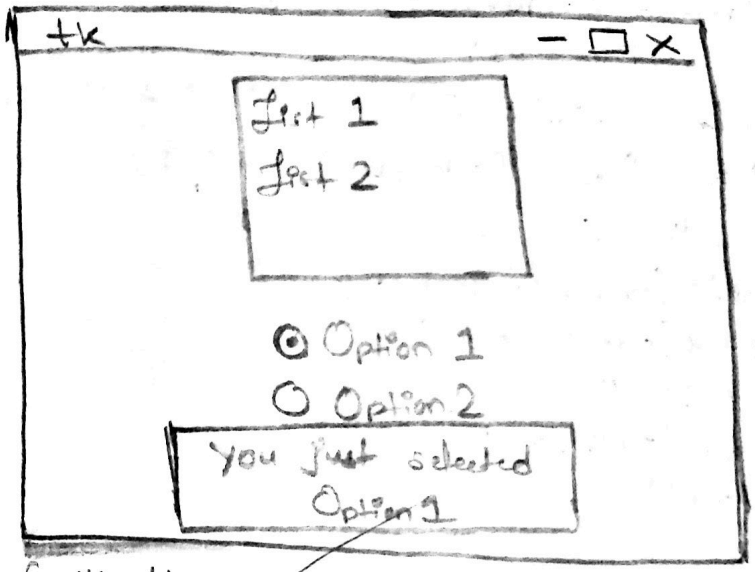
Step III: Use the geometry() method for laying the window

Step IV: Create an object using the scrollbar() method

Step V: Use the pack method, along with the object with its side and fill attribute

Step VI: Use the mainloop() method with the parent window to trigger the whole event

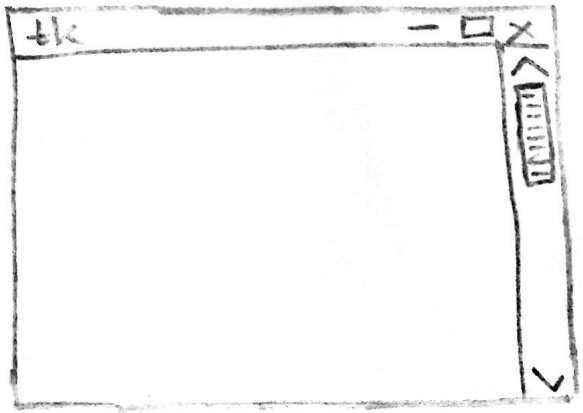
Output: #1



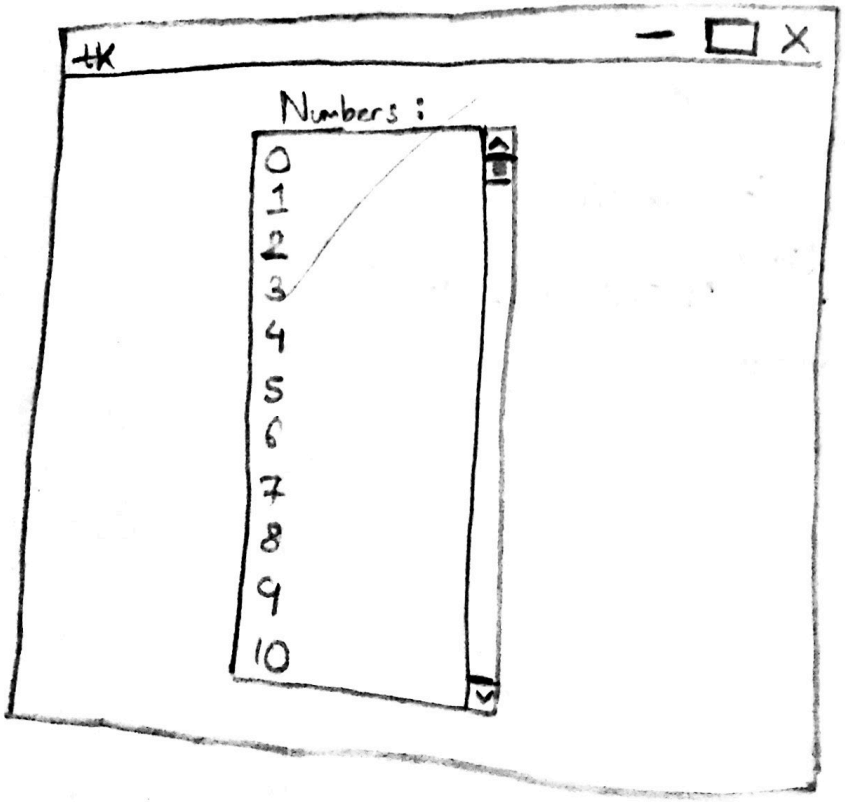
#2: Scrollbar()

