★★ INDEX ★★

No.	Title	Page No.	Date	Staff Member's Signature
1.	Demonstrate the use of different file accessing mode, different attributes, seq. read methods	02	28/11/19.	
2.	Iterators	25	17/12/19	
3.	Exception Handling.	28	19/12/19	
4.	Regular Expression.	31	9/1/2020	
5	GUI Components (A)	35	16/12/2020	
	GUI Components (B)	40	23/12/2020	
	- messagebox			
	- traversing			
	- Relief			
	- Displaying the image	45	6/12/2020	
	GUI Components (C)	47	13/2/2020	
	- Spinbox			
	- Paned window			
	- Canvas			
6	Database Connectivity	50	20/12/2020	

Practical No. 1

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

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 store the output in variable 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 it's 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 method)
fileobj.write("computer science subjects"+"\n")
fileobj.write("DBMS \n Python \n DS \n") # file write
fileobj.close() # file close

fileobj = open("abc.txt","r") # read mode
# read()
str1 = fileobj.read()
print("The output of read method:", str1)
fileobj.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)
fileobj.close()
>>>('The output of readline method:', 'computer science subjects \n')

# readlines()
fileobj = open("abc.txt","r")
str3 = fileobj.readlines()
print("The output of readlines method:", str3)
fileobj.close()
>>> ('The output of readlines method:', ['computer science
    subjects \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 = fileobj.mode
print ("file mode", c)
>>> ('file mode', 'r')
d = fileobj.softspace
print ("softspace", d)
>>> ('softspace :', 0)
```

```
# w+ mode
fileobj = open("abc.txt", "w+")
fileobj.write ("loukik sir")
fileobj.close()
```

```
# 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!
    'loukik sir')
```

```
# r+ mode
fileobj = open ("abc.txt", "r+")
s1 = fileobj.read(6)
print ("output of r+", s1)
file obj.close()
>>> ('output of r+', 'loukik')
```

```
# append mode
fileobj = open ("abc.txt", "a")
file obj.write ("data structures")
file obj.close()
fileobj = open ("abc.txt", "r")
s3 = fileobj.read()
print ("output of append mode :", s3)
fileobj.close()
>>> ('output of append mode :', 'loukik sir
    data structure')
```

Step 4: Now open the fileobj in write mode. write some another contents. close subsequently then again open the fileobj in 'wt' 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 fileobj 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 tellmethod and store the output consequently in variable.

Step 8: Use the seek method with the arguments with the opening the fileobj in read mode and closing subsequently.

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

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

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

# finding length of different 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: ', ['lowlevel or data structures'])
```


last

iter() and next()

```
mytuple1 = ("banana", "orange", "apple")
myiter1 = iter(mytuple)
print(next(myiter1))
myiter2 = iter(mytuple)
print(next(myiter1))
myiter1 = 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
unet1 = [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 by using for conditional statement. An iterable variable is to be declared in for loop which will iterate.

Step_3: Define a function name square with a parameter which will obtain output of square value of the given number. In similar fashion declare cube to get the value raised 3. and return the same.

Step_4: Call the declared function using function call.

Step5: Using for conditional statement specifying the range use the list tuple casting with map method declare a 'lambda' ie anonymous function and print the same.

Step6: Declare a distnum variable and declare some elements then use the map method with help of lambda function give two argument display the output.

Step7: Define a function even with a parameter then using conditional statements do Check whether the number is even and odd and return respectively

Step8: Define a class and within that define the iter() method which will initialise the first element within the container object.

Step9: Now use the next() and define the logic for displaying odd values.

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

for r in range(5):
    value = list(map(lambda x: x(r), funct1))
    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.num + 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
```

factorial.

```
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):
                fac = fac * i
            print(self.f-1, "! = ", fac)
        else:
            raise StopIteration
```

Output:

```
>>> f = fact()
>>> x = iter(f)
>>> x.next()
1! = 1
>>> x.next()
2! = 2
>>> x.next()
3! = 6
```

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.

Factorial:

Step 1: Define a iter() with arguments & initialize the value and return the value

Step 2: Define the next() with an argument and Compare the upper limits by using a Conditional Statement.

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

Jan 19/12

15

Practical No 3

Aim: Program to demonstrate exception handling.

Q 1] Write a program using the exception method of the native arithmetic errors.

Step 1: Use the try block and except the input using the raw input method and convert it into the integer datatype & subsequently terminate the block.

Step 2: Use the except block with the exception name as value error and display the appropriate message if the suspicious code is part of the try blocks.

Q. 2] Write a program for accepting the file in a given mode & use the environment error as an exception for given input

Step 1: Within the try block, open the file using the write mode and write some content in the file.

Step 2: Use the except block with IO error & display the message regarding missing of the file or incompatibility of the mode use the else block to display a message that the

Program:

```
while True:
    try:
        x = int(input("Enter class"))
        break
    except valueError:
        print("Enter Numeric value")
```

Output:

```
enter class: 467
```

Program:

```
try:
    fo = open("abc.txt", "w")
    fo.write("Vivek Tiwari")
except IOError
    print("Error while writing on the file").
else:
    print("Operation carried out succesfully")
    fo.close()
```

Output:

```
Operation carried out succesfully.
```

Program:

```
def assert_(n)
    assert (len(n)==0)
    print ("list is empty")
var1:[]
print (assert_(var1))
```

Output:

List is empty.

Program:

```
def acceptage():
    age = int(input("Enter age:"))
    if age > 30 or age < 16:
        raise ValueError
    return age
valid = False
while not valid:
    try:
        age = acceptage()
        valid = True
    except ValueError:
        print ("Not a valid age")
```

Output:

```
Enter age: 4
Not a valid age
Enter age: 18
```

operation is carried out sucessfuley.

Q3) write a program using the assert() to check if the list elements are empty.

Step1: Define a function which accepts an argument and check using the assert statement whether the given list is empty and accordingly return the message.

Step2: close the function and is the body of program and define certain elements in list & take some appropriate action.

Q4) write a program to check the range of the age of the students in given class and if the age do not fall in given range else the value error exception otherwise return the valid number.

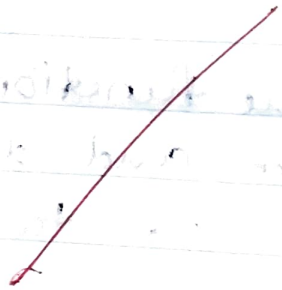
Step1: Define a function which will accept the age of the student from the standard input.

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

PS.

Step 3: Define the while loop to check whether the boolean expression holds true. Use the try block to accept the age of student & terminate the looping condition

Step 4: Use except with ValueError & print the message not a valid range



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'\dt'
string = 'hello 123, howdy 789, us howru'
output = re.findall(pattern, string)
print(output)
```

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

split()

```
import re
pattern = r'\dt'
string = 'hello 123, howdy 789, us howru'
output = re.split(pattern, string)
print(output)
```

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

Practical - 4

Topic: Regular expression

Step 1: Import re module declare pattern & declare sequence use match method with declare arguments, if arguments matched then print the same, otherwise print patter NOT FOUND!

Step 2: Import re module declare pattern with literal & meta Character. Declare string value. use the findall () with arguments & print the same.

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

Step 4: Import re module. Declare string & accordingly declare pattern replace blank space with no space use sub() with 3 arguments & print the string without spaces.

Step 5: Import re module declare a sequence use search method for finding subsequently use group() with dot operator as 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 statement. use if condition for checking first number is either 8 or 9 & next number are in range of 0 to 9 & check whether the entered numbers are equal to 10. If criteria matches print all number matches otherwise print failed.

```
# no-space
import re
string = 'abc defghi'
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 0x028IDF00>
    python.
```

```
# verifying the given set of phone numbers.
import re
list1 = ['8004567891', '9145673210', '7865432981',
         '9876543201']
for values 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 of domain

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

Counting of first 2 letters.

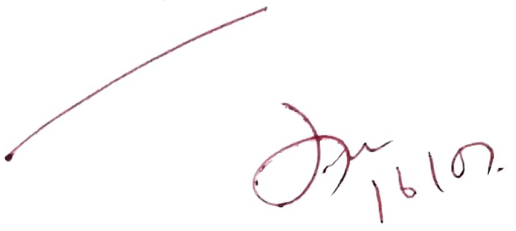
```
import re
s = 'mr.a, ms-b, ms.c, mr.t'
p = r'[/ms/mr/]'
o = re.findall(p,s).
print(o)
m = 0
f = 0
for v in o:
```

Step 7: Import re module declare a string use the module a string use the module with findall() for finding the vowels in the string & declare the same.

Step 8: Import re module, declare the host & domain name. Declare pattern for separating the host & domain name. use the findall() & 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 & subsequently use the if condition. Check whether condition satisfy add up the or else increment value. & display the values subsequently.

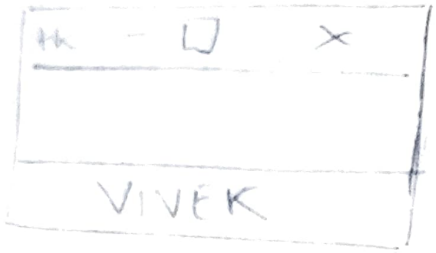
```
if ( v == 'ms' )
    f = f + 1
else:
    m = m + 1
print ( " No. of males is: ", m )
print ( " No. of females is: ", f )
>>> ['mr', 'ms', 'ms', 'mr']
('No of males is:' 2)
( No. of females is :; 2)
```



Text

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

Output :



Practical 5

Aim: GUI Components

Step 1: Use the tkinter library for importing the feature of the text widget

Step 2: Create a variable from text method and position it on the parent window.

Step 3: Use the pack method along with the object created from text method.

Step 4: Use the main loop method for triggering of corresponding events

Step 5: Use the tkinter library for importing the feature of text widget.

Step 6: Create a variable from text method and position it onto parent window.

Step7: Use the pack method along with the object created from text method and use the parameter.

Step8: side = LEFT , padx = 20
side = LEFT , pady = 30
side = TOP , ipadx = 40
side = TOP , ipady = 50

Step9: Use the main loop method for triggering of corresponding events.

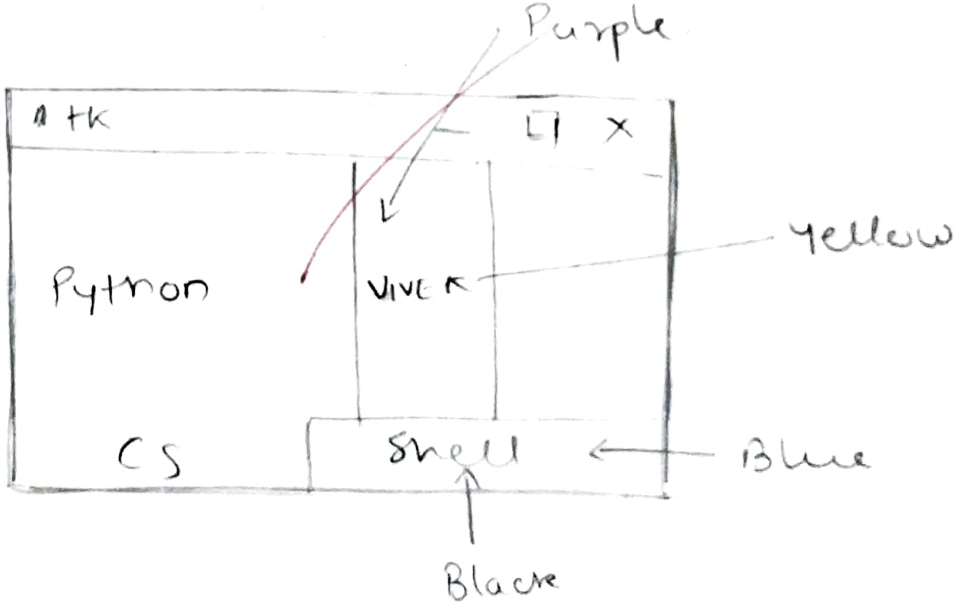
Step10: Now repeat the above steps with the label method which takes the following arguments:

i) Text attribute which defines string.
ii) Bg (background) colour.
iii) Fg (foreground) colour.
iv) Name of the parent window.
v) Use the pack method with relevant padding attribute.

Label

```
from tkinter import *
root = Tk()
l = Label(root, text = "PYTHON")
l.pack(side = LEFT, pady = 30)
l = Label(root, text = "VIVEK", bg = "purple", fg = "yell
l.pack(side = TOP, ipady = 40)
m = Label(root, text = "CS")
m.pack(side = LEFT, padx = 20)
m2 = packLabel(root, text = "shell", bg = "blue", fg = "blac
m2.pack(side = LEFT, ipadx = 50)
root.mainloop()
```

Output:



Radiobutton

```
from tkinter import *
root = Tk()
def sel():
    selection = "You selected the option " + str(var.get())
    l = label.config(root, text = "selection", justify = LEFT)
    l.pack(anchor = S)
    v = IntVar()
r1 = Radiobutton(root, text = "option 1", variable = "v",
                 value = "1", command = set)
r1.pack()
r2 = Radiobutton(root, text = "option 2", variable = "v",
                 value = "2", command = set)
r2.pack()
r3 = Radiobutton(root, text = "option 3", variable = "v",
                 value = "3", command = set)
r3.pack()
r4 = Radiobutton(root, text = "option 4", variable = "v",
                 value = "4", command = set)
r4.pack()
root.mainloop()
```

* WAP making use of the control variable and button widget for selection of the given option.

