

KARNATAK LAW SOCIETY'S

GOGTE INSTITUTE OF TECHNOLOGY

UDYAMBAG, BELAGAVI-590008

(An Autonomous Institution under Visvesvaraya Technological University, Belagavi)

(APPROVED BY AICTE, NEW DELHI)

Department of Computer Science Engineering



JOURNAL SUBMISSION

PYTHON PROGRAMMING (INTEGRATED LAB)

Submitted for the requirements of 4th semester B.E.in CSE for

“Python Programming (Integrated Lab) (18CSL46) ”

Submitted by

NAME	USN
Venkatesh D	2GI19CS175

Under the guidance of

Prof.Prasad Pujar

Dept. of CSE

KLS Gogte Institute of Technology, Belagavi

2020-2021

KARNATAK LAW SOCIETY'S

GOGTE INSTITUTE OF TECHNOLOGY

Udyambag Belagavi -590008

Karnataka, India.

Department of Computer Science and Engineering



Certificate

This is to certify that the Journal Termworks carried out by Name : Venkatesh G Dhongadi bearing **USN: 2GI19CS175** is submitted in partial fulfilment of the requirements for 4th semester B.E. in **PYTHON PROGRAMMING (INTEGRATED LAB)**,

COMPUTER SCIENCE AND ENGINEERING, Visvesvaraya Technological University, Belagavi. It is certified that all corrections/suggestions indicated have been incorporated in the journal. The journal has been approved as it satisfies the academic requirements in respect of research work prescribed for the said degree.

Date:

Place: Belagavi

Signature of Guide

Prof. Prasad Pujar

Prof., Dept. of CSE,

KLS Gogte Institute of Technology,

Belagavi

Name of the Examiners

1. _____

2. _____

Signature of the Examiners

1. _____

2. _____

Index:

Sl.no	Title	Page No.	Marks	Sign
1	Implementation of Lists	4		
2	Implementation of Dictionaries	14		
3	Implementation of Files.	20		
4	Implementation of OOP	30		
5	Implementation of Database	40		
6	Implementation of GUI	59		
7	Implementation of NumPy	70		
8	Implementation of Pandas	80		

TermWork-1

USN : 2GI19CS175

Student Name : Venkatesh

Title of the Experiment : Application of Lists

Experiment No. 1

Tw1 a :

Problem Statement :

Develop a program using Sieve of Eratosthenes method for computing primes up to a specified number.

In mathematics, the Sieve of Eratosthenes is a simple, ancient algorithm for finding all prime numbers up to any given limit. It does so by iteratively marking as composite (i.e., not prime) the multiples of each prime, starting with the first prime number, 2. The multiples of a given prime are generated as a sequence of numbers starting from that prime, with constant difference between them that is equal to that prime.

Program Source Code :

```
def insert(q,item):
    q.append(item)
    print(item," is inserted into the queue ")

def delete(q):
    if len(q)==0:
        print("Queue is empty ")
    else:
        print(q.pop(0)," deleted from queue ")

def display(q):
    if len(q)==0:
        print("Queue is empty ")
    else:
```

```

        print("Status of queue is :")

        for item in q:

            print(item,end=' ')

            #print()

q=[]

while True:

    print("\n 1: Insert 2: Delete 3: Display 4: Exit")

    ch=int(input("Enter the choice "))

    if ch==1:

        item=int(input("Enter the element "))

        insert(q,item)

    elif ch==2:

        delete(q)

    elif ch==3:

        display(q)

    elif ch==4:

        break

    else:

        print("Invalid choice!! Try Again")

```

Output 1 :

```

Enter n upto which prime numbers are required :20
2
3
5
7
11
13
17
19
In [26]:

```

Tw1 b :

Problem Statement :

Develop a menu driven program to implement a queue. The operations would be

- a. Add an item to the queue
- b. Delete an item from queue
- c. Display the queue

Program Source Code :

```
def generatePrimes(n):  
    primes=[]  
    print("Prime numbers upto ",n," are : ")  
    for i in range(2, n+1):  
        if i not in primes:  
            print(i,end=' ')  
            for j in range(i*i, n+1, i):  
                if j not in primes:  
primes.append(j)  
    print("\n\nList of composite numbers upto ",n," is ",primes)  
#driver code(main)  
n=int(input("Enter the number upto which primes are required : "))  
generatePrimes(n)
```

Output 1 :

```
1 :Insert 2:Delete 3:Display 4:Exit
Enter choice :1
Enter elemenet to insert :10
10 Inserted into the queue
1 :Insert 2:Delete 3:Display 4:Exit
Enter choice :1
Enter elemenet to insert :20
20 Inserted into the queue
1 :Insert 2:Delete 3:Display 4:Exit
Enter choice :1
Enter elemenet to insert :30
30 Inserted into the queue
1 :Insert 2:Delete 3:Display 4:Exit
Enter choice :3
Status of the Queue is :
10 20 30
1 :Insert 2:Delete 3:Display 4:Exit
Enter choice :2
10 deleted from the queue
1 :Insert 2:Delete 3:Display 4:Exit
Enter choice :2
10 deleted from the queue
1 :Insert 2:Delete 3:Display 4:Exit
Enter choice :3
Status of the Queue is :
20 30
1 :Insert 2:Delete 3:Display 4:Exit
Enter choice :2
20 deleted from the queue
1 :Insert 2:Delete 3:Display 4:Exit
Enter choice :3
Status of the Queue is :
30
1 :Insert 2:Delete 3:Display 4:Exit
Enter choice :3
Status of the Queue is :
30
1 :Insert 2:Delete 3:Display 4:Exit
Enter choice :2
30 deleted from the queue
1 :Insert 2:Delete 3:Display 4:Exit
Enter choice :3
Empty Queue
1 :Insert 2:Delete 3:Display 4:Exit
Enter choice :4
In [27]:
```

TermWork-2

USN : 2GI19CS175

Student Name : Venkatesh

Title of the Experiment : Implementation of Dictionaries

Experiment No. 2

Tw2 a :

Problem Statement :

Develop a menu driven Python program to implement the mapping:

{USN1: [m,m2,m3], USN2: [m1,m2,m3].... } and perform the following using functions:

- a. Add an entry
- b. Delete an entry
- c. Display all entries
- d. Compute and display average of best two marks for a specific USN

Program Source Code :

```
def addStudent(stud):
    usn=input("Enter USN : ")
    (m1,m2,m3)=[int(m) for m in input("Enter three sub marks : ").split()]
    stud[usn]=(m1,m2,m3)

def delStudent(stud):
    usn=input("Enter USN to be deleted : ")
    if usn in stud:
        stud.pop(usn)
        print(usn," deleted")
    else:
        print(usn," does not exist")

def computeStudent(stud):
    usn=input("Enter USN to compute average : ")
```



```

if usn in stud:
    (m1,m2,m3)=(stud[usn][0],stud[usn][1],stud[usn][2])
    small=m1 if m1<m2 and m2<m3 else m2 if m2<m3 else m3
    avg=(m1+m2+m3-small)/2
    print("USN : ",usn)
    print("Marks are : ",m1,m2,m3)
    print("Average marks : ",avg)
else:
    print(usn," does not exist ")

```

```

def disStudent(stud):
    print("Student Details : ")
    for usn,data in stud.items():
        print("USN: ",usn," Marks: ",data[0], data[1], data[2])

```

```

stud={}
while True:
    print("\n1: Add Student 2: Delete Student 3: Compute Avg 4: Display Details 5: Exit")
    ch=int(input("Enter your choice : "))
    if ch==1:
        addStudent(stud)
    elif ch==2:
        delStudent(stud)
    elif ch==3:
        computeStudent(stud)
    elif ch==4:
        disStudent(stud)
    elif ch==5:
        break

```

else:

print("Invalid choice!!..Try Again.... ")

Output 1 :

```
1:Add 2:Delete 3:Compute Avg 4:Dispaly 5:Exit
Enter choice:1

Enter USN:2GI19CS001

Enter three marks:23 24 25
1:Add 2:Delete 3:Compute Avg 4:Dispaly 5:Exit
Enter choice:1

Enter USN:2GI19CS002

Enter three marks:23 22 23
1:Add 2:Delete 3:Compute Avg 4:Dispaly 5:Exit
Enter choice:4
Student details
USN 2GI19CS001 mark: 23 24 25
USN 2GI19CS002 mark: 23 22 23
1:Add 2:Delete 3:Compute Avg 4:Dispaly 5:Exit
Enter choice:3

Enter USN to compute average :2GI19CS002
USN 2GI19CS002
Marks are : 23 22 23
Average marks are : 23.0
1:Add 2:Delete 3:Compute Avg 4:Dispaly 5:Exit
Enter choice:2

Enter USN to be deleted :2GI19CS001
2GI19CS001 USN deleted
1:Add 2:Delete 3:Compute Avg 4:Dispaly 5:Exit
Enter choice:4
Student details
USN 2GI19CS002 mark: 23 22 23
1:Add 2:Delete 3:Compute Avg 4:Dispaly 5:Exit
Enter choice:5
```

In [29]:

Tw2 b :

Problem Statement :

Store the following information in a dictionary:

Course Code: Course Name, Faculty, Number of registrations.

Perform the following operations using functions:

a. Find Course Name that has highest number of registrations.

b. Given the Course Code, display the associated details.

Display details of all courses.

Program Source Code :

```
def highReg(courses):  
    pass  
  
def courseDetails(courses):  
    code=input("Enter the course code to display its details : ")  
    if code in courses:  
        print("Course_code: ",code, " Course_Name: ", courses[code][0], "  
Faculty_Name: ", courses[code][1], " No._of_Registrations: ",courses[code][2])  
  
def disDetails(courses):  
    for key in courses.keys():  
        print("Course_code: ",key, " Course_Name: ", courses[key][0], " Faculty_Name: "  
, courses[key][1], " No._of_Registrations: ",courses[key][2])  
  
courses = {'19GD21': ['DBMS', 'SFR', 120], '12DF56': ['XCCV', 'FGG',110]}  
  
while True:  
    print("\n1: Courses having highest registration \n2: Display Details of given course  
code \n3: Display all course Details \n4: Exit")  
    ch=int(input("Enter your choice : "))  
    if ch==1:  
        highReg(courses)
```

```

elif ch==2:
    courseDetails(courses)
elif ch==3:
    disDetails(courses)
elif ch==4:
    break
else:
    print("Invalid choice!!..Try Again.... ")

```

Output 1 :

```

1 Store Information
2 Find which course has highest number of registrations
3 Find details of a particular course using course code
4 Display all the course details
5 Exit

Enter Your Choice :: 1

Enter Course Code :: 18CS01

Enter the Course name :: Python Programming

Enter the Faculty name :: SVM

Enter the number of registrations :: 30

1 Store Information
2 Find which course has highest number of registrations
3 Find details of a particular course using course code
4 Display all the course details
5 Exit

Enter Your Choice :: 1

Enter Course Code :: 18CS02

Enter the Course name :: DBMS

Enter the Faculty name :: SFR

Enter the number of registrations :: 23