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

Output:



```
from tkinter import *
root = Tk()
root.geometry(680x500)
l1 = Label(root, text = "Numbers:").pack()
frame = Frame(root).pack()
l = Listbox(frame, width = 20, height = 20, font = "Times New
        Roman").pack(side = LEFT, fill = Y)
scrollbar = Scrollbar(frame, orient = "Vertical")
scrollbar.config(command = l.yview)
scrollbar.pack(side = RIGHT, fill = Y)
for x in range (100)
    l.insert (END, str(x))
root.mainloop()
```



I. Import the relevant libraries from the tkinter method.

II. Create an corresponding object of parent window

III. Use the geometry method to specify the dimensions

IV. Use the label widget along with the parent window object created and subsequently use the pack() method.

V. Use the frame widget along with a parent window object and subsequently specifying the pack() method.

VI. Use the listbox method along with the attributes like width, height, font. Do create a listbox methods Object use pack() for the same

VII. Use the scrollbar() method with an object also use the attribute of vertical alignment and use pack() method to display it.

VIII. Use the mainloop() method to display the and trigger the corresponding event.

#4

Step I:- Import relevant method from tkinter library

Step II:- Define the object corresponding to parent window and define the size of parent window in terms of dimensions

Step III:- Now define the frame object from the method and place it on to the parent window.

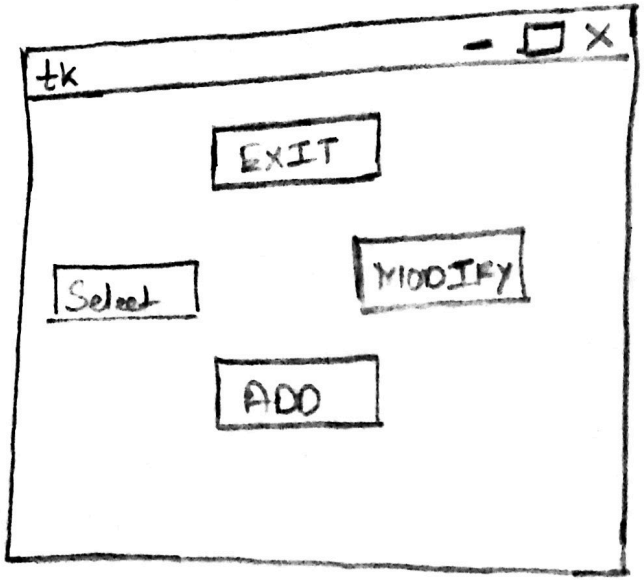
Step IV:- Create another frame object termed as the left frame and but it on the parent window on its LEFT side

Step V:- Similarly define the RIGHT frame and subsequently define the button object placed on the given frame with its attribute as text, activebg, fg.

Step VI Now use the pack() method along with its side attribute

Step VII:- Similarly create the button object corresponding to the MODIFY operation put it onto the frame object on the side = RIGHT

Step VIII Create another button object & place it on the RIGHT frame and label the button as ADD



Add another button object & put it on the top of the frame and label it as EXIT

Use the pack() method for each of the object created and use the mainloop() method.

Practical 75(c)

Aim:- GUI Components

#1

Step I: Import the relevant method from tkinter library

Step II: Import tkMessagebox

Step III: Define a parent window object along with the parent window

Step IV: Define a function which will use tkmessagebox with showinfo() method along with info window attribute

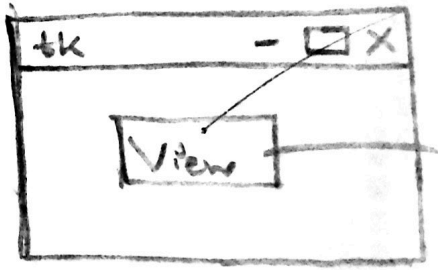
Step V: Declare a button with parent window object along with the command attribute.

Step VI: Place the button widget onto the parent window and finally call the mainloop() for triggering of the events declared above

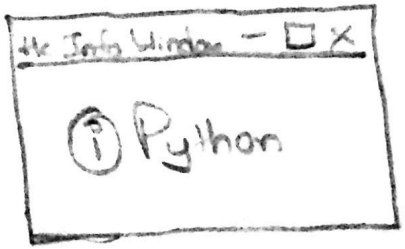
#Messagebox:-

```
from tkinter import *
import tkMessageBox
root = Tk()
def function():
    tkMessageBox.showinfo("Info Window", "Python")
    b1 = Button(root, text="View", command=function)
    b1.pack()
root.mainloop()
```

→ **Output:-**



On clicking the button the following window will pop-up!