Step 1: Use the tkinter to import relevant method.

Step 2: Define a function which tells the user about the given selection need of the multiple options available.

Step 3: Use the configuration method along with label object and call the variable as an argument within the method.

Step 4: Now define the parent window & define the option using control variable

Step 5: Now create an object from the radio button method which will take the following arguments.

(i) Positioning on parent window
(2) Defining the text variable [1, 2, 3, 4]
(3) Define the variable argument.
(4) Corresponding value and trigger the given function.

Step 6: Pack method for the corresponding radio objects so created & specify the attribute as an anchor attribute

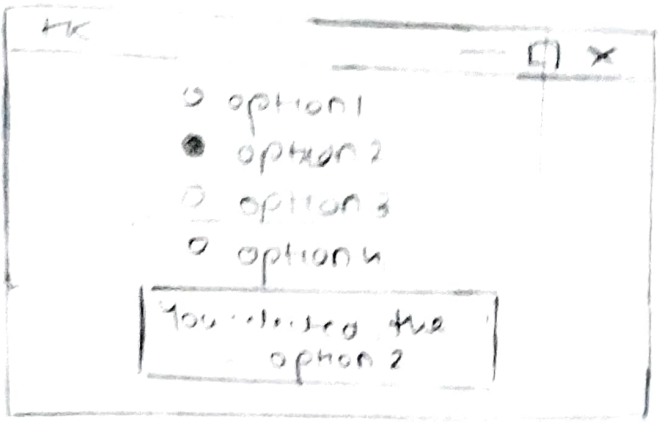
Step 7: Now define the label object from the corresponding method and place it on parent window.
Subsequently, use pack method for this window and make use of the mainloop method.

Step 8: Import the relevant method from tkinter library.

Step 9: Define the object corresponding to the object window and define the size of parent window in terms of number of pixels.

Step 10: Now define the frame object from the method and place it onto the parent window.

output:

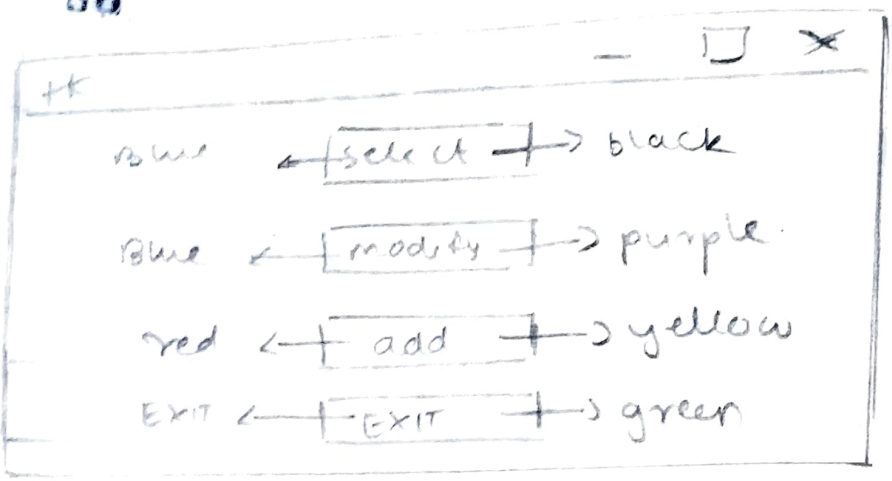


Frame Object

```
from tkinter import *
top = Tk()
top.geometry('100x200')
frame = Frame(top)
frame.pack()
leftframe = Frame(top)
Leftframe.pack(side=LEFT)
b1 = Button(frame, text="select", activebackground="red",
            fg='black')
b1.pack()
b2 = Button(frame, text="modify", activebackground
            ="blue", fg="purple")
b2.pack()
b3 = Button(frame, text="Add", activebackground="yellow",
            fg="red")
b3.pack()
b4 = Button(frame, text="exit", activebackground="green",
            fg="black")
b4.pack()
top.mainloop()
```

Output:

88


++
blue
select
black
Blue
modify
purple
red
add
yellow
EXIT
EXIT
green

Step11: Create another frame object the left frame and put it onto parent window on its left side.

Step12: Similarly define the right frame and subsequently define the button object placed onto the given frame with the attribute as text active bg and fg.

Step13: Now use the pack method along with the side attribute.

Step14: Similarly, create the button object corresponding to modify option and put it into the frame object with side equal to right attribute.

Step15: Add another button and put it on right frame object and term it as EXIT.

Step16: Use the pack method for all the object and finally use the main loop method.

Practical-5(B)

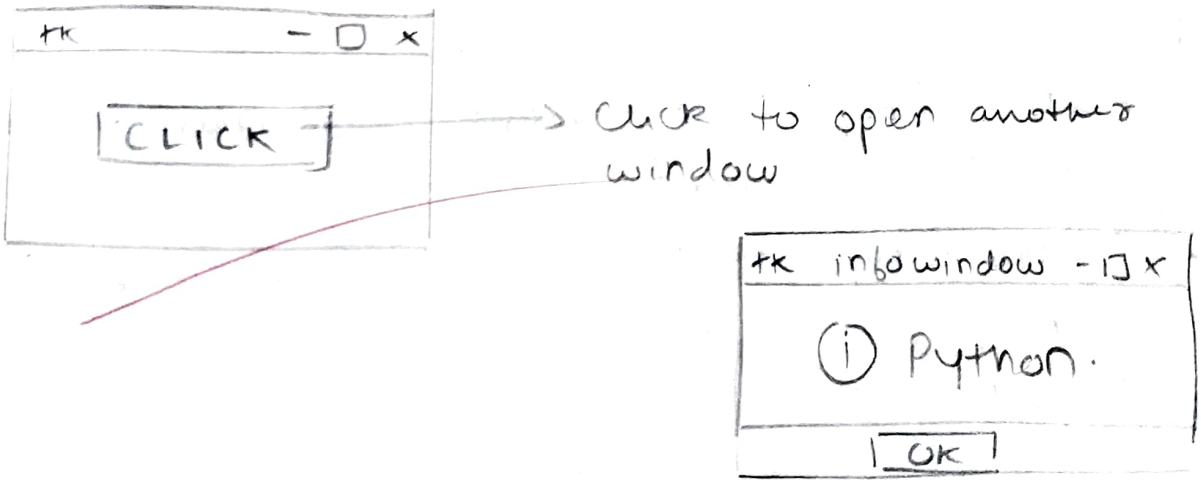
Aim: GUI Components.

Step1: Import the relevant methods from tkinter library.
Step2: Import tkmessagebox
Step3: Define a parent window object along with the parent window
Step4: Define a function which will use the tkmessagebox with showinfo method along with info window attribute.
Step5: Declare a button with parent window object along with the Command attribute.
Step6: 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 tkMessageBox
root = Tk()
def function():
    tkMessageBox.showinfo("info window","python")
    B1 = Button(root, text = "CLICK",command=msg)
    B1.pack()
    root.mainloop()
```

output:



Multiple window (Traversing).
Different button (Relief())

```
from Tkinter import *
root = Tk()
root.minsize(200, 200)
def main():
    top = Tk()
    top.config(bg="Blue")
    top.title("window 1")
    top.minsize(200, 200)
    L1 = Label(top, text = "Counter Strike
    \n Games to play: \n PUBG
    \n Red dead Redemption \n GTA V")
    L1.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 parentwindow object along with minsize function for window size

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

Step 4: Similarly define the function record and use the attribute 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.

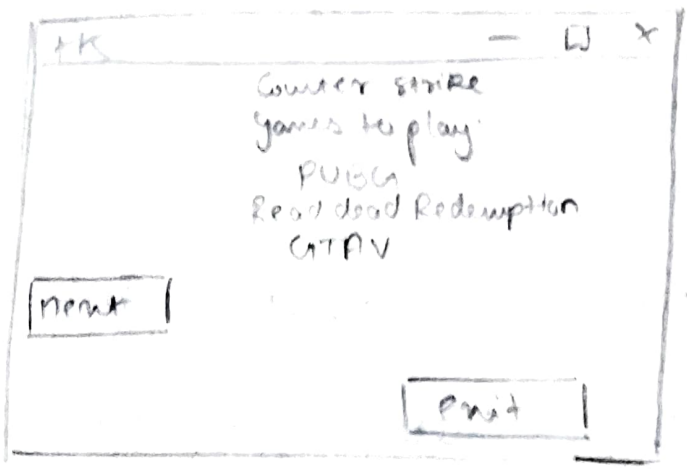
Step 6. Finally call the mainloop() for event driven programming.

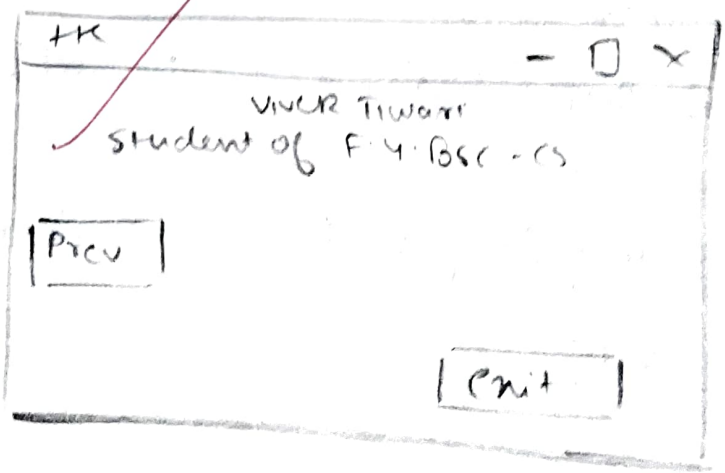
```
def second():
    top2 = Tk()
    top2.config(bg="black")
    top2.title("window 2")
    top2.minsize(200,200)
    L2= Label(top2, text="Vivek Tiwari In
      Student of F.Y.Bsc-CS")
    L2.pack()
    B3=Button(top2, text="prev", command=main)
    B3.pack(side=LEFT)
    B4=Button(top2, text="exit", command=terminate)
    B4.pack(side=RIGHT)
    top2.mainloop()

def button():
    top3=Tk()
    top3.geometry("200x200")
    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()
```

```
b
    BS = Button (top3, text = "Ridge Button", relief = RIDGE)
    BS.pack()
    top3.mainloop()
def terminate():
    quit()
BG Button (root, text = "GAME Details", command = main)
BG.pack()
BZ = Button (root, text = "BUTTON Details", command = button
BZ.pack()
root.mainloop()
```

Output:





Relief Style:

Step1: use the button object with the following attributes
1. The parent window
2. Text attribute.
3. Relief.

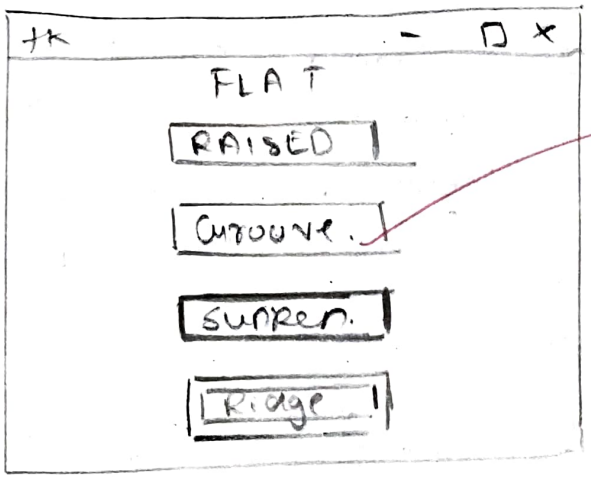
Step2: use the corresponding pack method for the respective button objects and trigger the corresponding event.

Step3: Finally use the mainloop method.

relief style.

```
from tkinter import *
root = Tk()
b1 = Button (root, text = "FLAT", relief = FLAT)
b1.pack()
b2 = Button (root, text = "RAISED", relief = RAISED)
b3 = Button (ro
b3.pack()
b2.pack()
b3 = Button (root, text = "Groove", relief = GROOVE)
b3.pack()
b4 = Button (root, text = "Sunken", relief = SUNKEN)
b4.pack()
b5 = Button (root, text = "Ridge", relief = RIDGE)
b5.pack()
root.mainloop()
```


Output:



Displaying the image

```
from tkinter import *
root = Tk()
root.title("This is title")
root.maxsize(1000, 900)
root.config(bg="black")
leftframe = Frame(root, bg="blue", height="400", width="200
leftframe.grid(row=0, column=0)
rightframe = Frame(root, bg="lightblue", height="400", width
rightframe.grid(row=0, column=0)
Label(leftframe, text="Python", height="2", width="20").
    grid(row=0, column=0)
image1=PhotoImage(file="image1.gif")
image1.subsample(1,2)
image2=PhotoImage(file="image1.gif")
image2.subsample(3,4)
Label(leftframe, image=image1).grid(row=0, column=1,
    padx=20, pady=10)
Label(rightframe, image=image2).grid(row=0, column=1,
    padx=10, pady=10)
toolbar=Frame(leftframe, width=200, height=400, bg="white
    grid(row=2, column=0)
Label(toolbar, text="Personal Info", height=2, width
  relief=RAISED).grid(row=0, column=0, padx=20, pady=10)
def name():
    print("My name is: Vivek")
def hob():
    print("My hobby is: Reading")
def add():
    print("Vasai")
def dob():
    print("13/11/2001")
```

Displaying the image:

Step1: Create an object corresponding to the parent window and use the following 3 methods Title, Maxsize / Minsize, Config.

Step2: Create a leftframe object from the frame method and place it onto the parent window with the height, width & the bg specified. Also use the grid method with row, column, padx, pady specified.

Step3: Now Create a right frame object from the frame method with width, height specified & the row, column value specified.

Step4: Create a label object from the label method and place it onto the leftframe with font attribute denoting the original image, with relief attribute used as RAISED value and subsequently use grid method with row, column value specified as (0,0) with some external padding values.

Step5: Now use the photoImage method with the file attribute specified

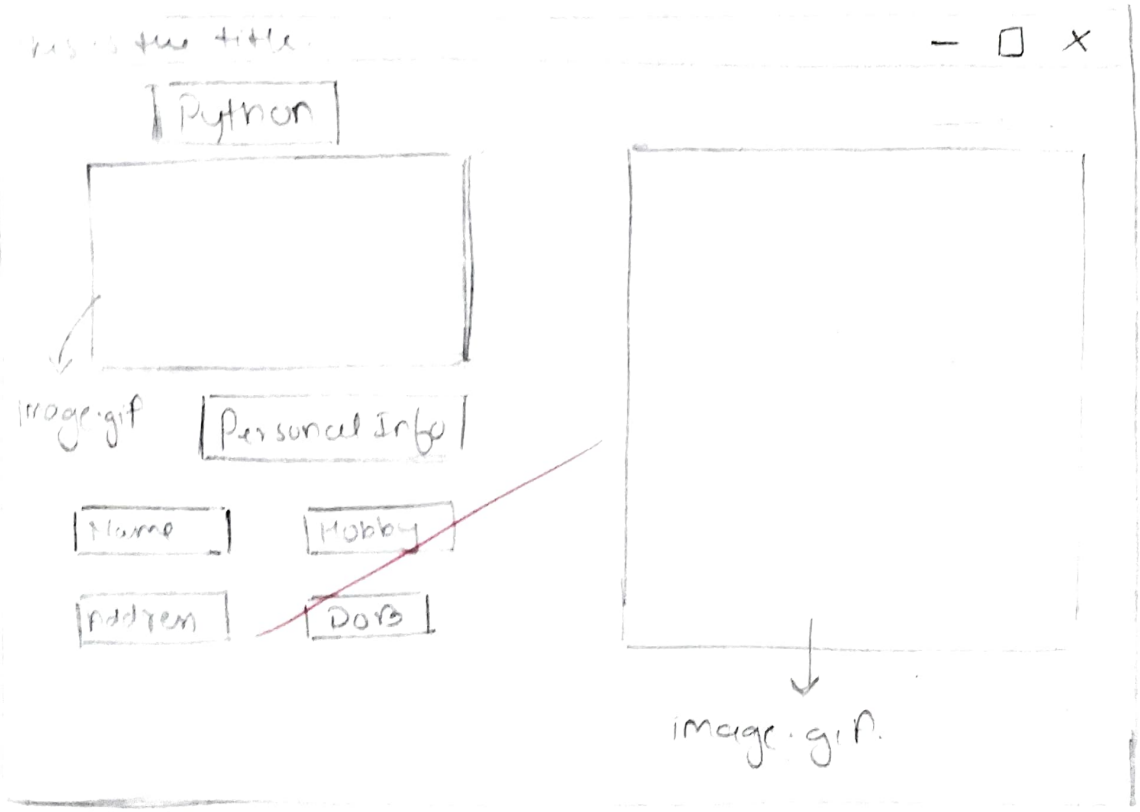
Step6: Use the subsample method with the object of the image and give the x, y co-ordinate values.

Step7: use the label method & position it onto the leftframe and placing the image after the Sampling and use the grid method for the positioning in the first row.

Step 8: Create another label object positioning it onto the rightframe & specifying the image & background attribute with row & column attribute specify it as (0,0)
Step 9: Now create a toolbar object from the Frame method & position it onto the leftframe with the height & width specified & position it onto the second row.
Step 10: Now define the various function for different toolbar options provided in the leftframe.
Step 11: From the label method position the text on the toolbar use the relief attribute & corresponding grid value & incorporate the internal padding as well.
Step 12: Create the label method position it on the toolbar with the next title as personal information & position it on same row but next column.
Step 13: Now make the use of mainloop method.

```
Button(toolbar, text="Name", height=1, width=10, command=name).
  grid(row=1, column=0)
Button(toolbar, text="Hobby", height=1, width=10, command=hob).
  grid(row=1, column=1)
Button(toolbar, text="Address", height=1, width=10, command=add).
  grid(row=2, column=0)
Button(toolbar, text="DOB", height=1, width=10, command=dob).
  grid(row=2, column=1)
root.mainloop()
```

Output:



Spinbox Widget

```
from tkinter import *
root = Tk()
S1 = Spinbox (root, from_=0, to = 50)
S1.pack(anchor=S)
root.mainloop
```

Output:

Practical - 5 (c)

Aim: GUI Components

Spinbox:

Step 1: Create an object from the tk method and subsequently create an object from spinbox method.

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

Paned window:

Step 1: Create an object from paned window method & use the pack method to make the object visible.

Step 2: Now create an object from the entry widget & place it onto the parent window & use the add method.

Step 3: Similarly, create an object from paned window and add it onto the existing window.

Step 4: Create an object using scale method & place it onto the preceding paned window & use the add method accordingly.

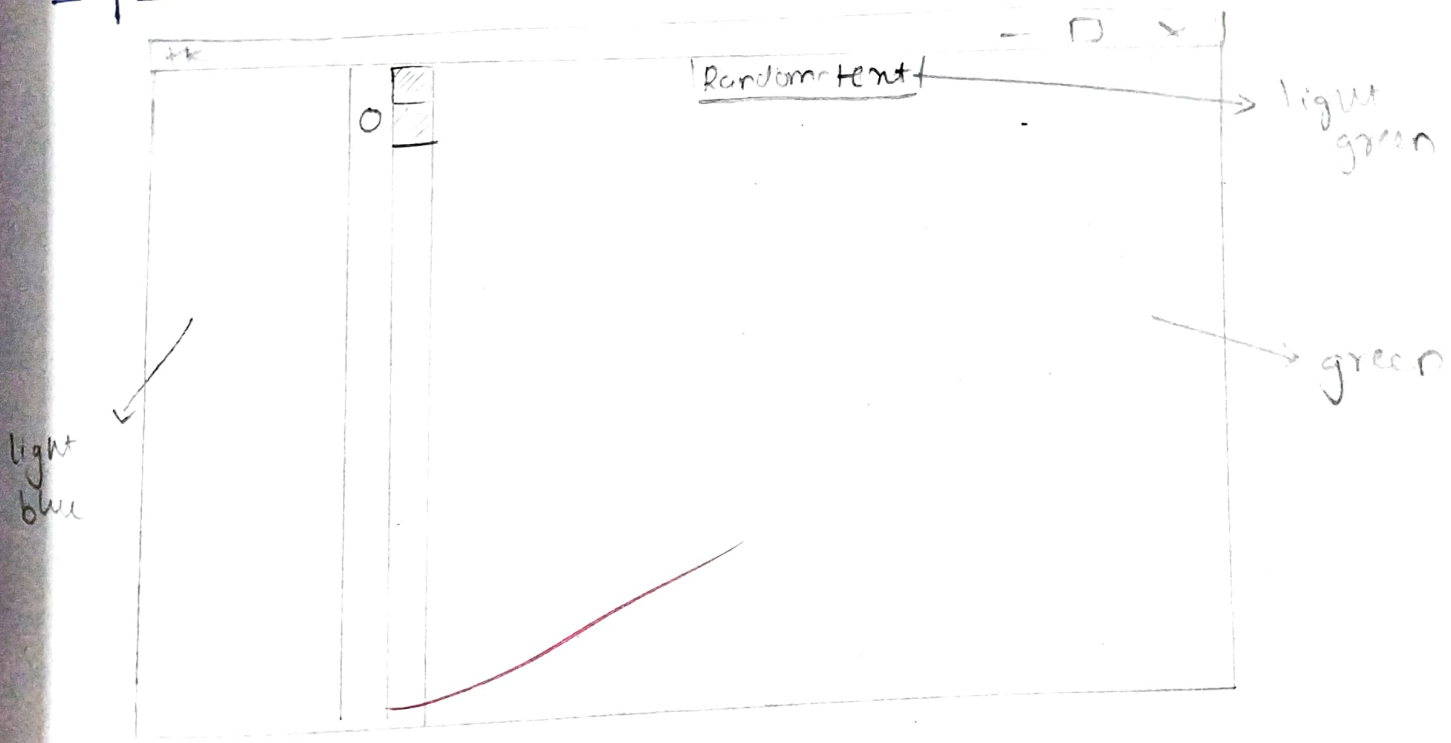
Step 5: Create a button widget, place it on paned window & def.

Step 6: Use the pack method & the mainloop method for the corresponding event to be triggered.

Paned window


```
from tkinter import *
root = Tk()
m1 = PanedWindow()
m1.pack(fill=BOTH, expand=20)
e = Entry(m1, bd=10, bg="light blue")
m1.add(e)
m2 = PanedWindow(m1, orient=VERTICAL)
m1.add(m2)
top = Scale(m2, orient=VERTICAL, bg="green")
m2.add(top)
bottom = Button(m2, text="Random text", bg="light green")
bottom.pack()
root.mainloop()
```

Output:

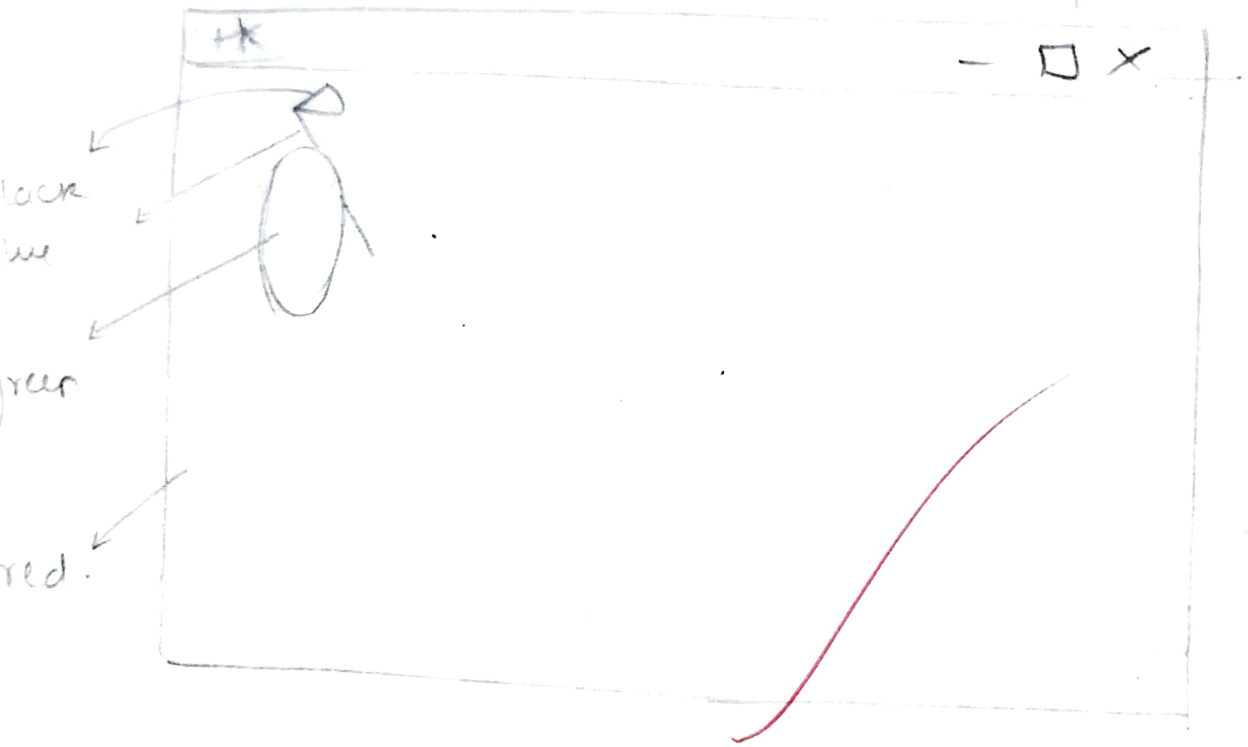


Canvas widget

8A

```
From tkinter import *
root = Tk()
c = Canvas(root, height = 1000, width = 400, bg = "red")
arc = c.create_arc(10, 20, 30, 40, start = 0, extent = 50, fill = "black")
line = c.create_line(20, 30, 35, 89, fill = "Blue")
oval = c.create_oval(15, 98, 36, 45, fill = "green")
c.pack()
```

Output:



Canvas:

Step1: Create an Object from canvas widget by using the attribute height, width, bg & parent window.

Step2: Use corresponding method for drawing the simple geometrical shapes & specify co-ordinate values.

Step3: Similarly create line & create oval method along with co-ordinate values & the fill attribute for specifying the colours.

Step4: Use the pack & mainloop method.

Practical -6

Aim: Database Connectivity

Code 1:

Step 1: Import the relevant libraries for the database connection & the operating system functionality.

Step 2: Now create an object for making the connection to the given database.

Step 3: Further create an object corresponding to the cursor area for execution of the different query statements.

Step 4: Use the object so create for implementing the structure of the database & the values within the database.

Step 5: Now with the object so created, use the insert statement for entering the values corresponding to different fields, corresponding the datatype.

Step 6: Now use the execute statement along with the object created to access the values from database.

Source Code:

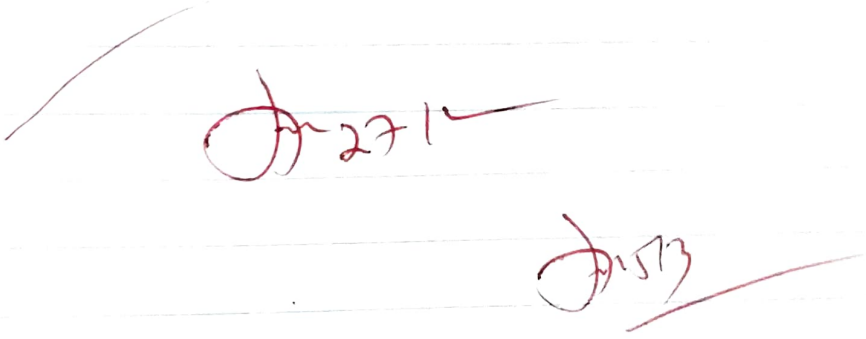
```
import os, sqlite3
conn = sqlite3.connect("employee.db")
cur = conn.cursor()
cur.execute('(create table Database (name text, Roll no- int,
 DoB date )')
cur.execute("Insert into Database values ("Vivek', 1806,
 2001-11-13)")
conn.commit()
cur.execute('Select * from Database')
print (cur.fetchall())
conn.close()
```

Output:

```
[('Vivek', 1806, 2001-11-13)]
```

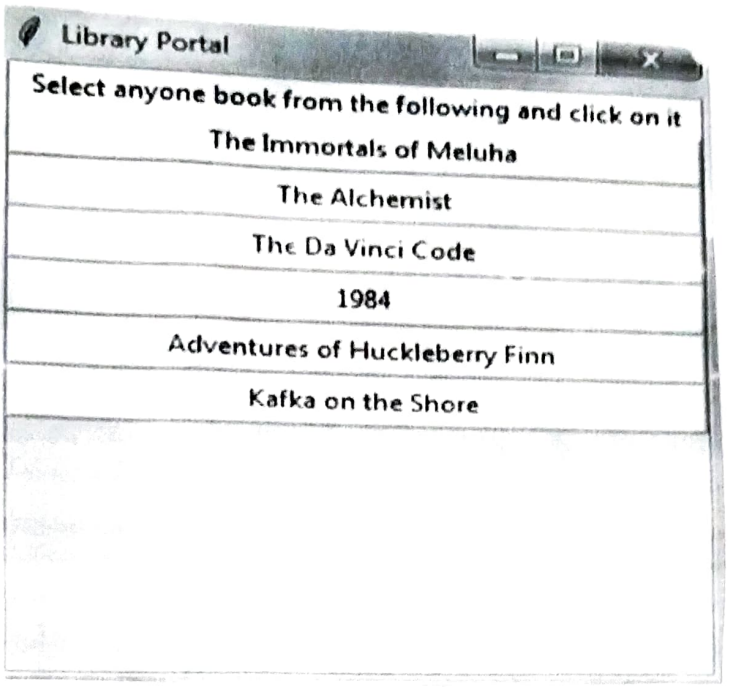
Step 7: Finally use the fetch() or fetchall() for displaying the values from the table using the object created.

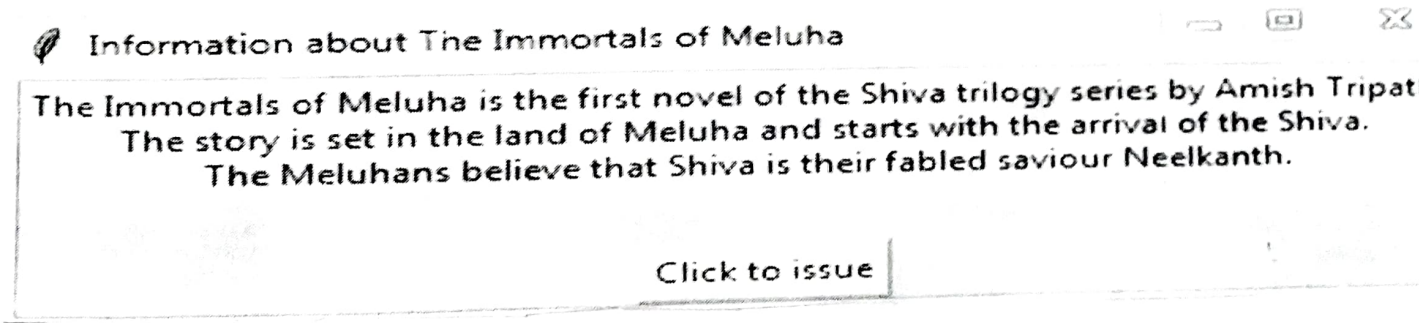
Step 8: Execute() & drop table syntax for terminating the database. & finally use close()

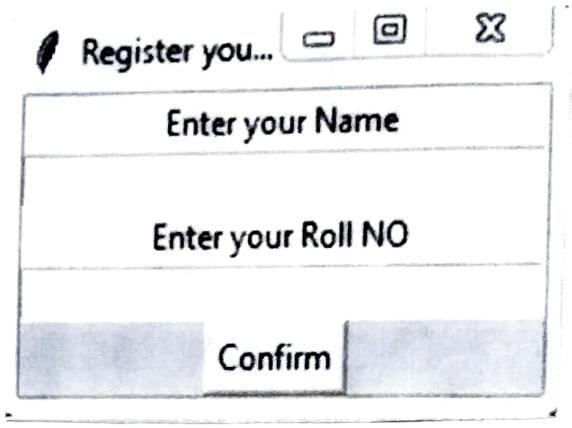


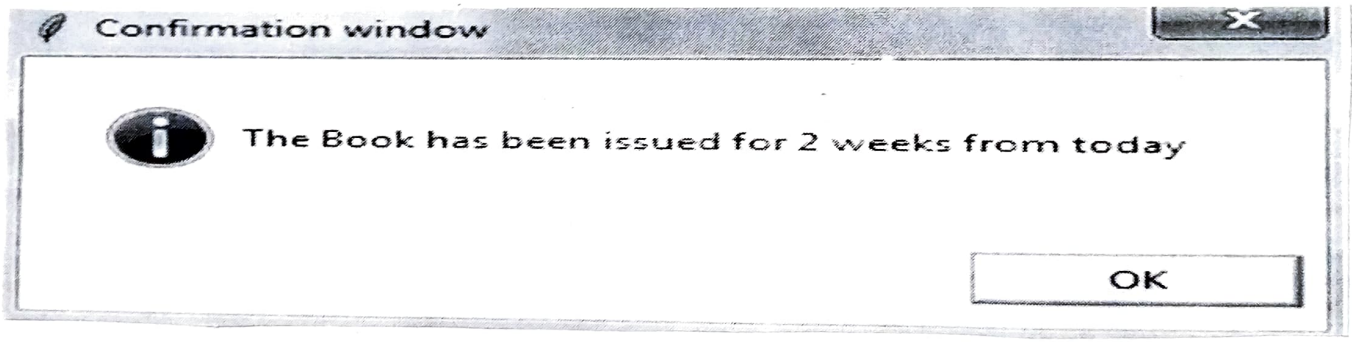
GUI Project (Library Portal)

```
from tkinter import *
from tkinter import messagebox
root=Tk()
root.config(bg="light blue")
root.title("Library Portal")
root.minsize(300,300)
def msgbx():
    messagebox.showinfo("Confirmation window","The Book has been issued for 2
weeks from today")
def book1():
    top=Tk()
    top.config(bg="light blue")
    top.minsize(100,100)
    top.title("Information about The Immortals of Meluha")
    l3=Label(top,text="The Immortals of Meluha is the first novel of the Shiva
trilogy series by Amish Tripathi.\n"
             "The story is set in the land of Meluha and starts with the arrival
of the Shiva.\n"
             "The Meluhans believe that Shiva is their fabled saviour
Neelkanth.").pack(fill=BOTH)
    b1=Button(top,text="Click to issue",command=entry).pack(side=BOTTOM)
    top.mainloop()
def book2():
    top3=Tk()
    top3.config(bg="light blue")
    top3.minsize(100,100)
    top3.title("Information about The Alchemist")
    l2=Label(top3,text="The Alchemist is a novel by Brazilian author Paulo Coelho
that was first published in 1988."
             "\nOriginally written in Portuguese, it became a widely translated
international bestseller. ").pack()
    b2=Button(top3,text="Click to issue",command=entry).pack(side=BOTTOM)
    top3.mainloop()
def book3():
    top4=Tk()
    top4.config(bg="light blue")
    top4.minsize(100,100)
    top4.title("Information about The Da Vinci Code")
    l4=Label(top4,text="The Da Vinci Code is a 2003 mystery thriller novel by Dan
Brown."
             "\nIt is Brown's second novel to include the character Robert
Langdon: the first was his 2000 novel Angels & Demons.").pack()
    b3=Button(top4,text="Click to issue",command=entry).pack(side=BOTTOM)
    top4.mainloop()
def book4():
    top5=Tk()
    top5.config(bg="light blue")
    top5.minsize(100,100)
    top5.title("Information about 1984")
    l5=Label(top5,text="Nineteen Eighty-Four: A Novel, often published as 1984,
is a dystopian novel by English novelist George Orwell."
             "\nIt was published in June 1949 by Secker & Warburg as Orwell's
ninth and final book completed in his lifetime.").pack()
    b4=Button(top5,text="Click to issue",command=entry).pack(side=BOTTOM)
    top5.mainloop()
def book5():
    top6=Tk()
    top6.config(bg="light blue")
    top6.minsize(100,100)
    top6.title("Information about Adventures of Huckleberry Finn")
    l6=Label(top6,text="Adventures of Huckleberry Finn is a novel by Mark Twain,
first published in the United Kingdom in December 1884 and in the United States
in February 1885.").pack()
    b5=Button(top6,text="Click to issue",command=entry).pack(side=BOTTOM)
    top6.mainloop()
def book6():
    top7=Tk()
    top7.config(bg="light blue")
```


Library Portal
Select anyone book from the following and click on it
The Immortals of Meluha
The Alchemist
The Da Vinci Code
1984
Adventures of Huckleberry Finn
Kafka on the Shore


Information about The Immortals of Meluha
The Immortals of Meluha is the first novel of the Shiva trilogy series by Amish Tripathi.
The story is set in the land of Meluha and starts with the arrival of the Shiva.
The Meluhans believe that Shiva is their fabled saviour Neelkanth.
Click to issue


Register you...
Enter your Name
Enter your Roll NO
Confirm


Confirmation window
The Book has been issued for 2 weeks from today
OK

```
    top7.minsize(100,100)
    top7.title("Information about Kafka on the shore")
    l6=Label(top7,text="Kafka on the Shore is a 2002 novel by Japanese author
Haruki Murakami."
             "\nIts 2005 English translation was among The 10 Best Books of 2005
from The New York Times and received the World Fantasy Award for 2006.").pack()
    b6=Button(top7,text="Click to issue",command=entry).pack(side=BOTTOM)
    top7.mainloop()
def entry():
    top2=Tk()
    top2.title("Register your Details")
    top2.config(bg="light blue")
    top2.minsize(200,100)
    l7=Label(top2,text="Enter your Name").pack(fill=BOTH)
    name=Entry(top2,bd=1).pack(fill=BOTH)
    l8=Label(top2,text="Enter your Roll NO").pack(fill=BOTH)
    name2=Entry(top2,bd=1).pack(fill=BOTH)
    b7=Button(top2,text="Confirm",command=msgbx).pack()
r1=Button(root,text="The Immortals of Meluha",fg="blue",activebackground="light
green",command=book1)
r1.pack(fill=BOTH)
r2=Button(root,text="The Alchemist",fg="blue",activebackground="light
green",command=book2)
r2.pack(fill=BOTH)
r3=Button(root,text="The Da Vinci Code",fg="blue",activebackground="light
green",command=book3)
r3.pack(fill=BOTH)
r4=Button(root,text="1984",fg="blue",activebackground="light
green",command=book4)
r4.pack(fill=BOTH)
r5=Button(root,text="Adventures of Huckleberry
Finn",fg="blue",activebackground="light green",command=book5)
r5.pack(fill=BOTH)
r6=Button(root,text="Kafka on the Shore",fg="blue",activebackground="light
green",command=book6)
r6.pack(fill=BOTH)
root.mainloop()
```