Output:

Enter 1st value = 9

Enter 2nd value = 6

1. addition
2. subtraction
3. Multiplication.
4. Division
5. Modulo Division.

Enter selection: 1.

addition = 15

Enter selection: 2

subtraction = 3

01/10/19

PRACTICAL - No. 1

Aim:- Demonstrate the use of different file accessing modes different attributes read methods.

Step I: 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 II: Now use the fileobject for finding the name of the file, the file

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

Step IV :- Now open the fileobj in write mode, write some another content. close subsequently then again open the file object in 'w+' mode. 'r+' is the update mode and write contents.

Step V :- Open fileobj in read mode and display the update written contents and close. Open again in 'r' mode with parameter passed and display the output simultaneously.

Step VI :- Now open fileobj in append mode. Open write method and write the content close the fileobj again open the fileobj in read mode and display the 'appending' output.

```
# wt mode
fileobj = open ("abc.txt","wt")
fileobj.write ("Loukik sir")
fileobj. close ()

fileobj = open ("abc.txt", "r")
print (fileobj . read ())
fileobj. close ()
```

>>> Loukik sir

```
# rt mode.
fileobj = open ("abc.txt", "rt")
s= fileobj. read ()
print (s)
fileobj. close ()
```

>>> Loukik sir.

```
# Append Mode
fileobj = open ("abc.txt", "a")
fileobj.write ("Data Structures")
fileobj. close ()
```

```
fileobj = open ("abc.txt", "rt")
Print (fileobj . read ())
fileobj . close ()
```

>>> Loukik sir Data structures.

```
# tell ()
fileobj = open ("file.txt", w")
file obj.write ("Rohit")
fileobj. close ()
fileobj = open ("file.txt","r")
printf = fileobj. tell ()
print = ("tell ():", printf)
fileobj. close ()
```

>>> 'tell():', 7.

```
# seek ()
fileobj= open ("file.txt", "r")
s = fileobj. seek (0,0)
print ("seek (0,0) is:",s)
fileobj. close
```

>>> 'seek (0,0) is:', None

Aim :- Demonstrate the use of tell() and seek() method.

Step 1 :- Open the fileobj in read mode. Declare a variable and perform fileobj to perform tell method and store the output simultaneously in variable.

Step 2 :- Use the seek method with the arguement, while opening the fileobj in read mode and closing subsequently.

10/12

3/12/19 PRACTICAL No. 2

Aim :- Iterators.

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

Step 2 :- The similar output can be obtained by using for conditional 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.

```
# iter() and next()
mytuple = ("banana", "Apple", "Pineapple")
myiter1= iter(mytuple)
print(next(myiter1))
myiter2 = iter(mytuple)
print(next(myiter1))
myiter3 = iter(mytuple)
print(next(myiter1))
```

```
>>> banana
    Apple
    Pineapple.
```

```
# for loop
mytuple1 = ("Rohit", "Studd", "Ariyay")
for x in mytuple1:
    print(x)
```

```
>>> Rohit
    Studd
    Ariyay
```

```
# Square and cube
    def square(x):
        y = x * x
        return y
    def cube(x):
        z = x * x * x
        return z
funct = [square, cube]
```

```
for r in range(5):
    value = list(map(lambda x: x(r), [pow]))
    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))   >>> [0,4,5,7,9,11,13,15,20,19,25]
# odd numbers.
class odd:
    def __iter__(self):
        self.num = 1
        return self
    def __next__(self):
        num.self = num
        num.self += 2
    def __next__(self):
        num = self.num
        self.num += 2
        return num
```

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.

I/O Error:

```
try:
    fo = open ("abc.txt", "w")          → #R
    fo.write ("Python is an intendent language.")
except IOError:
    print ("Enter appropriate mode for file operation!")
else:
    print ("Operation is successful!")
```

>>> Operation is successful!

R output :-

>>> Enter appropriate mode for file operation!

Value Error :-

```
try:
    X = int (input("Enter a statement :"))
except ValueError:
    print ("ARITHMETIC ERROR!")
else:
    print ("Operation is successful!")
```

>>> Enter a statement : abc
ARITHMETIC ERROR!
>>> Enter a statement : 123
Operation is successful!

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

Step 8:- For the exception to be raised use the except keyword (and) i.e. TypeError print Incompatible values.

Step 9:- Use except with error of zero division error and print the message according that is if entered number is zero not able to perform operation!

Step 10:- Declare static variables and values.