→ Multiple Window:-

Step I: Import the relevant methods from the tkinter library along with parent window object declared.

Step II: Use parentwindow object along with minsize function for window size

Step III: Define a function main, with its parent window object and use config(), title(), minsize(), Label() and Button() method with use of pack() for each object created and finally declaring the mainloop() method

Step IV: Similarly for the second window define the function and the use the attributes accordingly.

Step V: Declare another function button along with parent object and declare button with - attributes like RAISED, FLAT, RIDGE, GROOVE, RAISED or SUNKEN along with relief of method.

Step VI: Finally call the mainloop() for the main parent window and for the whole program to be driven.

Practical 85(d)

Aim: Demonstrating use of Spinbox, Paned Window and Canvas widget

1) Algorithm: (Paned Window)

1) A paned window is container widget that may contain any no. of panes, arranged horizontally or vertically.

2) Each pane contains widget and pair of panes seprated by a movable sash.

3) Moving a sash caves movements on widgets on other side of the sash

4) Syntax of paned window is
w = PanedWindow(master, option, ...)

5) Most commonly used attribute for paned window are:-
bg
borderwidth
cursor
orient
relief
sashcursor
sashrelief
sashwidth
sashhandle
width

6) Paned window can have these methods
add (child, option, ...)
get (startindex, [Lend Index])
config (option, ...)

II) Spinbox Widget:

1) The Spinbox Widget is a variant of standard Tkinter Entry widget can be used to select optin from a fixed no. of values

2) This widget can be used instead of an ordinary entry widget, because in this a user only has a limited no. of values to choose from

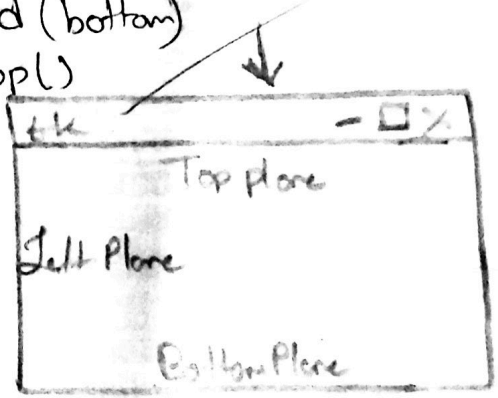
3) The main difference between Spinbox & entry is that one can specify what values to display either using a range method or a tuple.

III) Canvas Widget:

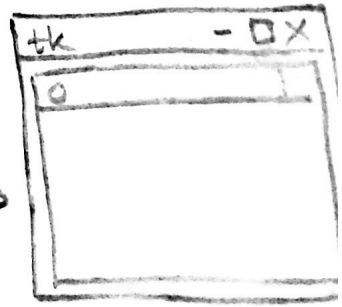
The Canvas widget provides graphic facilities for the user :-
Among those graphical objects are lines, circles, image and even other widgets. With canvas widget one can draw shapes, graphs & plots, create graphic editors & implement various kinds of custom widgets.

1) Paned Window:

```
from tkinter import*
m1 = PanedWindow()
m1.pack(fill = BOTH, expand = 1)
l = Label(m1, text = "Left Plane")
m1.add(l)
m2 = PanedWindow(m1, text orient = VERTICAL)
m1.add(m2)
top = Label(m2, text = "Top plane")
m2.add(top)
bottom = Label(m2, text = "Bottom plane")
m2.add(bottom)
mainloop()
```

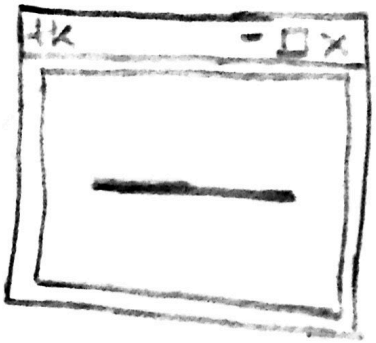


2)
```
from tkinter import*
root = Tk()
w = Spinbox(root, from = 0, to = 10)
w.pack()
root.mainloop()
```



#3:

```
from tkinter import*
root = Tk()
canvas_width = 80
canvas_height = 40
w = Canvas(root, width = canvas_width, height =
                    canvas_height)
w.pack()
y = int(canvas_height/2)
w.create_line(0, y, canvas_width, y, fill = "#47
root.mainloop()
```



→ A Program to draw lines acts as a canvas

→ The method 'create line (words, options)' is used to draw a straight lin

→ ~~After these coordinates~~ The method "words" are given as 4 integer no.s. x1, x2, y1, y2. This means that line goes from point (x1, y1) to (x2, y2)

→ After these coordinates follow as comma, seprated by list of additional parameter which may be empty

→ We can set the color of the line by using the attribute ~~fill = "47 042"~~ fill = #476042

→ We can create a canvas and draw a straight horizontal line on that.

→ The casting to an integer value in the assignment "~~y = canvas~~ y = int(canvas_height/2)" because the create_line can work with float values as well.

Practical 9

→ Aim:- Database Connectivity (using Sqlite3)

→ Algorithm:

1) Import the relevant libraries for the database connect and the operating system functionality.

2) Create an object for making connection to the g database

3) Further create an object corresponding to the cursor area for execution of different query subject

4) Use the cursor object so created for implementing the structure of the database and value with the database

5) Use the execute method for implementation of sele Clause for filtering the information.

6) Now use the fetchall() method along with the cursor object for displaying the values onto the string.

#Code:

```
import os, sqlite3
conn = sqlite3.connect("Student.db")
cur = conn.cursor()
cur.execute('Create table Data(Name char, RollNo int)')
cur.execute('Insert into Data values ("Simit", 7), ("Trupal",
                                    10),("Dhiru", 8))
conn.commit()
cur.execute('Select Name from Data')
cur.fetchall()
cur.close()
```

Output:

["Simit", "Trupal", "Dhiru"]

```
import socket
def server_prog()
host=socket.gethostname()
port=5000
server_socket=socket.socket()
server_socket.bind((host,port))
server_socket.listen(2)
conn, address=server_socket.accept()
print("Connection from:"+str(address))
while True:
    data=conn.recv(1024).decode()
    if not data:
        break
        print("From Connected user:"+str(data))
        data=input('->')
        conn.send(data.encode())
        conn.close()
server_program()
```

Output:

```
Python 3.6 socket_server.py
Connection from: ('127.0.0.1', 57822')
From connected user: Hi!
-> Hello
From connection user: Awsome!
Bye!
```

Practical-8

1) Server

Algorithm:

1) Import 'socket' from the python library to implement it in the program

2) define a function 'server_program()' to state its functioning

3) store a method gethostname() with socket in host veiable

4) Store a port address in a veiable such that port no should be above 1024.

5) Get instance socket.socket() method and keep it in a veiable.

6) Take the bind host address and port together

7) Configure the no. of clients the server can listen simultaneously

8) Accept new connection and print the connection add.

9) Accordingly if the data is not received then break, send the data to the client and then close the connection.

2) Client:

1) Import socket from python library

2) Define a user defined function for client program

3) ~~Run the local~~ get host name ~~from~~ and store it in a variable

4) Store the port no. in a variable, then instantiate

5) Connect to the server using connect() method

6) Take the input and then send the message

7) Receive the response with recv() method with port

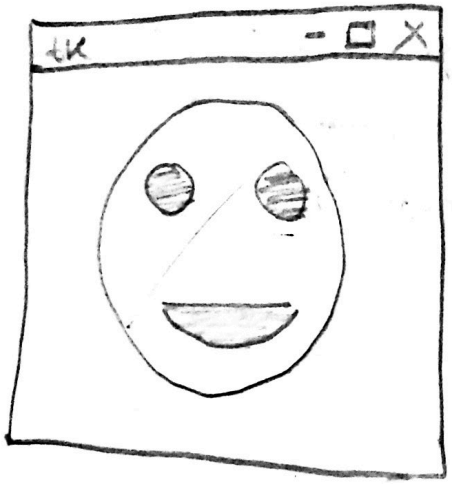
8) Show the terminal and then again take the input

9) Call the defined function and run the program

```
import socket
def client_program():
host = socket.gethostname()
port = 5000
client_socket.connect((host,port))
message = input(" ->")
while message.lower().strip() != 'bye':
    client_socket.send(message.encode())
    data = client_socket.recv(1024).decode()
    print('Received from server: ' + data)
    message = input(" ->")
    client_socket.close()
client_program()
```

```
from tkinter import *
root = Tk()
c = canvas (root, width = 500, height = 500)
c.pack()
face = c.create_oval ((50, 50, 350, 350), outline = "bl
                      fill = "yellow")

eye1 = c.create_oval (125, 125, 275, 175, fill = "
eye2 = c.create_ oval (125, 125, 275, 175, fill = "bl
mouth = c.create_arc (125, 225, 275, 275, fill = "b
root.mainloop (c)
```



