:S

## Practical :-1

Objective :- Demonstrate the use of different file accessing modes, different attributes and methods

Step1 :- Create a file object using open method and use the write access mode followed by writing some contents onto the the file and then closing the file.

Step2 :- Now open the file in read mode and then use read(), readline() and readlines() and store the output in variable and finally display the contents of variable.

Step3 :- Now use the file object for finding the name of the file, the file mode in which its opened whether the file is still open or close and finally the output of the softspace attribute.

```
file obj = open(...)
file obj.write("computer science subjects"+"\n")
file obj.write("DBMS\n PYTHON\n DS\n")
file obj.close() # file close
file obj = open("abc.txt","r") # read mode
#read()
str1 = file.obj.read()
print("The output of read method:", str1)
file obj.close()
>>>('The output of read method:', 'computer science
    subjects\n DBMS\n PYTHON\n DS\n')
#readline()
file obj = open("abc.txt","r")
str2 = file obj.readline()
print("The output of readline method:" str2)
file obj.close()
>>>('The output of readline method:', 'computer
    Science subjects\n')
#readlines()
file obj = open("abc.txt","r")
str3 = file obj.readlines()
print("The output of readlines method:", str3)
file obj.close()
>>>('The output of readlines method:', ['computer science
    subjects\n', 'DBMS\n', 'PYTHON\n', 'DS\n'])
# file attributes
a = file obj.name
print("name of file (name attribute):", a)
>>>(name of file (name attribute)', 'abc.txt')
b = file obj.closed
print("(close) attribute:", b)
>>>('(close) attribute:', 'True')
```

```
c = file obj.mode
print("file mode", c)
>>>('file mode', 'r')
d = file obj.software
print ("software", d)
>>> ('software :', 0)
```

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

# read mode
file obj = open("abc.txt", "r")
S = file obj.read()
Print("output of read mode", S)
>>>('output of read mode', 'Rahul')
```

```
# wt mode
file obj = open("abc.txt", "wt")
file obj = write("Rahul")
file obj.close()

# rt mode
file obj = open("abc.txt", "rt")
S1 = file obj.read(6)
Print("output of rt", S1)
file obj.close()
>>>('output of rt', 'Rahul')
```

```
# Append mode
file obj = open("abc.txt", "a")
file obj.write("data structure")
file obj.close()
file obj = close Open("abc.txt", "r")
S3 = file obj.read()
print("output of append mode:", S3)
file.close()
>>>('output of append mode:', 'Rahul data
                              structure')
```



Step 4 :- Now open the file obj in write mode write some another content close it subsequently then again open the file obj in 'wt' mode that is the update mode and write contents.

Step 5 :- Open file obj in read mode display the update, written contents and close open again in 'rt' mode with parameter passed out display the output subsequently.

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

Step 7:- Open the file obj in read mode, declare a variable and perform file obj dot tell method and store the output consequently in variable.

Step 8:- Use the seek method with the argument 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.



```
# tell()
file obj = open("abc.txt","r")
   pos = file obj.tell()
print("tell():", pos)
file obj.close
>>>('tell():', 0L)

# seek()
file obj = open("abc.txt", "r")
      st = file obj.seek(0,0)
print("seek(0,0) is: ", st)
file obj.close
>>>('seek(0,0) is', None)
file obj = open("abc.txt","r")
     st1 = file obj.seek(0,1)
print("seek(0,1) is: ", st1)
file obj.close
>>>('seek(0,1) is:', None)
file obj = open("abc.txt", "r")
     st2 = file obj.seek(0,2)
print("seek(0,2) is: ", st2)
file obj.close
>>>('seek(0,2) is:', None)

# finding length of different lines exist within lines
file obj = open("abc.txt", "r")
   stat = file obj.readlines()
print("output:" stat)
for line in stat:
        print(len(line))
file obj.close()
>>>('output:', ['Rahul data structure'])
   26.
```

class odd

```
class odd:
    def __iter__(self):
        self.num = 1
        return self
    def next(self):
        if self.num <= 0
            num = self.num
            self.num += 2
            return num
        else:
            raise StopIteration
y = iter(odd())
while true:
    print(next(y))
```

Output :-

1
3
5
7
9
11
13
15
17



Practical :- 2

Objective :- To demonstrate the use of iterators & iterable.

# odd numbers.

Algorithm :-

Step 1 :- Define a iter method with an argument and initialize the value and return the value.

Step 2 :- Define the next method with an argument and compare the upper limit by using a conditional statement Increment the value by 2.

Step 3 :- Now create an object of the given class and pass this object in the iter method.

# powel.

Algorithm:-

Step 1:- Define iter method with 3 Arguments. Initialize the first argument at 1 Initialize the other two arguments as "Enter the number" and "Maximum Limit of power" respectively.

Step 2:- Define the next method with an argument and compare it by using a conditional statement. Increment the value by 1.

Step 3:- Now create an object of the given class and pass this object in the iter method & use the while conditional statement to print.



```
class myiter:

    def __iter__(pow):
        pow.n = 1
        pow.n1 = int(input("Enter the number"))
        pow.n2 = int(input("In Maximum Limit
                            of power"))
        return pow
    def next(pow):
        if pow.n <= pow.n2:
            num = pow.n1 ** pow.n
            pow.n += 1
            return num
        else:
            raise StopIteration
y = iter(myiter())
while True:
    print(next(y))
```

Output :-

```
>>> Enter the number = 2
>>> Maximum Limit of power = 4
2
4
8
16
```

```
class myrange:
    def __iter__(self):
        self.a = 1
        return self
    def __next__(self):
        if self.a <= 20:
            x = self.a
            self.a += 1
            return x
        else:
            raise StopIteration

myclass = myrange()
mydata = iter(myclass)
for x in mydata:
    print(x)
```

Output :-

| | |
|---|---|
| 1 | 11 |
| 2 | 12 |
| 3 | 13 |
| 4 | 14 |
| 5 | 15 |
| 6 | 16 |
| 7 | 17 |
| 8 | 18 |
| 9 | 19 |
| 10 | 20 |



# Range

Algorithm :-

Step 1 :- Define a iter method with an argument & initialize the value & return that value

Step 2 :- Define the next method with an argument & compare the upper limit by using a conditional statement.

Step 3 :- Now create an object of the given class & pass this object in the iter method & use the for statement to print.

# multiple

Algorithm:-

Step I:- Define a iter() with an argument and initialize the value and return the value.

Step II:- Define a next() with an argument and compare the upper limits by using conditional statement

Step III:- Now create an object of the given class & pass this object in the user method.

```
class multi:
    def __iter__(self):
        self.m = 1
        return self
    def next(self):
        if self.m<=10:
            num = self.m
            self.m+=1
            table = 2*num
            print("2 *", num, "=", table)
        else:
            raise StopIteration.
x = iter(multi())
while True:
    print(next(x))
```

output:-

Enter a number :- 2

2X1 = 2
2X2 = 4
2X3 = 6
2X4 = 8
2X5 = 10
2X6 = 12
2X7 = 14
2X8 = 16
2X9 = 18
2X10 = 20

```
def accept_age():
    age=int(input("Enter your age:"))
    if age>30 or age<15:
            raise value error
    else:
        print("your age is",age)

valid=false
while not valid:
    try:
        age=accept_age()
        valid=True
    except,value error:
        Print("your age is not in range")
```

Output:-

Enter your age:40
your age is not in range

Enter your age:18
your age is 18.



## Practical:-3

Aim:- Demonstrate the use of expection handling

1) WAP to check the range of age of the students in given class & if age does not fall in given range we value error exception otherwise return the valid no.

Algorithm:-

Step1:- Define a function which will accept the age of the student from standard input.

Step2:- use if condition to check whether the input age fells in range & so return the age else use value error exception.

Step3:- Define the while loop to check whether the boolean expression holds true. we the true block to accept the age of student & terminated the looping condition.

a) WAP to check whether the number in give class & if the number is a floating point use value ~~whether~~ error as exception for the given input.

Algorithm:-

Step 1:- use try block & accept the input using input() & convert it into integer datatype and subsequently terminate the block.

Step 2:- use the except block with exception as value error & display appropriate message in suspicious code is part of try block

```
while True:
    try:
        a=int(input("Enter a number"))
        print("valid number")
        break
    except value error:
        print("Not a valid number! try again")
```

output:-

Enter a number: 15.6
Not a valid number! try again

Enter a number: 15
valid number.

```
def divide(a,b):
    ans = a/b
    return ans
while True:
    try:
        a = int(input("Enter 1st number:"))
        b = int(input("Enter 2nd number:"))
        ans = divide(a,b)
        print("divide of ",a,"and",b,"is",ans)
        break
    except zero division Error:
        print("Error!")
```

Output :-

Enter 1st number : 1

Enter 2nd number : 1

Division of 1 and 1 is 0



3) WAP to demonstrate use of zero division error.

Algorithm :-

Step 1 :- Use the try block & accept the input using input() & then convert it into integer datatype.

Step 2 :- Define a function with 2 parameters & to divide the number given by user.

Step 3 :- Define while loop to check whether the boolean expression holds True.

Step 4 :- Use except with zero division error & print the message.

24/12/19

## Practical :- 4

Aim :- Demonstrate the use of regular expression.

Theory :- Regular expression represents the seq of character which is mainly used for finding & replacing the given patterns in a string and for this use import re module and common usage of regular expression involved following foundation.

- Searching a given string
- finding a string
- Breaking a string into smaller substring.
- Replacing part of string.

Q1) Write a regular expression aggregating numeric and alphabetic values from a given string.

Algorithm :-

step1 :- Now apply string & pattern in findall() and display the output.

Skp2 :- \d is used for matching all decimal digits whereas \D is used to match non decimal digits.

code :-

```
import re
String = "hello 1234 abc 4567 "
result = re.findall ("\d+", string)
result1 = re.findall ("\D+", string)
print (result)
print (result1)
```

output :-

```
>>> ['1234', '4567']
>>> ['hello', 'abc']
```

8.

code 2:-

```
import re
String = "python is an important language"
result = re.search ("\APython", string)
print (result)
if result:
    print("Match found")
else:
    print("Match not found")
```

Output:

```
>> <re.Match object: span=(0,6);
        match = "python">

>> match found.
```



2) write a regular expression for finding the match string at the beginning of given sequence.

Algorithm:-

Step1:- Import re module and apply a string

Step2:- use search() with "\APython" and string as two parameters.

Step3:- Now display the output

Step4:- Now use if conditional statement for user to know whether the match is found or not.

3) write a regular expression to check whether the given mobile number starts with 8 or 9 and the total length of digit should be almost 10.

Algorithm:-

Step1:- Import re module and apply a string of mobile no. 8.

Step2:- Now use for conditional statement to find if the number is starts with 8 or 9 and the total number should length of 10. use match() inside for statement to find the match in given string.

Step3:- use if conditional statement to know whether we have a match or not if we have use group() to display the output and if we don't display incorrect mobile no.



code 3:-

```
import re
li=["9876543210","8699337204","8097888436",
    "7654321098"]
for element in li:
    result=re.match("[8-9]{1}[0-9]{9}",
                    element)
    if result:
        print("correct mobile no")
        print(result.group(1))
    else:
        print("Incorrect mobile no")
```

Output:

```
>>> correct mobile no
9876543210
correct mobile no
8699337204
correct mobile no
8097888436
Incorrect mobile no.
```

Code 4:-

```
import re
string = "python is interpreted language"
result1 = re.findall("\w*", string)
result2 = re.findall("\w+", string)
print(result1)
print(result2)
```

Output:-

```
>>> ['python', '', 'is', '', 'interpreted', '', 'language']
['python', 'is', 'interpreted', 'language']
```



4) write a regular expression for extracting a word from given string along with space character in between the word and subsequently extract the word without space character.

Algorithm:-

Step1:- Import re module and apply a string.

Step2:- We findall() to extract a word from given string

Step3:- use "\w*" to extract word along with space & we "\w+" to extract word without space.

Step4:- Now display the output

5) Write a regular expression for extracting first and last word from a string:

Algorith:-

Step1:- Import re module and apply a string.

Step2:- we findall() in which use "\w+" as one parameter to find first world of string then use "\w+$" as parameter to find last word of string

Step3:- Now display the result.

6) write a regular expression for extracting the date in format dd-mm-yyyy by using the findall() where the string had following format Rahul 301 24-12-2019.

Algorithm:-

Step1:- Import re module and apply string.

Step2:- use findall() method and use '\d{2}-\d{2}-\d{4}' as an parameter

Step3:- Now display the output.

code 5:-

```
import re
string = "python is important"
result = re.findall("\w+", string)
result1 = re.findall("\w+$", string)
print (result)
print (result1)
```

Output:-

```
>>> ['python']
>>> ['important']
```

code 6:-

```
import re
string = "Rahul 301 24-12-2019"
result = re.findall("\d{2}-\d{2}-\d{4}", string)
print (result)
```

Output:-

```
>>> ['24-12-2019']
```

code 7:-

```
import re
string="abc@tcsc.edu"
result1=re.findall("\w+",string)
result2=re.findall("\w+.\w+",string)
result3=re.findall("[\w\.-]+",string)

print(result1)
print(result2)
print(result3)
```

Output:-

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

1) write a re for extracting the
   (1) username from email id
   (2) hostname from email id
   (3) both username & hostname from email id

Algorithm:-

Step1:- import re module and apply a string

Step2:- use findall() to find username, hostname and both of email id

Step3:- use "\w+" for username, use "\w.\w+" for hostname and use "[\w\.-]+" for both as parameter in findall()

Step4:- Display the output

07/01/2021

# Practical :- 5

Topic :- GUI components

#1

Step1 :- Use the tkinter library for importing the features of the text widget.

Step2 :- create an object using the TK()

Step3 :- create a variable using the widget label and use the text method.

Step4 :- use the mainloop() for triggering of the corresponding above mention events.

~~step5~~ :-

#2:

Step1 :- use the tkinter library for importing the features of the text widget.

Step2 :- Create a variable from the text method and position it on the parent window.

Step3 :- Use the pack() along with the object created from the text() and use the parameters.

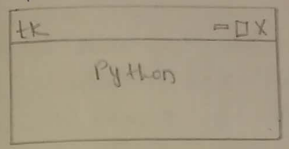
1) side = LEFT, padx=20
2) Side = LEFT, pady=30



#1: creation of parent window

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

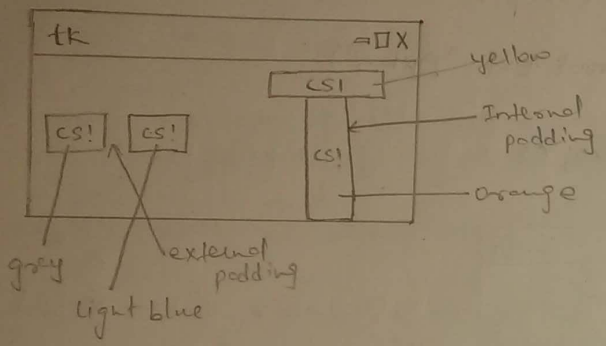
Output :-




#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 = "black",
           font = "10")
l3.pack(side = TOP, ipadx = 40)
l4 = Label (root, text = "CS", bg = "orange", fg = "black",
           font = "10")
l4.pack(side = TOP, ipady = 50)
root.mainloop()
```

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

## Practical :- 5(A)

Aim :- GUI components

#1 :-

Step1 :- Import the relevent methods from the tkinter library create an object with the parent window

Step2 :- use the parent window object along with the geometry() declaring specific pixel size of the parent window

Step3 :- Now define a function which tells the user about the given selection made from multiple option available.

Step4 :- Now define the parentwindow and define the option with control variable.

Step5 :- use the listbox() and insert options on the parent window along with the pack() with specifying anchor attribute.

Step6 :- 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 argument / corresponding value & triggered the function defined.

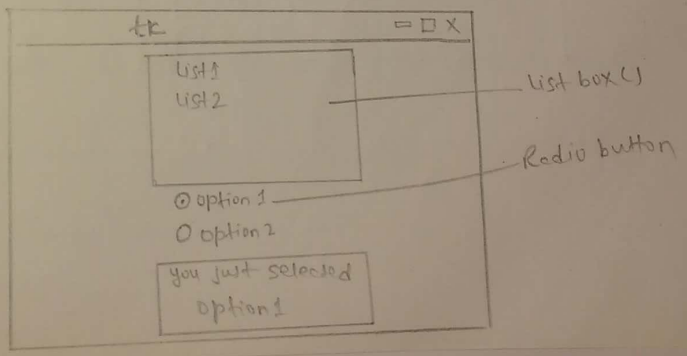


#1:

# 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, 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 = "option1", variable = var,
                 value = "option1", command = select)
r1.pack(anchor = N)
r2 = RadioButton(root, text = "option 2", variable = var,
                 value = "option 2", command = select)
r2.pack(anchor = N)
root.mainloop()
```

Output :-

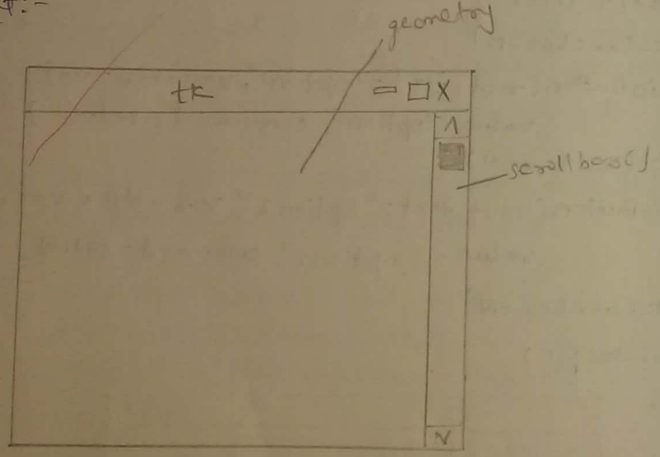


#2:-

#scrollbox()

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

Output:-






Step 7:- Now call the pack() for radio object so created and specify the argument 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 parent window

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 the scroll or object with side and fill attributes.

Step 6:- use the mainloop with the parent object.

#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(680x500) or any other suitable pixel value.

Step 4:- use the label widget along with the parent object created and subsequently use 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 parent attributes like width, height font.

Step 7:- use the Scrollbar() with an object. use the attribute of vertical the configure the same with object created from the Scrollbar() and the pack()

Step8:- Trigger the events using mainloop

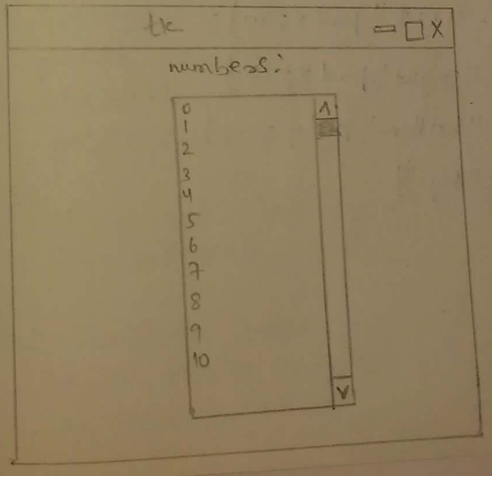


#3:-

#using frame widget.

```
from tkinter import *
window = Tk()
window.geometry("680 X 500")
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(11):
    listNodes.insert(END, str(x))
window.mainloop()
```

Output:-



#4:-

```
from tkinter import*
window=TK()
window.geometry("650X500")
frame=Frame(window)
frame.pack()
leftframe = Frame(window)
leftframe.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,text="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")
```



#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 defines the frame object from the method and place it on to 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 on the given frame with the attribute as text, active background and foreground.

Step 6:- Now we 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".

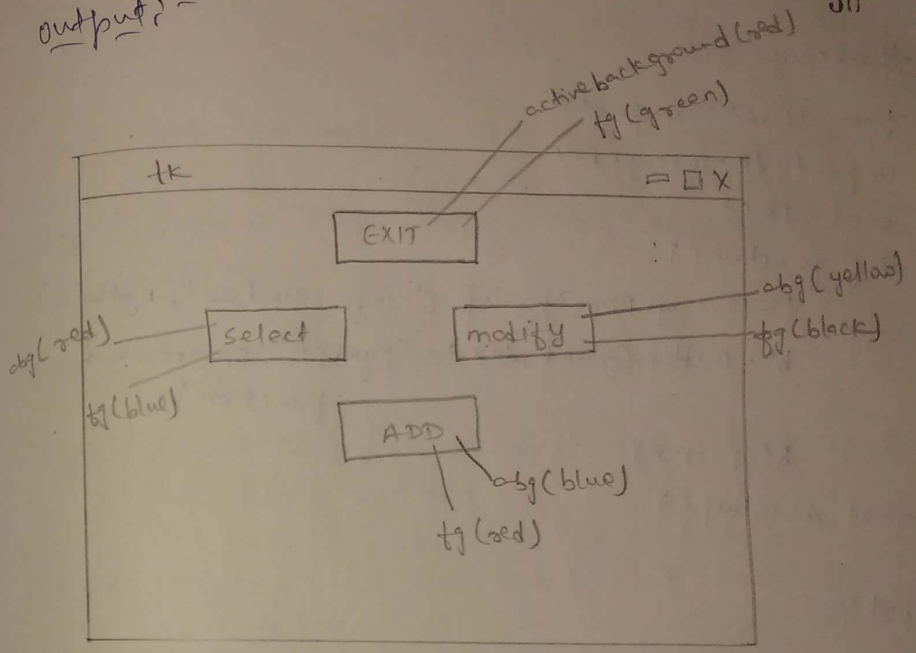
Step 8:- Create another button object & place it onto the RIGHT frame & label the button as ADD.

Step 9:- Add another button & puts 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().



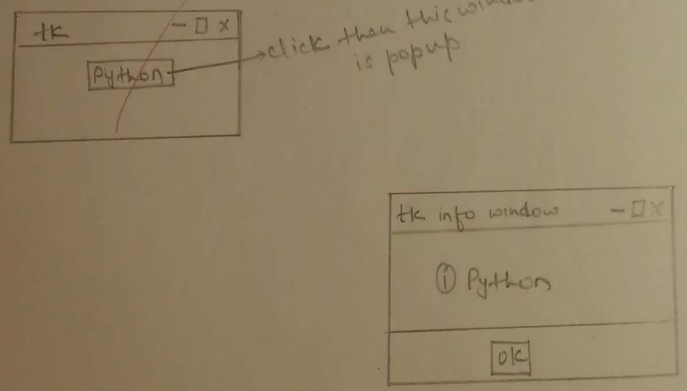
output:-



```
#message box
from Tkinter import *
import tkMessagebox
root = Tk()
def function():
    tkMessageBox.showinfo("info window", Python)
    b1 = Button(root, text = "python", command
                                    = function)

    b1.pack()
root.mainloop()
```

Output:-






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.

Step1:- Import the relevant methods from the tkinter library along with parent window object declared.

Step2:- use parentwindow object along with minsize function for window size.

Step3:- Define a function main, declare parent window object and use config(), title(), minsize(), label() as well as button() and use pack() & mainloop() simultaneously.

Step4:- similarly define the function second and use the attributes accordingly.

Step5:- Declare another function button along with parent object and declare button with attributes like FLAT, RIDGE, GROOVE, RAISED, SUNKEN along with the relief widget.

Step6:- finally called the mainloop() for event driven programming.

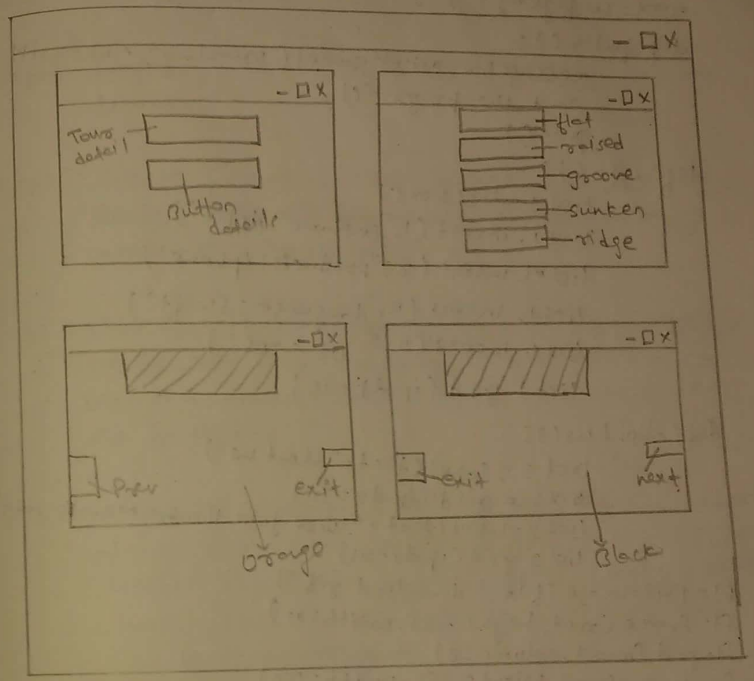


```
# Multiple window
# Different button (Relief())
from Tkinter import *
root = Tk()
root.minsize(300,300)
def main():
    top = Tk()
    top.config(bg="black")
    top.title("HOME")
    top.minsize(300,300)
    l = Label(top, text="SAN FRANCISCO
    \n Places of Interests \n Golden Gate Bridge
    \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: Rahul yadav \n
    For more details contact to our official
    account")
```

```
    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 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()
```

Output :-



```
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 aboutus():
         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=45)
f1.grid(row=1, column=0)
f2=frame(root, height=250, width=500)
f2.grid(row 1, column=1)
P2=P1.subsample(5,4)
l1=label(f1, image=P2, relief=FLAT)
l1.grid(row=1, column=0, padx=20, pady=15)
l2=label(f2, image=P1, relief=SUNKEN)
l2.grid(padx=25, pady=10)
```



## Practical :- 5(b)

Aim :- GUI components

Step1 :- Import relevant methods from the tkinter library

Step2 :- Create parent window object and use the .config method along with background color attribute specified.

Step3 :- Define a function finish with the messagebox widget which will display a message i.e a warning message and subsequently terminate the program.

Step4 :- 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 and finally use the grid() with ipadx attributes.

Step5 :- Define a function about us with label widget and text attribute and subsequently use the grid()

Step6 :- Use photo image 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 and subsequently use the grid() with row & column attribute specified.

Step 8:- Similarly create another frame object as declared by step 2.

Step 9:- create another frame object & use the subsample(5,4).

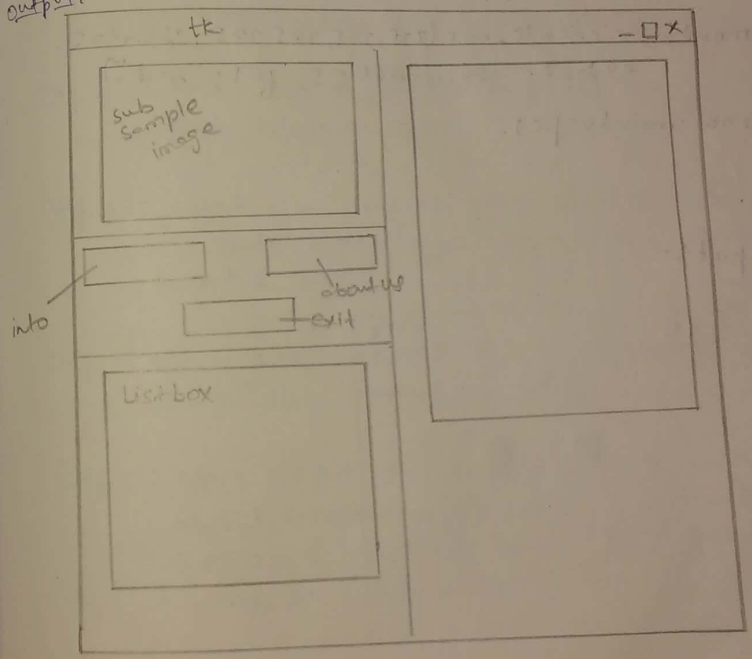
Step 10: use label widget along with the frame object, relief attribute and subsequently use the grid().

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



```
b1 = Button(f1, text="information", 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()
```

output:-

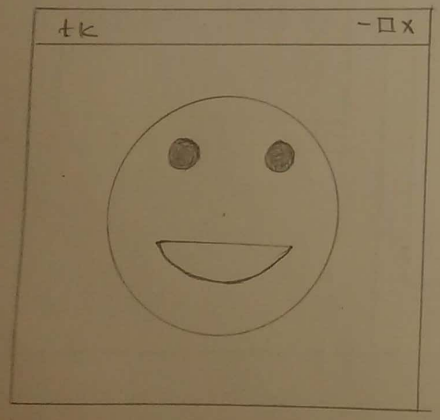


#code:-

```
from tkinter import *
root = Tk()
c = canvas(root, width=500, height=500)
c.pack()
face = c.create_oval(50,50,350,350, outline=
       "black", fill="yellow")
eye1 = c.create_oval(125,125,175,175, fill="black")
eye2 = c.create_oval(125,125,175,175, fill="black")

mouth = c.create_arc(225,225,275,275, start=0,
        extent=-180, width=5, fill="red")
root.mainloop().
```

#Output:-






## Practical:-6

Aim:- Demonstrate the use of GUI by creating a human face and converting celsius into fahrenheit.

Q1) W.A.P to draw human face using GUI

Algorithm:-

Step 1:- Import relevant method from tkinter library.

Step 2:- create an object corresponding to the parent window from TK()

Step 3:- create an object from canvas() & place it onto parent window along with height & width.

Step 4:- Now use pack() for positioning of widget onto the parent window.

Step 5:- Now create an object face & use object.create_oval() with coordinates 50, 50, 350, 350 & outline='black', fill="yellow" as attribute to create face.

Step 6:- Now create eye1 object & again use object.create_oval() with appropriate coordinates along with fill as attribute to create left eye.

Step 7: Now repeat the same step 6 to create right eye.

Step 8: create an object mouth & use object.create_arc() with appropriate coordinates, start=0, extent=-180 & fill="red", width=3 as attribute to create mouth.

Step 9: finally use the mainloop()

# code:-

```
from tkinter import*
window = Tk()
fahrenheit = DoubleVar()
fahrenheit.set(32.0)
def convert(celsius):
    fahrenheit.set((9.0/5.0)* celsius +32)
l1=Label(window, text="Temperature in celsius:")
l1.grid(row=0, column=1)
e = Entry(window, textvariable = celsius)
e.grid(row=0, column=1)
celsius = IntVar()
l2= Label(window, textvariable = fahrenheit)
l2.grid(row=2, column=0, columnspan=2)
B = Button(window, text="calculate", command=
    lambda: convert(celsius.get()))
B.grid(row=1, column=0, columnspan=2)
window.mainloop()
```



92) WAP to convert celsius into fahrenheit using GUI

Algorithm:-

Step 1: Import all the relevant method in tkinter library.

Step 2: creat object corresponding to the parent window from Tk().

Step 3: Now initialize fahrenheit as DoubleVar() & set it to 32.0

Step 4: Now define a function convert with argument celsius. to convert celsius into fahrenheit using. set().

Step 5: Now create an object l2 using label() & place it onto parent window & use text attribute as enter a no.

Step 6: Now use grid() for position the object onto the parent window.

Step 7: Initialize celsius as integer using IntVar().

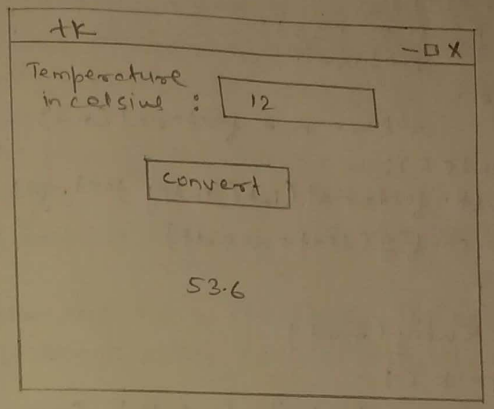
Step 8: Create another object & use entry widget to enter the input and place it onto the parent window.

Step 9: Now use grid() for positioning the object onto parent window with textvariable attribute.

Step 10: Now again use label() along with textvariable attribute to display output & use grid() for positioning.

Step 11: Finally use mainloop().

# output :-

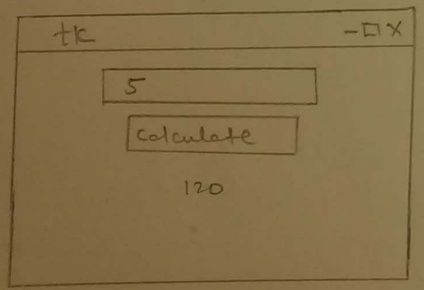


# Code :-

```
from tkinter import *
def factorial(n):
    if n==0 or n==1:
        return 1
    else:
        return n * factorial(n-1)
def calculate():
    result = factorial(int(entry_text.get()))
    info.config(text=result)

root = Tk()
entry_test = Entry(root)
entry_test.pack()
btn = Button(root, text="calculate", command=
        calculate)
btn.pack()
info = label(root, text="factorial")
info.pack()
root.mainloop()
```

# Output :-






Aim :- write a program to find factorial of number & use arithmetic operations on two numbers using GUI.

(A) WAP to find factorial of number using GUI.

Algorithm :-

Step 1 :- Import relevant methods from tkinter library.

Step 2 : Now define a function factorial to calculate factorial using recursive function.

Step 3 : Define another function calculate to all factorial function.

Step 4 :- Now create an object with entry() and use pack() for positioning on parent window.

Step 5 :- Now create an object with button() along with command = attribute to calculate factorial.

Step 6 : Now again create an object with label() to show output.

Step 7 : finally use the mainloop().

Q2) WAP to perform arithmetic operation on 2 number using GUI.

Algorithm:-

Step 1:- Import relevant methods from tkinter library

Step 2: Now create an object corresponding to parent window

Step 3: Now define a function calculate to carry out arithmetic operations on 2 numbers.

Step 4: Now create an object with label() as num1 & num2 and use grid() to place it onto parent window.

Step 5: create objects with entry() to take input from user().

Step 6: Now initialize V as integer using intVal()

Step 7: Now create 4 objects with Radiobutton() to choose any one of arithmetic operations & use grid() for positioning

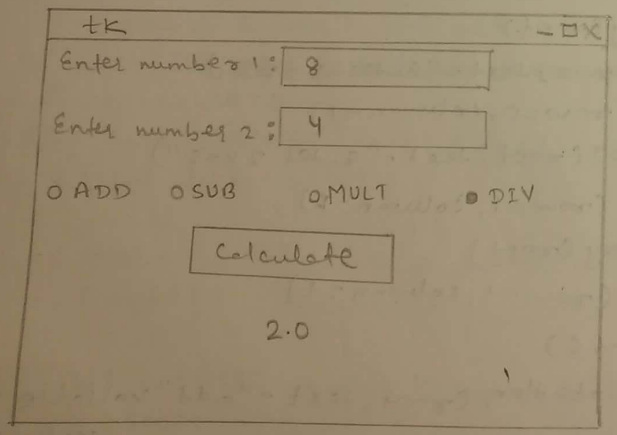


# code:-

```
from tkinter import *
def calculate():
    if int(v.get())==1:
        res=int(e1.get())+int(e2.get())
        l3.config(text=res)
    elif int(v.get())==2:
        res=int(e1.get())-int(e2.get())
        l3.config(text=res)
    elif int(v.get())==3:
        res=int(e1.get())*int(e2.get())
        l3.config(text=res)
    else:
        res=int(e1.get())/int(e2.get())
        l3.config(text=res)
root=Tk()
l1=Label(root,text="Enter a no:")
l1.grid(row=0,column=0)
e1=Entry(root)
~~e1.grid(row,text="Enter 2 no:")~~
e1.grid(row=0,column=1)
l2=Label(root,text:"Enter 2 no's")
l2.grid(row=1,column=0)
e2=Entry(root)
e2.grid(row=1,column=1)
V=IntVar()
r1=Radiobutton(root,text="add",variable=V,
                                value=1)
r1.grid(row=2,column=0)
```

```
r2 = gRadiobutton (root, text = "sub", variable = v,
                   value = 2)
r2.grid (row=2, column = 1)
r3 = Radiobutton (root, text = "mult", variable = v,
                   value = 3)
r3.grid (row = 2, column = 2)
r4 = Radiobutton (root, text = "Div", variable = v,
                   value = 4)
r4.grid (row = 2, column = 3)
B = Button (root, text = "Calculate", command =
                   calculate)
B.grid (row = 3, column = 1, columnspan = 2)
l3 = Label (root)
l3.grid (row = 4, column = 1)
root.mainloop()
```

# Output :-






onto parent window.

Step 8 : Now create an object with button() along with command attribute to carry out the arithmetic operations of user choice.

Step 9 : Now create an object with label() to show output

Step 10 : Finally use the mainloop().

# Practical :- 8

Aim :- Demonstrate the use of Socket module and server client programs.

1. WAP to demonstrate use of Socket module and server client programs.

Algorithm :-

Step 1 : Import the socket module to import relevant methods.

Step 2 : Define a function as server_program to get hostname

Step 3 : Now get value for port variable to initialize port no. above 1024

Step 4 : Use socket() to get instance.

Step 5 : Now use bind() function to bind host address and port together to configure how many client the server can list simultaneously.

Step 6 : Now use accept() to accept new connection.



# Code :-

```
import socket
def server_program:
        host = socket.gethostname()
        port = 5000
        server_socket = socket.socket()
        server_socket.blind((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()
```

# Output

```
$ Python 3.6 socket_server.py
Connection from : ('127.0.0.1', 57822)
from connected user : Hi
-> Hello
from connected user : How are you?
-> Good
from connected user : Awesome!
-> OK then, bye!
```

#code:-

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

#Output:-

```
$ Python 3.6 socket_client.py
→ Hi
Received from server: Hello
→ How are you?
Received from server: Good
→ Awesome
Received from server: OK then, bye!
→ Bye
```



Step 7: Now print the address

Step 8: use while loop as true to receive data stream

Step 9: Now close the program.

2. (for Socket client program)

Algorithm:

Step 1: Import Socket module to import methods that are relevant

Step 2: Define a function client_program get the hostname & give port a value 5000

Step 3: Now again initialize by using Socket. Socket()

Step 4: use connect() to connect the server.

Step 5: Now take the input("+")

Step 6: use while conditional loop to send a message.

Step 7: Now use decode to receive response.

Step 8: Now show the data

Step 9: Again take input.

Step 10: close the program by using close().

# code in shell environment.

```
>>> import Sqlite3
>>> conn = sqlite3.connect("student1.db")
>>> cur = conn.cursor()
>>> cur.execute('create table student
    (roll_no int(5) primary key, name varchar(50)
    not null, address varchar(50) not null,
    class varchar(50), dob date)')
<sqlite3.cursor object at 0X0322EBE0>
>>> cur.execute('insert into student values
    (101,"Rahul","kandivali","FyCs","29/09/2002")')
<sqlite3.cursor object at 0X0322EBE0>
>>> cur.execute('insert into student values
    (102,"Tushar","kandivali","FyCs","15/07/2001")')
<sqlite3.cursor object at 0X0322EBE0>
>>> ~~select~~ cur.execute("select* from student")
<sqlite3.cursor object at 0X0322EBE0>
>>> cur.fetchall()
[(101,'Rahul','kandivali','FyCs','29/09/2002')
(102,'Tushar','kandivali','FyCs','15/07/2001')]
>>> cur.execute("update student set dob="15/07/2000"
    where roll_no = 102")
<sqlite3.cursor object at 0X0322EBE0>
>>> cur.execute('select * from student where
        dob ="29/09/2002")
<sqlite3.cursor object at 0X0322EBE0>
```



## Practical :- 9

Aim :- Demonstrate the use of database connectivity.

Algorithm :-

Step 1 : Import sqlite3 module to import relevant methods.

Step 2 : Now initialize a variable conn to connect by using connect() to a new database using extention.db

Step 3 : Now initialize a variable to connect to cursor()

Step 4 : Now use cur.execute() to create a table, insert values into table & use DML, DDL, statements to manipulate the data in this database.

Step 5 : use fetchall() to show the output.

Step 6 : to use commit to ~~store~~ th Save all changes.

Step 7 : use close() to terminate the program.

```
>>> cur.fetchall()
[(101, 'Rahul', 'Kandivali', 'FyCs', '29/09/2002')]

>>> cur.execute('commit')
<sqlite3.cursor object at 0x0322EBCO>

>>> cur.close()
```