Step 11:- For multiline exception use the error types by separating them with a comma.

```
# using except keyword
  try:
      a=open ("abc.txt", "w")
      a.write (" python")
  except IOError:
      print (" Error!")
    else:
      print successful
    def x():
        l=[ ]
        print (len(L))
    def y():
        li=[2,4,4,1]
        print (len(li))
    print (x())
    print (y())
```

Output: successful
0
None
4
None.

```
# raise Keyword:
  try:
    a=int(input("Enter a number:"))
    raise value Error
  except value Error:
    print ("Enter integer value!")
```

>>> Enter: xyz

Enter integer values!

24/12/19 IE

PRACTICAL NO: 4

TOPIC :- REGULAR EXPRESSION.

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

Step 2 :- Import re module declare pattern with literal and meta character. Declare string value. Use the findall () with arguments and print the same.

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

```
#no_space:
  import re
  string = 'abc def ghi'
  pattern = r'\s+'
  replace = ''
  v1 = re.sub (pattern, replace, string)
  print (v1)
>>> abcdefghi

# group ()
  import re
  sequence = 'python is an interesting language'
  v= re. search ('\Apython', sequence)
  print (v)
  V1= v.group()
  print (v1)
>>> <_sre. SRE_ Match object at 0X02810F00>
    python
# verifying the given set of phone numbers.
  import re
  list1=['8004567891', '9920982913', '9167563272', '9076269712']
  for value in list1:
    if re. match (r' [8-9] {1} [0-9] {9}',
      value or len (value)==10):
      print ("Criteria matched for cell number!")
  else:
    print ("Criteria failed!")
>>> Criteria matched for cell number!
  Criteria matched for cell number!
  Criteria failed!
  Criteria matched for cell number!
```

Step 4 :- Import re module declare string and accordingly declare pattern replace the blank space with no_space . Use sub () with 3 argument and print the string without spaces.

Step 5 :- Import re module declare a sequence use search method for finding subsequently use the group () with dot operator as search() gives memory coration using group () it will show up the matched string.

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

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

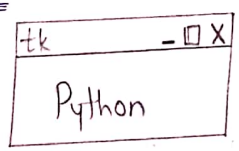
Step 8:- Import re module declare the host and domain name declare pattern for separating the host & domain name. Use the findall() and print the output respectively.

Step 9:- Import re module enter a string use pattern to display only two elements of the particular string use findall() declare two variable with initial value as zero use for condition and subsequently use the if condition check whether condition satisfy add up the or else increment value and display the values subsequently.

Creation of parent Window :-

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

Output :-



2 :-

```
from Tkinter import *
root Tk()
l = label (root, text = "Python")
l. pack()
l1 = label (root, text = "CS!", bg = "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)
```

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

1) side = LEFT, padx = 20
2) side = LEFT, pady = 30
3) side = TOP, ipadx = 40
4) side = TOP, ipady = 50

Step 4: Use the mainloop() for the triggering of the corresponding events.

Step 5: Now repeat above steps with the label() which takes the following arguments

1) Name of the parent window.
2) Text attribute which defines the string.
3) The background color (bg).
4) The foreground (fg) and than use the pack() with a relevant padding attributes.

PRACTICAL NO. 5(B)

Aim :- Gui Components.

#1 :

step 1 : Import the relevant methods from the tkinter library, create an object with the parent window.

Step 2 : Use the parent window object along with the geometry() declaring specific pixel size of the parent window.

Step 3 : Now define a function which tells the user about the given selection made from multiple option available.

Step 4 : Now define the parent window and define the option with control variable.

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

Step 6 : Create an object from radiobutton which will take following arguments.
parentwindow object, text variable which will take the values option no. 1,2,3,.....
variable arguments, corresponding value & trigger the function declared.

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.

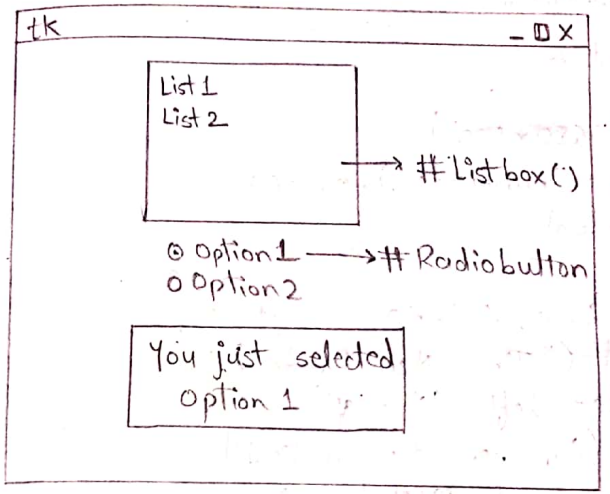
Step 3 : Use the geometry() for laying of the window.

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

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

Step 6 : Use the mainloop with the parent object.

Output :-

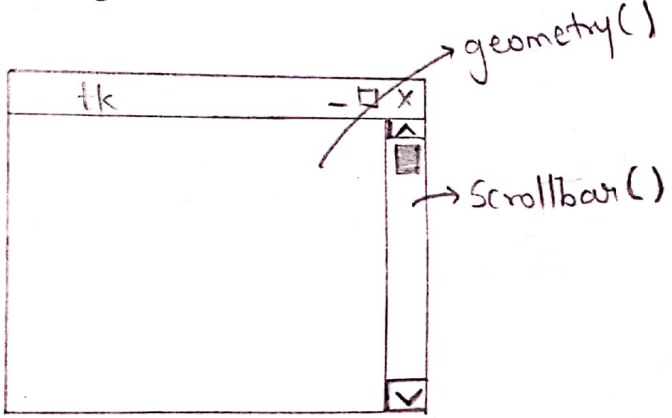


#2 :

Scrollbar() :

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

Output :

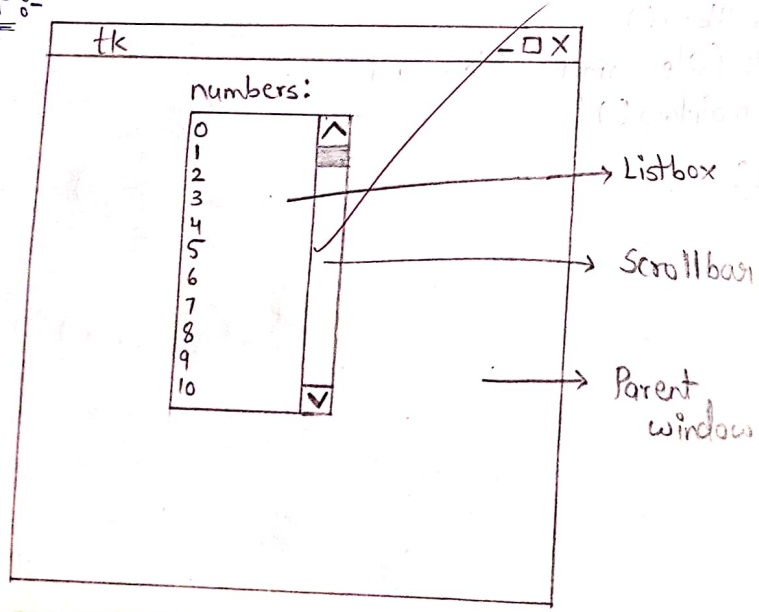


#3:

Using frame widget:

```
from tkinter import *
window = Tk()
window.geometry("680x500")
Label(window, text = "numbers:").pack()
frame = Frame(window)
frame.pack()
listNodes = Listbox(frame, width = 20, height = 20,
                    font = ("Times New Roman", 10))
listNodes.pack(side = "left", fill = 'y')
Scrollbar = scrollbar(frame, orient = "vertical")
Scrollbar.config(command = listNodes.yview)
Scrollbar.pack(side = "right", fill = "y")
for X in range(100):
    listNodes.insert(END, str(X))
window.mainloop()
```

Output:-



#3 :-

Step 1 : Import the relevant libraries from the tkinter method.

Step 2 : Create an corresponding object of the parent window.

Step 3 : Use the geometry manager with pixel size (680 x 500) or any other suitable pixel value.

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

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

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

Step 7 : Use the scrollbar() with an object use the attributes of vertical then configure the same with object created from the scrollbar() & use pack()

Step 8 : Trigger the events using mainloop.

#4:

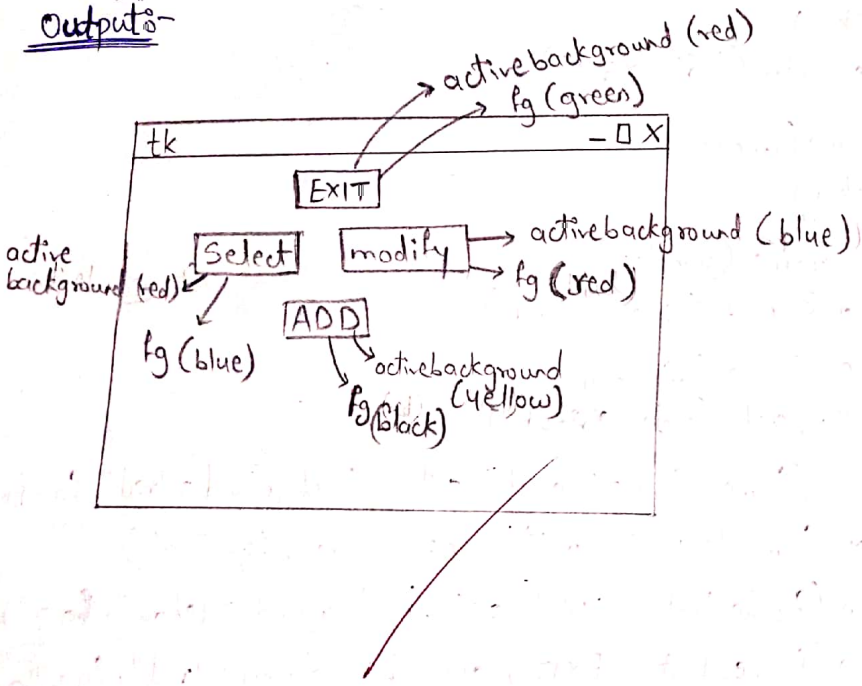
```
from tkinter import *
window= Tk()
window.geometry("680x500")

frame =Frame(window)
frame.pack()

leftframe = Frame(window)
leftframe.pack(side = "left")
rightframe = Frame(window)
rightframe.pack(side ="right")
b1= Button(frame, text ="select", active background ="red", fg="blue")
b2= Button(frame,text = " modify", bg = "blue", fg= "red")
b3 =Button(frame,text =" ADD", active background ="yellow", fg ="black
b4 =Button(frame, text= "EXIT", active background ="red", fg = "gree

b1. pack(side ="LEFT", padx =20)
b2. pack(side = "right", padx = 30)
b3. pack(side =" bottom ", pady =20)
b4. pack(side ="top")
```

Output:-



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

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

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

2:11

PRACTICAL NO. 5(C)

Aim :- Gui Components.

Step 1 :- Import the relevant methods from tkinter library.
Step 2 :- Import tkmessagebox
Step 3 :- Define a parent window object along with the parent window.
Step 4 :- Define a function which will use tkmessagebox with showinfo method along with info window attribute.
Step 5 :- Declare a button with parent window object along with the command attribute.
Step 6 :- place the button ~~window~~ widget onto the parent window and ~~display~~ finally call mainloop() for triggering of the events called above.

```
# 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 Interest: \n
    Golden Gate Bridge \n Lombard street \n Chinatown \n coit Tower")
    L.pack ()
    b1 = Button (top, text = "next", command = second)
    b1.pack (side = RIGHT)
    b2 = Button (top, text = "exit", command = terminate)
    b2.pack (side = LEFT)
top.mainloop ().
```


Step 1 :- Import the relevant methods from the tkinter library along with parent window object declared.

Step 2 :- Use parent window object along with minsize function for window size.

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

Step 4 :- Similarly, define the function second and use the attributes accordingly.

Step 5 :- Declare another function button along with parent ~~window~~ object and declare button with attributes like FLAT, RIDGE, GROOVE, RAISED, SUNKEN along with the relief widget.

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

```
def second():
    top2=Tk()
    top2.config(bg="Orange")
    top2.title("About us!")
    top2.minsize(300,300)
    L=Label(top2, text="Created by: Rohit Gupta \n
    for more details contact to our official account")
    L.pack()
    b3=Button(top2, text="prev", command=main)
    b3.pack(side=LEFT)
    b2=Button(top2, text="exit", command=terminate)
    b2.pack(side=RIGHT)
    top2.mainloop()
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()
```


PRACTICAL No. 5 (D)

Aim :- GUI COMPONENTS.

Step 1 :- Import relevant methods from the tkinter library

Step 2 :- Create parent window object and use the config method along with background colour attribute specified.

Step 3 :- Define a function finish with the message box widget which will display a message i.e. a warning message and subsequently terminate the program.

Step 4 :- Define a function 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 attribute.

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

Step 6 :- Use photoimage widget with file_and filename with gif attribute.

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

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

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

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

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


```
from Tkinter import *
root = Tk()
root = config(bg = "grey")
def finish():
    messagebox.askokcancel("Warning", This will end the program")
    quit()
def info():
    list1 = Listbox()
    list1.insert(1, "Co. Name : Apple")
    list1.insert(2, " Products : iphone")
    list1.insert(3, " Language : swift")
    list1.insert(4, " OS: Ios")
    list1.grid(ipadx = 30)
def aboutus():
    list2 = label(text = " About us")
    list2.grid(ipadx = 30)
    list3 = Label(text =" steve Jobs theatere March 2020")
    list3.grid(ipadx = 24)
p1 = Photoimage(file = " download.gif ")
f1 = frame(root, height = 35, width = 5)
f1.grid(row=1, column = 0)
f2=frame(root, height = 250, width = 500)
f2.grid(row = 1, column = 1)
p2 = p1.subsample(5,4)
l1 = Label(f1, image = p2, relief = FLAT)
l1.grid(row= 1, column = 0, padx = 20, pady = 15)
l2 = Label(f2, image = p1, relief = SUNKEN)
l2.grid(padx = 25, pady = 10)
b1 = Button(f1, text = " Information ", relief = SUNKEN, command = info)
b1.grid(row = 1, column = 0)
```

PRACTICAL NO. 5 (E)

Aim :- GUI COMPONENTS

Algorithm :-
#spinbox & paned-window.

Step 1 :- Create an object from ~~tkinter~~ tk method subsequently create an object from spinbox method.

Step 2 :- Make the obj so created on to parent window and trigger the corresponding events.

Step 1 :- (paned window) Create an object from paned window method and use the pack method with attribute 'fill' and 'expand'.

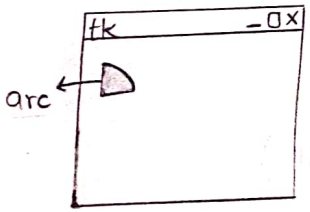
Step 2 :- Create an object from the label method and put it on to the paned window with text attribute and use the add method to embed the new object

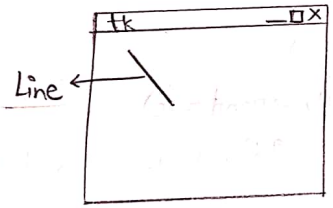
Step 3 :- Similarly, create a second paned window object & add it on to the first paned window.

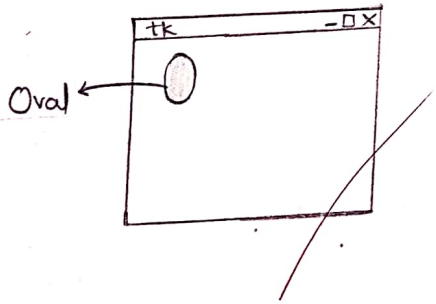
Canvas Program :-

```
from tkinter import *
root = Tk()
root.config(bg="royal blue")
C = Canvas(root, height = 400, width = 400, bg="orange")
coord = 10,20, 30,40
Carc = C.create_arc(coord, fill = 'purple')
Cline = C.create_line(coord, fill = 'red')
Coval = C. create_oval(coord, fill = 'pink')
C.pack()
mainloop()
```

Output :-







Step 1 :- Create an object from the canvas method and use the attribute height, width, colour and the parend window object.

Step 2 :- Use the method to create line, create oval & create arc along with the canvas object so created and co-ordinate value.

Step 3 :- Similarly, use the other method and call the pack method and the mainloop method.

PRACTICAL NO. 6

Topic :- Database Connectivity.

Algorithm :-

Step 1 :- Import dbm library and use the open method for creating the database by specifying the name of the database along with the corresponding flag.

Step 2 :- Use the object so created for excessing the given website & the corresponding regular name for the website.

Step 3 :- Check whether the given URL address with the regular name of the page is not equal to ~~none~~ none then display the message that the particular URLs is found else not found.

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

Code :-

```
import os, sqlite3
connection = sqlite3.connect("Customer.db")
cursor1 = connection.cursor()
cursor1.execute("create table customer(name char(10), ID int(5))")
cursor1.execute("insert into customer values('Rohit', 1629),
                                             ('Himanshu', 2916)")
connection.commit()
cursor1.execute("select * from customer")
A = cursor1.fetchall()
for i in A:
    print(i)
connection.close()
```

Output :-

```
>>> [('Rohit', 1629), ('Himanshu', 2916)]
```


sqlite

Step 1 : Import the corresponding library for making the database correction which are os, sqlite3.

Step 2 : Now create the connection object using sqlite3 library & connect method for creating the new database.

Step 3 :- Now create the cursor object using the cursor method from the connection object created in the earlier step.

Step 4 :- Now we the execute method for creating the table with the column name and the respective datatype.

Step 5 :- Now with the cursor object use the insert statement for entering the values corresponding to the different fields considering the datatype.

Step 6 : Use the commit method to complete the transaction using the connection object.

Step 7 : Use the execute statement along with cursor object from excessing the values from the database using the select / from / where clause.

Step 8 :- Finally use the fetch method for displaying the values from the table using the cursor object.

Step 9 :- Use the execute method & the drop table syntax for terminating the database & finally use the close method.