```

```
1 Store Information
2 Find which course has highest number of registrations
3 Find details of a particular course using course code
4 Display all the course details
5 Exit

Enter Your Choice :: 2

18CS01 has the highest number of registrations of :: 30

1 Store Information
2 Find which course has highest number of registrations
3 Find details of a particular course using course code
4 Display all the course details
5 Exit

Enter Your Choice :: 3

Enter the course code to find the details :: 18CS002

Invalid Course Code!!

1 Store Information
2 Find which course has highest number of registrations
3 Find details of a particular course using course code
4 Display all the course details
5 Exit

Enter Your Choice :: 3

Enter the course code to find the details :: 18CS02

Details of course code 18CS02 is ::
Course Name :: DBMS
Course Code :: 18CS02
```

```
Details of course code 18CS02 is ::
Course Name :: DBMS
Course Code :: 18CS02
Faculty Name :: SFR
Number of registrations :: 23

1 Store Information
2 Find which course has highest number of registrations
3 Find details of a particular course using course code
4 Display all the course details
5 Exit

Enter Your Choice :: 4

All Course Details
course_name :: Python Programming
faculty :: SVM
num_of_registrations :: 30

course_name :: DBMS
faculty :: SFR
num_of_registrations :: 23

1 Store Information
2 Find which course has highest number of registrations
3 Find details of a particular course using course code
4 Display all the course details
5 Exit

Enter Your Choice :: 5
Exiting ...

In [33]:
```

TermWork-3

USN : 2GI19CS175

Student Name : Venkatesh

Title of the Experiment : Implementation of Files

Experiment No. 3

Tw3 a :

Problem Statement :

Write a Python program to read the book information from the user and store in a CSV file containing rows in the following format:

bookNo, title, author, price

Develop a menu-driven program (with functions for each) with the following options:

1:Search Book by author

2:Search Books below specified price (Raise an exception if price entered is ≤ 0)

3:Search Books where title contains the specified word

4:Exit

Program Source Code :

```
import csv

def searchByAuthor():
    author = input("Enter author name to search : ")

    with open("books.csv") as f:
        reader = csv.reader(f)
        result = []

        for row in reader:
            if row[2]==author:
                result.append(row)

        if result == []:
            print("No books with the specified author name")
        else:
```

```
print("Books with the specified author are ")
```

```
for line in result:
```

```
    print(line)
```

```
def searchByPrice():
```

```
    price = int(input("Enter budget price: "))
```

```
    try:
```

```
        if price <= 0:
```

```
            raise ValueError("Invalid input for price")
```

```
        with open("books.csv") as f:
```

```
            reader = csv.reader(f)
```

```
            result = []
```

```
            for row in reader:
```

```
                if int(row[3]) < price:
```

```
result.append(row)
```

```
    if result == []:
```

```
        print("No Books found within the specified price")
```

```
    else:
```

```
        print("Following books are within the budget..")
```

```
        for line in result:
```

```
            print(line)
```

```
    except ValueError as ve:
```

```
        print(ve)
```

```
def searchByKeyword():
```

```
    word = input("Enter keyword to find in title: ")
```

```
    with open("books.csv") as f:
```

```
        reader = csv.reader(f)
```

```
        result = []
```

```

        for row in reader:
            if word in row[1]:
result.append(row)
        if result == []:
            print("No books contain the specified keyword")
        else:
            print("Books found ")
            for line in result:
                print(line)

data = []
while True:
    bookNo=int(input("Enter book number: "))
    title=input("Enter title: ")
    author=input("Enter author: ")
    price=int(input("Enter price: "))
    data.append([bookNo, title, author, price])
    ch = input("Add more? (Y/N) :")
    if ch != 'Y':
        break
    with open("books.csv","w") as f:
        writer = csv.writer(f, lineterminator='\n')
        writer.writerows(data)

while True:
    print("1.Search By Author\n2.Search By Price\n3.Search By Keyword\n4.Exit")
    ch=int(input("Enter your choice : "))
    if ch == 1:
        searchByAuthor()

```



```
elifch == 2:
    searchByPrice()
elifch == 3:
    searchByKeyword()
elifch == 4:
    break
else:
    print("Invalid choice! Try Again!")
```

Output 1 :

```
Enter book number:1
Enter title:Design & Analysis of Algorith
Enter author:Anany Levitin
Enter price:600
Add more? (Y/N)Y
Enter book number:2
Enter title:Database Management System
Enter author:Navate
Enter price:555
Add more? (Y/N)Y
Enter book number:3
Enter title:Python Programming
Enter author:Michael Urban and Jua1 Murach
Enter price:740
Add more? (Y/N)Y
```

```
Enter book number:4

Enter title:Software Engineering

Enter author:Roger S Pressman

Enter price:680

Add more? (Y/N)N
1:Search by Author
2:Search for price
3:Search for keyword
4:Exit

Enter your choice:1

Enter author name to search:Roger S Pressman
Books with the specified author are...
['4', 'Software Engineering', 'Roger S Pressman', '680']
1:Search by Author
2:Search for price
3:Search for keyword
4:Exit

Enter your choice:2

Enter budget price:740
Following books are within the budget...
['1', 'Design & Analysis of Algorith', 'Anany Levitin', '600']
['2', 'Database Management System', 'Navate', '555']
['4', 'Software Engineering', 'Roger S Pressman', '680']
1:Search by Author
2:Search for price
3:Search for keyword
```

```
Enter your choice:3

Enter keyword to find in title:Operating
No books contain the specified keyword.
1:Search by Author
2:Search for price
3:Search for keyword
4:Exit

Enter your choice:2

Enter budget price:-1
Invalid input for price.
1:Search by Author
2:Search for price
3:Search for keyword
4:Exit

Enter your choice:4
```

```
In [22]:
```

Tw3 b :

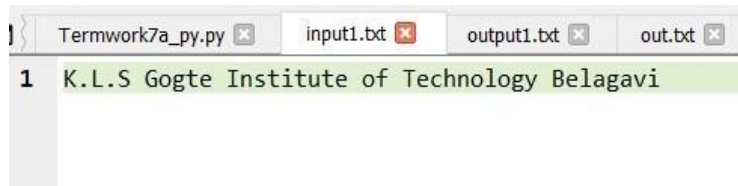
Problem Statement :

Write a Python program to read from a file input.txt and write the contents in reverse order to file output.txt, raise an exception if there is no content in input.txt.

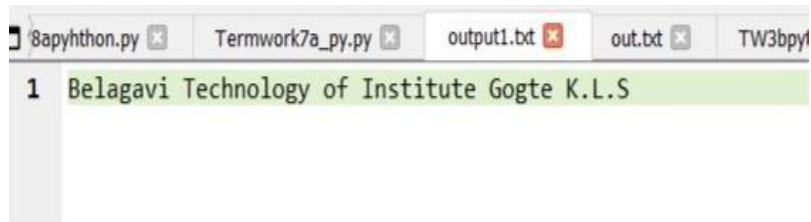
Program Source Code :

```
f2 = open("output.txt","w")
with open("input.txt") as f1:
    try:
        if len(f1.read()) == 0:
            raise ValueError("The file is empty!")
        f1.seek(0)
        lines = f1.readlines()
        start = len(lines) - 1
        for i in range(start, -1, -1):
            f2.write(lines[i])
        f2.close()
    except ValueError as ve:
        print(ve)
```

Output 1 :

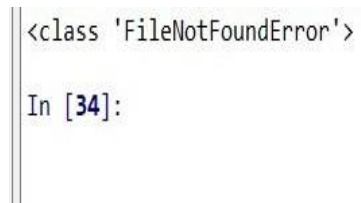


```
Termwork7a_py.py x input1.txt x output1.txt x out.txt x
1 K.L.S Gogte Institute of Technology Belagavi
```



```
8apython.py x Termwork7a_py.py x output1.txt x out.txt x TW3bpyl
1 Belagavi Technology of Institute Gogte K.L.S
```

Output 2 :



```
<class 'FileNotFoundError'>
In [34]:
```

TermWork-4

USN : 2GI19CS175

Student Name : Venkatesh

Title of the Experiment : Implementation of OOP

Experiment No. 4

Tw4 a :

Problem Statement :

Create an object-oriented program that allows you to enter data for customers and employees.

Problem Details

Create a Person class that provides attributes for first name, last name, and email address. This class should provide a property or method that returns the person's fullname.

Create a Customer class that inherits the Person class. This class should add an attribute for a customer number.

Create an Employee class that inherits the Person class. This class should add an attribute for a PAN number.

The program should create a Customer or Employee object from the data entered by the user, and it should use this object to display the data to the user. To do that, the program can use the isinstance() function to check whether an object is a Customer or Employee object.

Program Source Code :

```
class Person: # Creating parent class "Person"

    def __init__(self, fname="", lname="", email=""): # creating parent class constructor

        self.fname = fname # Assigning the values for fname, lname, email

        self.lname = lname

        self.email = email

    def getFullname(self): # Method to get full name

        return self.fname+" "+self.lname # returns fname+lname to the calling function

class Customer(Person): # creating child class "Customer" that inherits parent class "Person"

    def __init__(self, fname, lname, email, cust_id): # constructor for class "Customer"
```

```

    # calling constructor of parent class
    Person.__init__(self, fname, lname, email)

    # creating and assigning the value for new attribute "cust_id" for child classs
    "Customer"

    self.cust_id = cust_id


def getCustId(self): # creating method that returns "cust_id" to the calling function
    return self.cust_id


class Employee(Person): # creating child class "Employee" that inherits parent class
    "Person"

    def __init__(self, fname, lname, email, PAN): # constructor for child class "Employee"
        Person.__init__(self, fname, lname, email)

        self.PAN = PAN


def getPAN_no(self):
    return self.PAN


def Display(Person): # global method to display the details

    # checks if the object passed a "Customer" class object by using "isinstance()" function
    if isinstance(Person, Customer):

        print("Its a Customer class Object")

        print(f"Customer Name :: {Person.getFullname()}")

        print(f"Customer Number :: {Person.getCustId()}")

    else:

        print("Its a Employee class Object")

        print(f"Employee Name :: {Person.getFullname()}")

        print(f"Employee PAN Number :: {Person.getPAN_no()}")


def main(): # main function to run the prog

```

```
# creating object for "Customer" class and passing the values to constructor
Cust_obj = Customer("Sneha", "Kulkarni", "sk@gmail.com", 1)

# creating object for "Employee" class and passing the values to the constructor
Emp_obj = Employee("Namrata", "Shivakale", "ns@gmail.com", "NO21")

Display(Cust_obj)

print()

Display(Emp_obj)

if __name__ == "__main__": # condition to check if there is a function named "main()" in the
program

    main() # class main function to run the program
```

Output 1 :

```
Its a Customer class Object
Customer Name :: Sneha Kulkarni
Customer Number :: 1

Its a Employee class Object
Employee Name :: Namrata Shivakale
Employee PAN Number :: NO21

In [38]:
```

Tw4 b :

Problem Statement :

Create a class Polygon that uses the initializer to initialize the number of sides and the default value of zero for each side as instance variables. It also defines member functions to read values for each side and displays the values. Derive two classes viz., Triangle and Rectangle from Polygon class. The two derived classes define member functions to compute area and display area.

Program Source Code :

```
"""
polygon inheritance
"""

class Polygon:
    def __init__(self, n):
        self.numSides = n
        self.sides = [0 for i in range(self.numSides)]

    def readSides(self):
        print("Enter the sides : ")
        # for i in range(self.numSides):
        self.sides = [float(input()) for i in range(self.numSides)]

    def dispSides(self):
        print("Sides are...")
        for i in range(self.numSides):
            print(self.sides[i])

class Triangle(Polygon):
    def __init__(self, n):
        super().__init__(n)

    def computeArea(self):
        for i in range(self.numSides):
```



```

        s=0
        s+=self.sides[i]
        a=s/2
        self.area=(a*(a-self.sides[0])*(a-self.sides[1])*(a-self.sides[2]))**0.5

    def dispArea(self):
        print("Area of Triangle is : ",self.area)

class Rectangle(Polygon):
    def __init__(self, n):
        super().__init__(n)
    def computeArea(self):
        for i in range(self.numSides):
self.area=(self.sides[0] * self.sides[1])
    def dispArea(self):
        print("Area of Rectangle is : ", round(self.area, 2))

print(" Triangle ")
t1 = Triangle(3)
t1.readSides()
t1.dispSides()
t1.computeArea()
t1.dispArea()
print(" Rectangle ")
r1 = Rectangle(2)
r1.readSides()
r1.dispSides()
r1.computeArea()
r1.dispArea()

```

Output 1 :

```
Created Trinagle Object ::  
Enter the values of sides ::  
  
3  
  
4  
  
5  
Sides are :: 3 4 5  
Area of Triangle is :: 6.0  
  
Created Rectangle Object  
Enter the values of sides ::  
  
4  
  
5  
Sides are :: 4 5  
Area of Rectangle is :: 20  
  
In [35]:
```

TermWork-5

USN : 2GI19CS175

Student Name : Venkatesh

Title of the Experiment : Implementation of Database

Experiment No. 5

Tw5 a :

Problem Statement :

Write a Python program to perform the following:

- a. Create a database named “products.db”
- b. Create a table named “products” that has the following fields:
 - prodID: int
 - name: text
 - quantity: int
 - price: real
- c. Insert n records into the table reading the values for each item from the user.
- d. Display the recordset after fetching all the rows.
- e. Delete a product whose product ID is entered by the user.
- f. Increase the price of all products whose current price is less than Rs.50, by 10%.
- g. Display all the products whose quantity is less than 40.

Program Source Code :

```
from sqlite3 import *
import sqlite3

def readItems():
    id=int(input("Enter iD "))
    name =input("Enter name ")
    qty=int(input("Enter qty "))
    price=float(input("Enter price "))
    return (id,name,qty,price)
```

```
conn=connect("products.db")
```

```
cur=conn.cursor()
```

```
query="create table products(prodID INTEGER,name TEXT ,quantity INTEGER ,PRICE  
REAL)"
```

```
cur.execute(query);
```

```
conn.commit()
```

```
n=int(input("How many items to read ?"))
```

```
for i in range(n):
```

```
    values =readItems()
```

```
    query="insert into products(prodID,name,quantity,price) values(?,?,?,?)"
```

```
    cur.execute(query,values)
```

```
    conn.commit()
```

```
print("The record in the table ")
```

```
query="Select * from products"
```

```
cur.execute(query)
```

```
for row in cur.fetchall():
```

```
    print(row)
```

```
pID =int(input("Enter product id "))
```

```
query="delete from products where prodID =?"
```

```
cur.execute(query,(pID,))
```

```
conn.commit()
```

```
query="update table products set price =price+price*0.1 where price <50"
```

```
cur.execute(query)
```

```
conn.commit()
```

```
print("The record in the table after update ")
query="Select * from products"
cur.execute(query)
for row in cur.fetchall():
    print(row)

query="select * from products where quantity<40"
cur.execute(query)
rows=cur.fetchall()
if rows==[]:
    print("No, products have quantity less than 40")
else:
    print("Products having quantity less than 40 are")
    for row in rows:
        print(row)
cur.close()
conn.close()
```

Output 1 :

```
How many item to add? 5
Enter product id :1
Enter product name :Pen
Enter quantity :30
Enter price :30
Enter product id :2
Enter product name :Mask
Enter quantity :50
Enter price :25
Enter product id :3
Enter product name :Notebook
Enter quantity :50
Enter price :100

Enter product id :4
Enter product name :Drafter
Enter quantity :20
Enter price :250
Enter product id :5
Enter product name :Container
Enter quantity :8
Enter price :300
The records in the table are :
(1, 'Pen', 30, 30.0)
(2, 'Mask', 50, 25.0)
(3, 'Notebook', 50, 100.0)
(4, 'Drafter', 20, 250.0)
(5, 'Container', 8, 300.0)
```

Tw5 b :

Problem Statement :

Develop a python program that does the following operations of an EMPLOYEE.

Fields are:EmpID,Name, Age,Salary

1. Add a record
2. Find all records above a certain salary
3. Update salary of all employees by 5% who have salary less 10,000
4. Delete all employees with age greater than 60

Program Source Code :

```
"""

basic=50000

da=70% of basic

hra - 10% of basic

ta - 5% of basic

deductions(tax) - 20% of basic

gross salary = basic+da+hra+ta

net salary= deductions-gross salary

"""

class Manager:

    def __init__(self, name, empID, basic):

        self.name=name

        self.empID=empID

        self.basic=basic

    def computeSalary(Self):

        print("To be o")

class HRManager(Manager):

    def __init__(self, name, empID, basic):

        super().__init__(name, empID, basic)
```

```
def computeSalary(self):  
    da = self.basic * 0.7  
hra = self.basic * 0.2  
dedu = self.basic * 0.05  
self.grossSal = self.basic+ da + hra  
self.netSal = self.grossSal - dedu  
  
def dispDetails(self):  
    print("Name : ", self.name)  
    print("EmpID : ", self.empID)  
    print("Basic Salary : ", self.basic)  
    print("Gross Salary : ", self.grossSal)  
    print("Net Salary : ", self.netSal)
```

```
class SalesManager(Manager):  
  
    def __init__(self, name, empID, basic):  
        super().__init__(name, empID, basic)  
  
    def computeSalary(self):  
        da = self.basic * 0.7  
hra = self.basic * 0.1  
dedu = self.basic * 0.05  
        ta = self.basic * 0.05  
self.grossSal = self.basic + da + hra + ta  
self.netSal = self.grossSal - dedu  
  
    def dispDetails(self):  
        print("Name : ", self.name)  
        print("EmpID : ", self.empID)  
        print("Basic Salary : ", self.basic)  
        print("Gross Salary : ", self.grossSal)
```



```

        print("Net Salary : ", self.netSal)

h1 = HRManager("Piyu", 1001, 10000)

h1.computeSalary()

h1.dispDetails()

s1 = SalesManager("SRM",2001,20000)

s1.computeSalary()

s1.dispDetails()

```

Output 1 :

```

Table has the following records
(101, 'SRM', 47, 45000)
(102, 'SFR', 51, 49000)
(103, 'SMR', 35, 35000)
(104, 'SKR', 62, 56000)
(105, 'SWR', 61, 12000)
(106, 'SGR', 43, 8000)
(107, 'TFR', 56, 7000)
(108, 'YFR', 60, 15000)

Enter salary level :30000
Following have salary more than 30000
(101, 'SRM', 47, 45000)
(102, 'SFR', 51, 49000)
(103, 'SMR', 35, 35000)
(104, 'SKR', 62, 56000)
Table after update operation is
(101, 'SRM', 47, 45000)
(102, 'SFR', 51, 49000)
(103, 'SMR', 35, 35000)
(104, 'SKR', 62, 56000)
(105, 'SWR', 61, 12000)
(106, 'SGR', 43, 8400)
(107, 'TFR', 56, 7350)
(108, 'YFR', 60, 15000)
Table after deletion operation is
(101, 'SRM', 47, 45000)
(102, 'SFR', 51, 49000)
(103, 'SMR', 35, 35000)
(106, 'SGR', 43, 8400)
(107, 'TFR', 56, 7350)
(108, 'YFR', 60, 15000)

```

In [29]:

TermWork-6

USN : 2GI19CS175

Student Name : Venkatesh

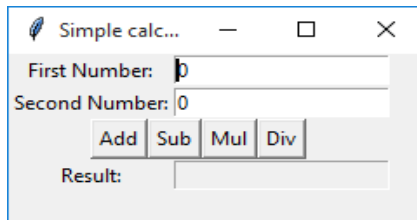
Title of the Experiment : Implementation of GUI

Experiment No. 6

Tw6 a :

Problem Statement :

Develop the following GUI application.



Program Source Code :

```
from tkinter import *

def add():
    val3.set(val1.get() + val2.get())

def sub():
    val3.set(val1.get() - val2.get())

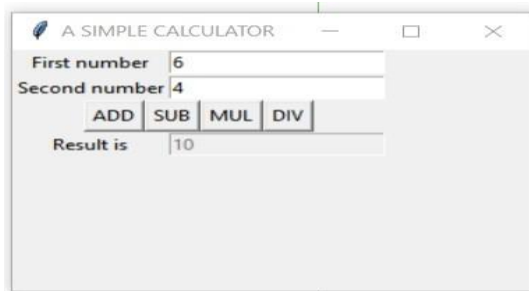
def mul():
    val3.set(val1.get() * val2.get())

def div():
    if val2.get()==0:
        messagebox.showerror(title='Fatal error',message="Cannot divide")
    else:
        val3.set(val1.get() / val2.get())

root = Tk()
```

```
root.geometry("300x200")
root.title("A Simple calculator")
lb1 = Label(root, text="First Number")
val1 = IntVar()
t1 = Entry(root, width=20, textvariable=val1)
lb2 = Label(root, text="Second Number")
val2 = IntVar()
t2 = Entry(root, width=20, textvariable=val2)
frm = Frame(root)
b1 = Button(frm, text='ADD', command= add)
b2 = Button(frm, text='SUB', command= sub)
b3 = Button(frm, text='MULTIPLY', command= mul)
b4 = Button(frm, text='DIVIDE', command= div)
lb3 = Label(root, text='RESULT')
val3 = IntVar()
t3 = Entry(root, width=20, state='disabled', textvariable=val3)
lb1.grid(row=0, column=0)
t1.grid(row=0, column=1)
lb2.grid(row=1, column=0)
t2.grid(row=1, column=1)
b1.grid(row=2, column=0)
b2.grid(row=2, column=1)
b3.grid(row=2, column=2)
b4.grid(row=2, column=3)
frm.grid(row=2, columnspan=2)
lb3.grid(row=3, column=0)
t3.grid(row=3, column=1)
root.mainloop()
```

Output 1 :



A SIMPLE CALCULATOR

First number 6

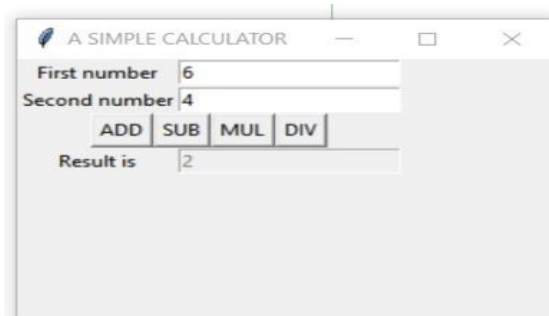
Second number 4

ADD SUB MUL DIV

Result is 10

This screenshot shows a window titled 'A SIMPLE CALCULATOR'. It contains two input fields: 'First number' with the value '6' and 'Second number' with the value '4'. Below these fields are four buttons labeled 'ADD', 'SUB', 'MUL', and 'DIV'. The 'ADD' button is highlighted. At the bottom, a label 'Result is' is followed by a text box containing the value '10'.

Output 2 :



A SIMPLE CALCULATOR

First number 6

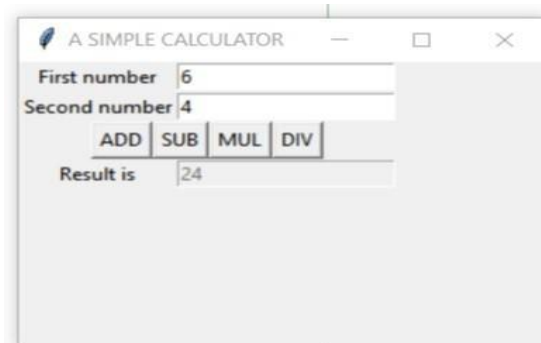
Second number 4

ADD SUB MUL DIV

Result is 2

This screenshot shows the same 'A SIMPLE CALCULATOR' window. The 'SUB' button is now highlighted. The 'Result is' text box now displays the value '2'.

Output 3:



A SIMPLE CALCULATOR

First number 6

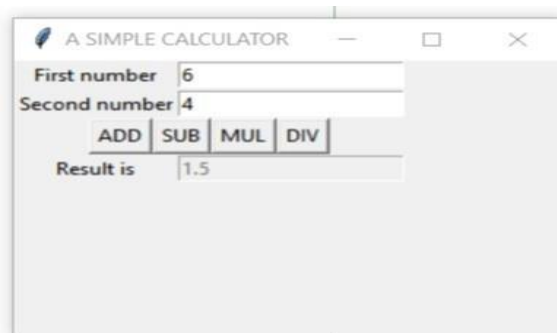
Second number 4

ADD SUB MUL DIV

Result is 24

This screenshot shows the 'A SIMPLE CALCULATOR' window with the 'MUL' button highlighted. The 'Result is' text box now displays the value '24'.

Output 4 :



A SIMPLE CALCULATOR

First number 6

Second number 4

ADD SUB MUL DIV

Result is 1.5

This screenshot shows the 'A SIMPLE CALCULATOR' window with the 'DIV' button highlighted. The 'Result is' text box now displays the value '1.5'.

Tw6 b :

Problem Statement :

Create a GUI program that calculates the hypotenuse of a right angled triangle after the user enters the lengths of the two sides and clicks the Calculate button.

Program Source Code :

```
from tkinter import *

import math

def computeHpy():

    a = side1.get()

    b = side2.get()

    result = math.sqrt(a*a + b*b)

    res.set("hypothenuse is : "+ str(round(result,2)))

root = Tk()

root.geometry("300x200")

root.title("Compute Hypotenuse")

lb1 = Label(root, text="Enter 1 side")

lb2 = Label(root, text="Enter 2 side")

side1 = IntVar()

side2 = IntVar()

t1 = Entry(root, width=20, textvariable=side1)

t2 = Entry(root, width=20, textvariable=side2)

b1 = Button(root, text="Compute hypothenuse", command=computeHpy)

res = StringVar()

lb3 = Label(root, text="", textvariable=res)

lb1.grid(row=0,column=0)
```

```
t1.grid(row=0,column=1)

lb2.grid(row=1,column=0)

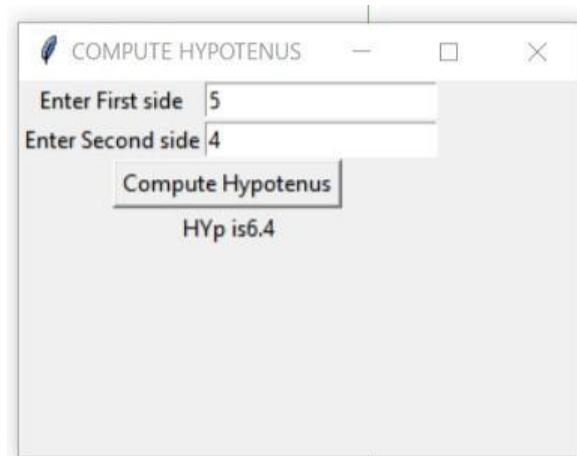
t2.grid(row=1,column=1)

b1.grid(row=2,columnspan=2)

lb3.grid(row=3,columnspan=2)


root.mainloop()
```

Output 1 :



TermWork-7

USN : 2GI19CS175

Student Name : Venkatesh

Title of the Experiment : Implementation of NumPy.

Experiment No. 7

Tw7 a :

Problem Statement :

Three IA's are conducted for a class of 10 students for the subject Maths. The name, marks and USN are read from a file in.txt. Find the average of the IA for each student and write the USN and average to a file out.txt. Display the highest average of the class on the console.

Program Source Code :

```
import numpy as np

arr = np.loadtxt("input.txt",dtype=str, delimiter=",")

marksArray = arr[:,1:4].astype(float) #shape(10,5)

usnArray = arr[:,4]

avg = np.mean(marksArray,axis=1)

with open("out.txt","w") as outFile:

    for i in range(len(usnArray)):

        outFile.write(usnArray[i] + " " + str(round(avg[i],2)) + "\n")

print("Highest average is", round(np.max(avg),2))


for usn in np.nditer(usnArray):

    print(usn)
```

Input

```
rs\mahantesh shivakale\Documents\input.txt.txt
kend.py x input.txt.txt x Studentfrontend.py x T
Sneha Kulkarni,80,90,91,2GI19CS110
Vaishnavi Vadakeri,90,92,89,2GI19CS111
Sahana Kulkarni,76,78,65,2GI19CS112
Nidhi T,65,78,56,2GI19CS113
Bhargavi S,65,45,74,2GI19CS114
Sndhya bhat,76,78,82,2GI19CS115
Pratima Kammar,54,65,58,2GI19CS116
Veda Kammar,82,92,85,2GI19CS117
Madhushri Rajoli,84,65,70,2GI19CS118
Namrata Shivakale,65,78,90,2GI19CS119
```

```
kend.py x out.txt x input.txt.txt x
2GI19CS110 87.0
2GI19CS111 90.33
2GI19CS112 73.0
2GI19CS113 66.33
2GI19CS114 61.33
2GI19CS115 78.67
2GI19CS116 59.0
2GI19CS117 86.33
2GI19CS118 73.0
2GI19CS119 77.67
```

Output 1 :

```
Highest average is 90.33
```

```
In [41]:
```


Tw7 b :

Problem Statement :

Write a NumPy program to convert a list of numeric values into a one-dimensional NumPy array and then reverse it. Also search for a specific element; if found display all its positions in the reversed array; else an error message.

Program Source Code :

```
import numpy as np

lst = [23,14,54,46,67,87,34,14,66,14,55]

#to reverse ndarray use slicing as ndarray_name

arr = np.array(lst)

print("Original list is\n", lst)

print("Original array is\n",arr)

arr = arr[::-1] #reverse array

print("Reversed array is\n", arr)

key = int(input("Enter element to search:"))

result = np.array(np.where(arr==key))

if result.size == 0:

    print(key, "does not exist!")

else:

    print(key, "exit at following indices: ")

    for index in result:

        print(index, end=" ")
```

Output 1 :

```
Original list is
[23, 14, 54, 46, 67, 87, 34, 14, 66, 14, 55]
Original array is
[23 14 54 46 67 87 34 14 66 14 55]
Reversed array is
[55 14 66 14 34 87 67 46 54 14 23]

Enter element to search:14
14 exists at following indices:
[1 3 9]

In [42]:
```

TermWork-8

USN : 2GI19CS175

Student Name : Venkatesh

Title of the Experiment : Implementation of Pandas

Experiment No. 8

Tw8 a :

Problem Statement :

Write a pandas program to reverse order (rows and columns) of a given database frame.

Original Data frame

	w	x	y	z
0	68	78	84	86
1	75	85	94	97
2	86	96	89	96
3	80	80	83	72
4	66	86	86	83

Program Source Code:

```
import pandas as pd
```

```
#df = pd.DataFrame({'W':[68,75,84,86],'X':[75,85,94,97],  
'Y':[86,96,89,96],'Z':[80,80,83,72], 'A':[66,86,86,83]})
```

```
df =  
pd.DataFrame({'W':[68,75,86,80,66],'X':[78,85,96,80,86],'Y':[84,94,89,83,86],'Z':[86,97,96,7  
2,83]});
```

```
    #'W':[67,45,34,56,78],
```

```
    #'X':[34,23,65,87,12],
```

```
    #'Y':[43,54,65,88,10],
```

```

#Z:[56,34,23,67,19])

#s[::-1]

print("Original dataframe is")

print(df)

print("Reverse column order is")

print(df.loc[:,::-1])

print("Reverse order is")

print(df.loc[::-1])

print("Reverse row order & reset index is")

print(df.loc[::-1].reset_index(drop=True))

```

Output 1 :

```

Original dataframe is
   W  X  Y  Z
0  68  78  84  86
1  75  85  94  97
2  86  96  89  96
3  80  80  83  72
4  66  86  86  83
Reverse column order is
   Z  Y  X  W
0  86  84  78  68
1  97  94  85  75
2  96  89  96  86
3  72  83  80  80
4  83  86  86  66
Reverse order is
   W  X  Y  Z
4  66  86  86  83
3  80  80  83  72
2  86  96  89  96
1  75  85  94  97
0  68  78  84  86
Reverse row order & reset index is
   W  X  Y  Z
0  66  86  86  83
1  80  80  83  72
2  86  96  89  96
3  75  85  94  97
4  68  78  84  86
In [43]:

```

Tw8 b :

Problem Statement :

Write pandas program to create a Data frame from csv and perform the following operations:

- Display column names of data frame
- Read the column name and display unique values
- Display frequency of occurrence of each unique value
- Count of total number of records in the Data frame

Program Source Code :

```
import pandas as pd
import numpy as np
car_data = pd.read_csv("mtcars.csv", index_col='model')
cols = list(car_data.columns)
print("Columns of the dataframe are")
for col in cols:
    print(col)
col_name = input("Enter the column name:")
print("Unique values in the", col_name, "are")
uniq_val = np.unique(car_data[col_name])
print(uniq_val)
col_values = list(car_data[col_name])
print("Frequency of occurrence")
for val in uniq_val:
    print(val, "appears", col_values.count(val), "times")
print("Total records in the dataframe is", len(col_values))
```

Output 1 :

Columns of the dataframe are

mpg
cyl
disp
hp
drat
wt
qsec
vs
am
gear
carb

Enter the column name:gear

Unique values in the gear are

[3 4 5]

Frequency of occurrence

3 appears 15 times

4 appears 12 times

5 appears 5 times

Total records in the dataframe is 32

In [44]: