1.a : year, name age

```
name=input("what is your name?")
age=int(input("enter your age:"))
year=str((2023-age)+100)
print(name+"you will be 100 years old in the year"+year)
```

1.b : even or odd

```
a=int(input("Enter any number:"))
if a % 2 ==0 :
   print("it's even number")
else:
   print("it's odd number")
```

1.c : Fibonacci series

```
v=int(input("Enter number of series"))
a=0
b=1
print(a)
print(b)
for n in range(0,v):
   if (n <= 1):
      c=n
   else:
         c=a+b
         a=b
         b=c
         print(c)
```

1.d : reverse number.

```
def revnum (num):
    sum=0
    while num != 0:
     rem = num % 10
     sum = rem + sum * 10
     num //= 10
    print(" reverse number:",sum)
num=int(input("enter any number:"))
revnum(num)
```

1.e : Armstrong number and palindrome

```
def armnum(num):
  sum=0
  temp=num
  while temp>0:
    digit=temp%10
    sum+=digit**3
    temp//=10
  if num==sum:
    print(num,"is an Armstrong number")
  else:
    print(num,"is not an Armstrong number")




def palnum(num):
  sum=0
  temp=num
  while num!=0:
    rem=num%10
    sum=rem+sum*10
    num=num//10
  if temp==sum:
    print(temp,"is a palindrome number")
  else:
    print(temp,"is not a palindrome number")
num=int(input("Enter any number:"))
armnum(num)
palnum(num)
```

1.f : factorial number.

```
def fact(x):
    if x==1:
     return 1
    else:
     return(x* fact(x-1))
num=int(input("enter any number:"))
print("the factorial of",num,"is",fact(num))
```

2.a : vowels

```
def vchk(ch):
    if(ch=='a' or ch== 'A' or ch== 'e' or ch=='E' or ch=='i' or ch=='I' or ch=='o' or ch=='O' or ch=='u'
or ch=='U'):
      print(ch,"is a vowel")
    else:
      print(ch, "is a consonant")

ch=input("Enter a single character(a-z/A-Z) only:")
vchk(ch)
```

2.b : lenghth of string

```
def calen(n):
   count=0
   for i in n:
      count =count+1
   return count
print(calen([1,3,5,7]))
print(calen("student"))
```

2.c : Historam (*)4,7,9

```
def histogram(items):
   for n in items:
      output=' '
      times=n
      while(times>0):
         output+='*'
         times=times-1
      print(output)
histogram([4,9,7])
```

3.a : pangram

```
import string,sys
def ispangram(str1,alphabet=string.ascii_lowercase):
   alphaset=set(alphabet)
   return alphaset <=set(str1.lower())
print(ispangram("the quick brown fox jumps over the lazy dog"))
```

3.b : Take a list and print out all element of the list that are less tham 5.

```
A=[1,1,2,3,5,8,13,21,34,55,89]
a=[1,1,2,3,5,8,13,21,34,55,89]
for i in a:
   if i<5:
      print (i)
```

4.a : Take 2 list and return true if they have atleast 1 common number.

```
def find_common(st1,st2):
  res = False
  for x in st1:
    for y in st2:
     if x==y:
       res = True
       return res
print(find_common([4,6,7],[4,3,13]))
print(find_common([10,9,8],[5,6,14]))
```

4.b : specified list after removing 0th,2nd,4th and 5th elements.

```
name=['Tanisha','Vaishnavi','Rani','Trupti','Kumari']
name=[x for (i,x) in enumerate(name) if i not in (0,2,4,5)]
print(name)
```

4.c : clone or a copy list.

```
L1=[8,11,13,22,27]
L2=list(L1)
print("L1:",L1)
print("L2:",L2)
```

5.a : Write a program to sort(ascending or descending)a dictionary by value.

```
import operator
d={1:22,3:13,4:8,2:11,0:27}
print(d)
t=sorted(d.items(),key=operator.itemgetter(0))
print("in ascending order by value:",t)
t=sorted(d.items(),key=operator.itemgetter(0),reverse=True)
print("in ascending order by value:",t)
```

5.b : concatenate dictonary to create a new 1.

```
dic1 = {1:10,2:20}
dic2 = {3:30,4:40}
dic3 = {5:50,6:60}
dic1.update(dic2)
dic1.update(dic3)
print(dic1)
```

5.c : sum all the items in a dictonary.

```
d={'t1':10,'t2':5,'t3':25}
print(d)
print("sum:",sum(d.values()))
```

6.a : Read an entire text file.

```
f = open('abc.txt','r')
t = f.read()
print(t)
f.close()
```

6.b : Append text to a file and display the text.

```
f = open('abc.txt','a+')
f.write('\n Easy to learn')
f = open('abc.txt','r')
t = f.read()
print(t)
f.close()
```

6.c : to Read last n lines of a file.

```
f = open('abc.txt','r')
t = f.readlines()
print(t[-1])
f.close()
```

7.a : Design a class store the info of a student & display the same.

```
class student:

 def info(self, studname, studaddr):

  print("Name:",studname," Address : ",studaddr)

obj= student()

obj.info('Veda','Airoli').
```

7.b : Implement the concept of inheritance.

```
class st:
    def s1(self):
        print("Base Class")
class st1(st):
  def s2(self):

    print("Derived Class")

t=st1()

t.s1()

t.s2()
```

7.c : Create a class called Numbers, which has a single class attribute called MULTIPLIER, and a constructor which takes the parameters x and y (these should all be numbers).

i) Write a method called add which returns the sum of the attributes x and y.

ii) Write a class method called multiply, which takes a single number parameter a and returns the product of a and MULTIPLIER. iii)

Write a static method called subtract, which takes two number parameters, b and c, and returns b-c.

iv) Write a method called value which returns a tuple containing the values of x and y. Make this method into a property, and write a setter and a deleter for manipulating the values of x and y.

->

```
class Numbers:
  MULTIPLIER = 3.5
  def _init_(self, x, y):
    self.x = x
    self.y = y
  def add(self):
    return self.x + self.y
  @classmethod
  def multiply(cls, a):
    return cls.MULTIPLIER * a
  @staticmethod
  def subtract(b, c):
    return b - c
  @property
  def value(self):
    return (self.x, self.y)
  def set_value(self, x, y):
    self.x = x
    self.y = y
  def del_value(self):
    del self.x
    del self.y
obj1 = Numbers(10, 20)
print("add() output=", obj1.add())
print("multiply() output=", Numbers.multiply(10))
print("subtract() output= ", Numbers.subtract(10, 5))
print("property output=", obj1.value)
print(obj1.del_value())
print(obj1.set_value(100, 200))
print("property output=", obj1.value)
```

8.a : Write the program for the following:

a) Open a new file in IDLE ("New Window" in the "File" menu) and save it as geometry.py in the directory where you keep the files you create for this course. Then copy the functions you wrote for calculating volumes and areas in the "Control Flow and Functions" exercise into this file and save it.

Now open a new file and save it in the same directory. You should now be able to import your own module like this:

```
import geometry
```

Try and add print dir(geometry) to the file and run it.

Now write a function pointyShapeVolume(x, y, squareBase) that calculates the volume of a square pyramid if squareBase is True and of a right circular cone if squareBase is False. x is the length of an edge on a square if squareBase is True and the radius of a circle when squareBase is False. y is the height of the object. First use squareBase to distinguish the cases. Use the circleArea and squareArea from the geometry module to calculate the base areas.

->

```
#geometry.py
import math


def sphereArea(r):
    return 4* math.pi*r**2


def sphereVolume(r):
    return 4* math.pi*r**3/3


def sphereMetrics(r):
    return sphereArea(r), sphereVolume(r)


def circleArea(r):
    return math.pi*r**2


def squareArea(x):
    return x**2


#demo.py
import geometry


def pointyShapeVolume(x, h, square):
    if square:
        base= geometry.squareArea(x)
    else:
        base = geometry.circleArea(x)
    return h* base / 3.0
print (dir(geometry))
print (pointyShapeVolume(4,2.6, True))
print (pointyShapeVolume(4, 2.6, False))
```

8.b : Write a program to implement exception handling.

```
try:
    num=int(input("Enter the number"))
    re=100/num
except(valueError,ZeroDivisionError):
    print("something is wrong")
else:
    print("result is",re)
```

9.a : `Write the program for the following:

a) Try to configure the widget with various options like: bg="red", family times", size=10

Explanation: In following program we have taken one label and three button widgets. on every button click we have changed three options of label as text, bg and font with size. For this we have written three functions in which with the help of widgets config() method we have changed option values of label and achieved our desired result.

Solution:

```
import tkinter as th win th.Tk) win.title("Practical 9 A")
def redClick():
label.config(text="Helvetica Font") label.config(bg="red") label.config(font("Helvetica". 16))
def greenClick()
label.config(text="Cambria Font")
label.config(bg="green") label.config[font=("Cambria", 18])
def yellowClick():
label.config(texts"Arial Four")
label.config(b="yellow") label.config[font="Calbari", 14))
label = tk Label(win, text="Practical 9-A".bg="white")
81= tk.Button(win, text="RED Click", relief raised command-redClick)
B1 packside "left")
label.pack()
82 tk.Button(win, text="Green Click", relief'raised',command-greenClick)
B2 pack(side "left")
83 = tk.Button(win, text-"Vellow Click", relief-rated command-yallowClick)
83 pack side = "left")
win.mainloop()
->
import tkinter as tk
win=tk.Tk()
win.title("Practical 9 A")
def redClick():
  label.config(text="Helvetica Font")
  label.config(bg="red")
  label.config(font("Helvetica",16))
def greenClick():
  label.config(text="Cambria Font")
  label.config(bg="green")
  label.config(font=("Cambria", 18))
def yellowClick():
  label.config(text="Arial Font")
  label.config(bg="yellow")
  label.config(font=("Calbari",14))
label = tk.Label(win, text="Practical 9 A",bg='white')
label.pack()
B1= tk.Button(win, text="RED Click", relief ='raised', command=redClick)
B1.pack(side="left")
```

```
B2=tk.Button(win, text="Green Click", relief='raised',command=greenClick)

B2.pack(side="left")

B3=tk.Button(win, text="Yellow Click", relief='raised', command=yellowClick)

B3.pack(side = "left")

win.mainloop()
```

9.b : Try to change the widget type and configuration options to experiment with other widget types like Message, Button, Entry, Checkbutton, Radiobutton, Scale etc.

Explanation: In following example we have created a GUI with dynamically changing

the widget type as well as options of the widget. On selection of the checkbox we have changed the entry box into the menu as well as changed the attributes of label, such as font type, colour and font face. For achieving this, we have created the changeable widgets inside the es except the button. We have created a swap() method to achieve all the changes and configuration. Here we have made use of widgets pack forget() method to toggle the wwidgets.

->

```
from tkinter import*


def swap():


 if v.get():


   e.pack_forget()


   mb.pack(anchor="w", side="right")


   l2.config(text="Use Menu Below.")


   l2.config(bg="yellow")


   l2.config(font=("Helvetica", 16, "italic"))
 else:

   mb.pack_forget()

   e.pack(anchor="w", side="left")

   l2.config(text="Use Entry Box Below.")

   l2.config(bg="green")

   l2.config(font=("Cambria",16,"bold"))

   e.focus()

t=Tk()

v=IntVar(t)

c=Checkbutton(t, command=swap, text="Select to use menu.", variable=v)

c.pack(anchor="w")

f1=Frame(t)

l1 = Label(f1, text="Select the Menu item of your choice:")

l1.pack(side="left")

l2 = Label(f1, text="Use Entry Box Below.",bg="green", font=("Cambria", 16, "bold"))
```

```
l2.pack(side="top")

f=Frame(f1)

f.pack(side="left")

e=Entry(f,width=35)

mb = Menubutton(f, width=25, text="Veg", indicatoron=1, relief="sunken", anchor="w")


m=Menu(mb, tearoff=0); mb.configure(menu=m)

for s in "Veg nonVeg Chinese French".split():

 m.add_command(label=s, command=lambda s=s: mb.configure(text=s))
f.pack()
f1.pack()
swap()
b=Button(t, text="place order",relief="raised",fg="red",command=t.destroy);
b.pack(side="top")
f.mainloop()
```

10.a : Design the database applications for the following:

a) Design a simple database application that stores the records and retrieve the same.

Explanation: In following example we have created a GUI which will accept the details of the books and it stores it in database on the click of insert button. When we press the show books button it displays the information stored in the database using list box.

Solution:

Step 1: First we need to create MySQL database and database table which is required for our application as follows:

```
Create database Library;

Use Library;

CREATE TABLE books (id int primary key auto_increment, title VARCHAR(50), author VARCHAR(30), publisher VARCHAR(25), year VARCHAR(10), edition VARCHAR(10));
```

->

```
from tkinter import*

from tkinter import messagebox

import mysql.connector as mysql


conn = mysql.connect(user='root', password='root',host='127.0.0.1')

cursor = conn.cursor()

cursor.execute("USE Library")

def insertCall():

   sql = "INSERT INTO books (title,author, publisher,year,edition) 
VALUES('"+bTitle.get()+"','"+aName.get()+"','"+pub.get()+"',"+year.get()+", '"+edi.get()+"')"

   cursor.execute(sql)

   conn.commit()


   bTitle.delete(0,END)

   aName.delete(0,END)

   pub.delete(0,END)

   year.delete(0,END)

   edi.delete(0,END)

   bTitle.focus()

   messagebox.showinfo(title='Confirmation', message="Information Inserted.")

def showBooks():

   sql = "SELECT*FROM books"

   cursor.execute(sql)

   results=cursor.fetchall()
```

```
    lb=label(win ,text="NO.--NAME--AUTHOR--PUBLISHER--YEAR--EDITION",width=70)
    lbl.grid(row=6, columnspan=2)

    listbox = Listbox(win, width=70)
    listbox.grid(row=7, columnspan=2)
    for row in results:
        listbox.insert(END, row)
    conn.commit()

win = Tk()
win.title("Book Details")

Label(win, text ="Book Title:").grid(row=0)
Label(win, text ="Author Name").grid(row=1)
Label(win, text ="Publisher").grid(row=2)
Label(win, text ="Year").grid(row=3)
Label(win, text ="Edition") .grid(row=4)

bTitle =Entry(win, width=30)
bTitle.grid(row=0, column=1)
aName=Entry(win,width=30)
aName.grid(row=1, column=1)
pub=Entry(win,width=30)
pub.grid(row=2, column=1)
year =Entry(win,width=30)
year.grid(row=3, column=1)
edi=Entry(win,width=30)
edi.grid(row=4, column=1)

b1=Button(win, text='Insert',command=insertCall)
```

```
b1.grid(row=5, column=0, padx=24)
b2-Button(win, text='Show Books', command=showBooks)
b2.grid(row=5, column=1, sticky=W, padx=38)
mainloop()
```

10.b : Design a database application to search the specified record from the database.

Explanation: In following program we have taken one entry box, five radio button in group and one button. On the click of button as per the radio button selected we are searching the attributes of the book. The results are displayed in the list. If no result found it displays the message box. We have used here MySQL like clause to retrieve the results. Note: To run below example we have used the same database that we have created for example "10) A" above.

->

```
#10B.py
from tkinter import*


from tkinter import messagebox
import mysql.connector as mysql



def showBooks():


  conn=mysql.connect(user='root', password='root',host='127.0.0.1')


  cursor = conn.cursor()


  cursor.execute("USE Library")


  sql = "SELECT* FROM books where "+var.get()+" like '%" +stext.get()+"%"


  cursor.execute(sql)


  results=cursor.fetchall()
  if not results:

     messagebox.showinfo(title='Sorry', message="No Record Found!")

  else:

   lbl=Label(win, text= "Search Results")

   lbl.grid(row=7, columnspan=2)

   listbox=Listbox(win, width=70)

   listbox.grid(row=8, columnspan=2)

   for row in results:  # Ietrating and assigning values to list
     listbox.insert(END, row)

   conn.commit()# 6: Making the changes permanánt in the database

win = Tk()

win.title("Book Search")



var=StringVar()

var.set("title")
Label(win, text="Select the Attribute to Search:").grid(row=0)
```

```
r1=Radiobutton(win, text="Search by Title",width=20, justify="center", variable=var,
value="title",anchor="w")

r1.grid(row=1,columnspan=2)

r2= Radiobutton(win, text="Search by Author",width=20, justify="center", variable=var,
value="author" ,anchor="w")

r2.grid(row=2,columnspan=2)

r3= Radiobutton(win, text="Search by Publication", width=20, justify="center", variable=var,
value="publisher",anchor="w")

r3.grid(row=3, columnspan=2)

r4= Radiobutton(win, text="Search by Year", width=20, justify="center", variable=var,
value="year", anchor="w")

r4.grid(row=4,columnspan=2)

l1=Label (win, text="Enter the text string to Search:")
l1.grid(row=5,column=0)

stext=Entry(win, width=20)

stext.grid(row=5,column=1)

b1=Button(win, text='Search Books', command=showBooks).grid(row=6,columnspan=2)

mainloop()
```

10.c : Design a database application to that allows the user to add, delete and modify the records.

Explanation: In following program we used a text entry to first retrieve the record for update. Once we have displayed the records by using callvalues() method. After editing the information user can click Update Record button to make the changes in database by using updateCall() method. Also for deletion of a data entry user have to click Delete Record button by using deleteCall() method. If the particular record for updating or deleting is not found then user is notified with the message.

->

```
from tkinter import *

from tkinter import messagebox

import mysql.connector as mysql


conn = mysql.connect(user='root', password='root',host='127.0.0.1')

cursor = conn.cursor()


def callValues():

   bName.config(state=NORMAL)

   aName.config(state=NORMAL)

   bPub.config(state=NORMAL)

   bYear.config(state=NORMAL)

   bEdi.config(state=NORMAL)

   cursor.execute("USE Library")

   sql = "SELECT title,author,publisher,year,edition FROM books WHERE 
title='"+bName.get()+"'"

cursor.execute(sql)

results = cursor.fetchall() # Fetching values from database

conn.commit()

if not results:

   messagebox.showinfo(title='Sorry', message="No Record Found!")

else:

   for row in results:
       aName.insert(0,row[1])
       bPub.insert(0,row[2])
       bYear.insert(0,row[3])
       bEdi.insert(0,row[4])
def updateCall():
   cursor.execute("USE Library")
   sql = "UPDATE books SET
title='"+bName.get()+"',author='"+aName.get()+"',publisher='"+bPub.get()+"',year="+
bYear.get()+", edition='"+bEdi.get()+"'WHERE title='"+bName.get()+"'"
   cursor.execute(sql)
   conn.commit()
   messagebox.showinfo(title='Confirmation', message="Information Updated")
   bName.delete(0,END)
   aName.delete(0,END)
   bPub.delete(0,END)
   bYear.delete(0,END)
   bEdi.delete(0,END)
def deleteCall():
   cursor.execute("USE Library")
   sql = "DELETE FROM books where title='"+bName.get()+"'"
   cursor.execute(sql)
   conn.commit()
   messagebox.showinfo(title='Confirmation', message="Information Deleted")

   win = Tk()
   win.title("Book Updation Form")

   Label(win, text="Enter Book Name").grid(row=0)
   bName = Entry(win,width=35)
   b.Name.grid(row=0,column=1)
```

```
   b1=Button(win,text="Author").grid(row=2)
   b1.grid(row=1,sticky=N,pady=4,columnspan=2)

   Lable(win,text="Author").grid(row=2)
   Lable(win,text="author").grid(row=2)

Label(win, text="Year").grid(row=4)
Label (win, text="Edition").grid(row=5)

aName=Entry(win,width=35,state=DISABLED)
aName.grid(row=2, column=1)
bPub=Entry(win,width=35,state=DISABLED)
bPub.grid(row=3, column=1)
bYear= Entry(win,width=35,state=DISABLED)
bYear.grid(row=4, column=1)
bEdi=Entry(win,width=35,state=DISABLED)
bEdi.grid(row=5, column=1)

b2=Button(win, text='Update Record', command=updateCall)
b2.grid(row=6, column=0, sticky=N, pady=4)
b3=Button(win, text='Delete Record', command=deleteCall)
b3.grid(row=6, column=1, sticky=N, pady=4)
mainloop()
```