Practical - 6

Aim: 1. Demonstrate the use of GUI by creating a human face and ~~factorial prog~~ Fahrenheit progm

i) Algorithm

1) Import relevant method from tkinter library

2) Create a parent window object using Tk() method

3) Create a object from canvas and place it on to the parent window

4) Use the create_oval(), ~~create~~ create_arc() methods to define the face and it features

5) Finally use the mainloop() method to run the progra

ii) Fahrenheit program

1) Import the relevant method from the tkinter library

2) Use the entry box widget to take the input & doublevar() and store it in a variable

3) Define a function that converts Celsius in fahrenheit by using

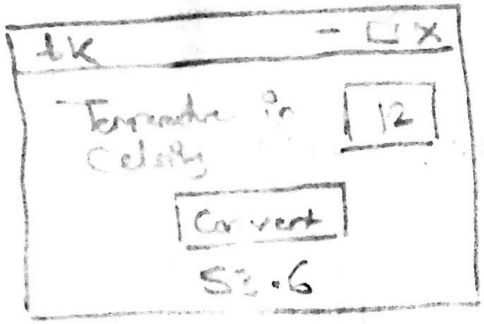
$$F^\circ = \frac{9}{5} \times {}^\circ C + 32$$

4) Use a label widget to display the Fahren value

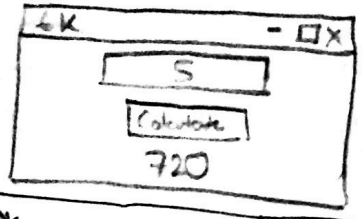
5) Use the button widget to call the convert and change the Fahrenheit variable value.

6) Call the mainloop() method.

```
from tkinter import*
root = Tk()
fahr = DoubleVar()
def convert(celsius):
    fahrenheit.set(9.0/5.0) * Celsius + 32)
l1=Label(root, text="Temperature in values :")
l1.grid(row=0, column=0)
e=Entry(root, textvariable=(Celsius)
e.grid(row=0, column=1)
Celsius=IntVar()
l2=Label(root, textvariable=Fahrenheit)
l2.grid(row=2, column=0, columnspan=2)
B=Button(root, text="Calculate", command=lambda
                                  convert(Celsius.get(
B.grid(row=1, column=0, columnspan=2)
root.mainloop()
```



```
from tkinter import*
def fact(n):
    if n==0 or n==1
        return 1
    else:
        return n*fact(n-1)
def c():
    result=fact(int(entrytext.get()))
    info.config(text=result)
r=tk()
entry=Entry(r)
entry.pack()
b=Button(r, text="Calculate", command=c())
b.pack()
info=Label(r, text="Factorial")
info.pack()
r.mainloop
```




```
from tkinter import*
def add(x,y):
    r.set(x.get()+y.get())
def sub(x,y):
    r.set(x-y)
def div(x,y):
    r set(x/y)
```

Practical-7

Aim: Factorial of a no. and calculate for attribute

→ **Algorithm:-**

(1) Import the relevant method from the tkinter library

(2) Define a function to calculate the factorial of a number

(3) Use the entry widget to take the input of the number

(4) Use the button widget to call the function in Step 2 and calculate the factorial

(5) Use the label() widget to display the factorial

6) Finally use the mainloop method.

II Calculator using GUI.

1) Define a function for add, sub, div, mul
2) Import the relevant method from the tkinter library
3) Use the entry() widgets to take the input, lab
4) Use the button widget to call such of the function in Step I.
5) Finally use the mainloop() method to run the program.

```
def mult(x,y):
    r.set(x*y)
n=Tk()
x=DoubleVar()
y=DoubleVar()
r=DoubleVar()
r1=Radiobutton(n, text="add", textvariable=r, lambda:add(
r2=Radiobutton(n, text="sub", textvariable=r, lambda:sub(
r3=Radiobutton(n, text="div", textvariable=r, lambda:div(
r4=Radiobutton(n, text="mul", textvariable=r, lambda:mult(
r1.pack()
r2.pack()
r3.pack()
r4.pack()
e1=entry(n,text="Entry an operand", textvariable=x)
e2=entry(n, text="Enter an operand", textvariable=y)
l=Label(w, variable=r)
e1.pack
e2.pack
l.pack
n.mainloop()
```

Output: