ACTIVITY 1

>>> Source Code

```
f = open("1.txt", "w")
f.write("I like to code\n")
f.write("I'm a python developer\n")
f.close()

f = open("1.txt", "r")
x = f.read().splitlines()
print(x[-1])

s = f.readline()
print(s[10:])

n = int(input("Enter the line number: "))
for i in range (len(x)):
    if i == (n - 1):
        print(x[i])
    else:
        break
f.close()

f = open("1.txt")
a = f.read().split()
c = 0
for i in a:
    if i[0] not in a:
        c = 1
    else:
        c += 1
    print(f"Words beginning with {i[0]} are: {c}")
f.close()
```

ACTIVITY 2

>>> Source Code

```
def isvowel():
      f = open("file1.txt")
      f1 = open("file2.txt", "a")
      a = f.read().split()
    for i in a:
        if a[0].lower not in "aeiou":
            f1.write(a)
        else:
            pass
    f.close()
    f1.close()

isvowel()
```

ACTIVITY 3

>>> Source Code

```
import csv
file = open("students.csv", "r")
obj = csv.reader(delimiter = '\t')

#1

def copyIntoTuple():
    listOfStudents = []
    for i in obj:
        listOfStudents.append(tuple(obj))
    print(listOfStudents)

copyIntoTuple()

#2

#Names of students with no. of year < 3
def numOfYear():
    for i in obj:
        if int(i['no of years']) < 3:
            print(i['firstname'], i['lastname'])

numOfYear()

#Number of people in each department
def numOfstd():
    d = {}
    for i in obj:
        if i['department'] in d:
            d[i['department']] += 1
        else:
            d[i['department']] = 1
    for i in d:
        print('{}: {}'.format(i,d[i]))

numOfstd()
```

ACTIVITY 4

>>> Source Code

```
f = open("myfile.txt")
x = f.read().split()

d = {}
for i in x:
    if i in d:
        d[i] += 1
    else:
        d[i] = 1
print(len(x))
print(len(d))

c = ''
e = 0
for i in d:
    if d[i] > e:
        e = d[i]
        c = i
a = d.get(key = c)
print(a)

def find_longest_word():
    s = ''
    g = 0
    for i in d:
        if len(i) > g:
            g = len(i)
            s = i
    return s
print(find_longest_word())

n = int(input("Enter a number: "))
def filter_long_words(n):
    l = []
    for i in d:
        if len(i) > n:
            l.append(i)
    return l
print(filter_long_words(n))

f.close()
```

ACTIVITY 5

>>> Source Code

```
import pickle as pk
import os

d = {}
def write():
    f = open("hotel.dat","ab")
    n = int(input("Enter the number of entries: "))
    for i in range (n):
        d["roomno"] = input("Enter room number: ")
        d["name"] = input("Enter name: ")
        d["duration"] = input("Enter the duration of stay: ")
        pk.dump(d, f)
    f.close()
    print("Data recorded succesfully.\n")

def read():
    f = open("hotel.dat","rb")
    try:
        while (True):
            obj = pk.load(f)
            print(obj)
    except EOFError:
        f.close()
        pass

def readspecific():
    f = open("hotel.dat","rb")
    n = input("Enter the room number: ")
    flag = False
    try:
        while (True):
            obj = pk.load(f)
            if obj["roomno"] == n:
                print(obj)
            flag = True
    except EOFError:
        f.close()
        pass
    if flag == False:
        print("Record not found\n")

def modify():
    f = open("hotel.dat","rb")
    f1 = open("new.dat","ab")
    n = input("Enter the room number: ")
```

```
    flag = False
    try:
        while (True):
            obj = pk.load(f)
            if obj["roomno"] == n:
                obj["duration"] = input("Enter the new duration: ")
                pk.dump(obj, f1)
            else:
                pk.dump(obj, f1)
            flag=True
    except EOFError:
        f.close()
        f1.close()
        pass
    os.remove("hotel.dat")
    os.rename("new.dat", "hotel.dat")
    if flag == False:
        print("Record not found\n")
    else:
        print("Record modified successfully\n")

def delete():
    f = open("hotel.dat","rb")
    f1 = open("new.dat","ab")
    n = int(input("Enter the room number you want to delete: "))
    flag = False
    try:
        while (True):
            obj = pk.load(f)
            if obj["roomno"] == n:
                pass
            else:
                pk.dump(obj, f1)
            flag = True
    except EOFError:
        f.close()
        f1.close()
        pass
    os.remove("hotel.dat")
    os.rename("new.dat", "hotel.dat")
    if flag == False:
        print("Record not found\n")
    else:
        print("Record deleted successfully\n")

while (True):
    print(">>>Welcome to hotel database<<<\n")
    print("1. Write a new entries")
    print("2. Read all the entries")
    print("3. Read a specific entry")
```

```
print("4. Modify an entry")
print("5. Delete an entry")
print("6. Exit")
v = int(input("Enter your choice: "))
if v == (1):
    write()
elif v == (2):
    read()
elif v == (3):
    readspecific()
elif v == (4):
    modify()
elif v == (5):
    delete()
elif v == (6):
    print("Program is terminating.")
    break
else:
    print("Enter the correct input")
```

ACTIVITY 6

>>> Source Code

```
import csv
file = open('placement.csv', 'r')
obj = csv.reader(file)

#print the data
for i in obj:
    print(i)

#total number of people who came for the test
def numOfPeople():
    c = 0
    for i in obj:
        if i['NAME'] not None:
            c += 1
    print('{} people came for the placement test'.format(c))

numOfPeople()

#top n names on basis of total marks
def totalMarks():
    top = 0
    for i in obj:
        total = 0
        total += int(i['MARKS 1']) + int(i['MARKS 2']) + int(i['MARKS 3']) +
int(i['MARKS 4']) + int(i['MARKS 5'])
        top = max(top, total)
        if total >= top:
            print(i['NAME'])

totalMarks()
```

ACTIVITY 7

>>> Source Code

```
def count(n):
    a = len(str(n))
    print(a)

def reverse(n):
    print(str(n)[::-1])

def hasdigit(n):
    if str(n).isdigit() == True:
        print(True)
    else:
        print(False)

def show(n):
    a = len(str(n))
    s = ''
    for i in (str(n)):
        if a != 0:
            s += str(int(i) * (10 ** (a-1))) + '+'
        else:
            s += str(int(i) * (10**(a-1)))
        a-=1
    print(s)

n=int(input('Enter a number: '))
count(n)
reverse(n)
hasdigit(n)
show(n)
```

ACTIVITY 8

>>> Source Code

```
n = int(input('Enter a number: '))

#1
def Generatefactors(n):
    l = []
    for i in range (1, int(n // 2) + 1):
        if n % i == 0:
            l.append(i)
    return l

#2
def isPrimeNo(n):
    flag = False
    for i in range (2, int(n ** 0.5) + 1):
        if n % i == 0:
            flag = True
            break
    if flag == True:
        print('{} is not a prime number'.format(n))
    else:
        print('{} is a prime number'.format(n))

isPrimeNo(n)

#3
def isPerfectNo(n):
    ln = Generatefactors(n)
    if sum(ln) == n:
        print('{} is a perfect number'.format(n))
    else:
        print('{} is not a perfect number'.format(n))

isPerfectNo(n)
```

ACTIVITY 9

>>> Source Code

```
def factorial(n):
    f = 1
    for i in range (1, n + 1):
        f = f * i
    return f

def pascal(n):
    for i in range (n):
        for j in range (1, n - i):
            print(" ", end = '')
        for k in range (0, i + 1):
            c = int(factorial(i) / (factorial(k) * factorial(i-k)))
            print(" ", c, end = "")
        print()

n=int(input("Enter the number of rows: "))
pascal(n)
```

ACTIVITY 10

>>> Source Code

```
def binary(n):
    l = []
    while n >= 1:
        if n % 2 == 0:
            l.append(0)
        elif n % 2 == 1:
            l.append(1)
        else:
            l.append(1)
            break
        n = n // 2
    l = l[::-1]
    s = ''.join(map(str, l))
    print(s)

def octal(n):
    l = []
    while True:
        if n // 8 < 8:
            l.append(n % 8)
            l.append(n // 8)
            break
        else:
            l.append(n % 8)
        n = n // 8
    l = l[::-1]
    s = ''.join(map(str, l))
    print(s)

def hex(n):
    l = []
    while True:
        if n // 16 < 16:
            l.append(n % 16)
            l.append(n // 16)
            break
        else:
            l.append(n % 16)
        n=n // 16
    l = l[::-1]
    s = ''
    for i in l:
        if i == 10:
            s += 'A'
        elif i == 11:
```

```
            s += 'B'
        elif i == 12:
            s += 'C'
        elif i == 13:
            s += 'D'
        elif i == 14:
            s += 'E'
        elif i == 15:
            s += 'F'
        else:
            s += str(i)
    print(s)

while True:
    n = int(input("Enter the Number: "))
    d = input("Enter 'B' for binary, 'O' for octal, 'H' for hexa-decimal or
'E' to exit: ")
    if d == 'B':
        binary(n)
    elif d == 'O':
        octal(n)
    elif d == 'H':
        hex(n)
    elif d == 'E':
        print("Terminating Program")
        break
    else:
        print("Enter a correct input")
```

ACTIVITY 11

>>> Source Code

```
def bubble(l):
    n = len(l)
    for i in range (n):
        for j in range (0, n - i - 1):
            if l[j] > l[j + 1]:
                l[j], l[j + 1] = l[j + 1], l[j]
    print(f"The sorted list is: {l}")

def binary(l, key):
    first = 0
    last = len(l) - 1
    flag = False
    while (first <= last and not flag):
        mid = (first + last) // 2
        if l[mid] == key:
            flag = True
            print(f"It is at {mid} index")
        else:
            if key < l[mid]:
                last = mid - 1
            else:
                first = mid + 1
    if flag == False:
        print("Element not found in the list")

def linear(l, key):
    flag = False
    for i in range (len(l)):
        if l[i] == key:
            flag = True
            print(f"It is at {i} index")
            break
    if flag == False:
        print("Element not found in the list")

l = list(map(int, input("Enter the list elements seperated by space:
").split()))
n = input("Enter 'B' for bubble sort, 'L' for linear search or 'BS' for
binary search: ")
if n == 'B':
    bubble(l)
elif n == 'L':
    key = int(input("Enter the element to be found: "))
    linear(l, key)
elif n == 'BS':
```

```
key = int(input("Enter the element to be found: "))
binary(l, key)
```

ACTIVITY 12

>>> Source Code

```
from tkinter import *

def si():
    a = int(p.get())
    b = int(t.get())
    c = int(r.get())
    s = (a * b * c) / 100
    global result
    result.set(str(s))

root = Tk()
root.title('SI Calculator')

p = StringVar(root)
t = StringVar(root)
r = StringVar(root)
result = StringVar(root)

label1 = Label(root, text = 'Simple Interest Calculator')
label2 = Label(root, text = 'Enter the principal amount: ')
text1 = Entry(root, textvariable = p)
label3 = Label(root, text = 'Enter the time period: ')
text2 = Entry(root, textvariable = t)
label4 = Label(root, text = 'Enter the rate of interest: ')
text3 = Entry(root, textvariable = r)

label1.grid(row = 0)
label2.grid(row = 1, column = 0)
text1.grid(row = 1, column = 1)
label3.grid(row = 2, column = 0)
text2.grid(row = 2, column = 1)
label4.grid(row = 3, column = 0)
text3.grid(row = 3, column = 1)

label5 = Label(root, textvariable = result)
button1 = Button(root, text = 'Submit', command = si)
button1.grid(row = 4)
label5.grid(row = 5)
root.mainloop()
```

ACTIVITY 14

>>> Source Code

```
queue = []

def push(q, n):
    q.append(n)

#1
def addMember():
    dataInput = list(map(str,input().split()))
    push(queue, dataInput)

#2
def length():
    ln = 0
    for i in queue:
        ln += 1
    print('Length of queue is {}'.format(ln))

#3
def numberOfApplicants():
    d = {}
    for i in queue:
        if i[2] in d:
            d[i[2]] += 1
        else:
            d[i[2]] = 1
    for i in d:
        print('{} class has {} applicants'.format(i,d[i]))
```

ACTIVITY 2

>>> Source Code

```
import mysql.connector as mc
connectMySQL = mc.connect(host = 'localhost', user = 'root', password =
'root', database = 'mydb')

cursor = connectMySQL.cursor(buffered = True)

def insert_values():
      item_code = input("Enter the ItemCode: ")
      item_name = input("Enter the ItemName: ")
      price = float(input("Enter the Price: "))
      sql = f"insert into ITEM values ('{item_code}', '{item_name}',
{price})"
      cursor.execute(sql)
      connectMySQL.commit()

def display_records():
      sql = "select * from ITEM"
      cursor.execute(sql)
      result = cursor.fetchall()
      for i in result:
            print(i)

def search():
      item_code = input("Enter the ItemCode: ")
      sql = "select * from ITEM where Itemcode = '{item_code}'"
      cursor.execute(sql)
      result = cursor.fetchone()
      print(result)
```

ACTIVITY 3

>>> Source Code

```
import mysql.connector as mc
connectMySQL = mc.connect(host = 'localhost', user = 'root', password =
'root', database = 'mydb')

cursor = connectMySQL.cursor(buffered = True)

def insert_values():
      roll_no = int(input("Enter RollNo: "))
      name = input("Enter Name: ")
      u_class = int(input("Enter Class: "))
      dob = input("Enter DOB in DD-MM-YYYY format: ")
      gender = input("Enter Gender: ")
      sql = f"insert into STUDENT values({roll_no}, '{name}', {u_class},
'{dob}', '{gender}')"
      cursor.execute(sql)
      connectMySQL.commit()

def update_values():
      roll_no = int(input("Enter RollNo: "))
      values = ['name', 'u_class', 'dob', 'gender']
      print(f'Available attributes to update are: {values}')
      user_input = input("Enter value to be updated: ")
      if user_input.lower() in values:
            if user_input.lower() == 'name':
                  name = input("Enter Name: ")
                  sql = f"update STUDENT set Name = '{name}' where RollNo =
{roll_no}"
                  cursor.execute(sql)
            elif user_input.lower() == 'u_class':
                  u_class = int(input("Enter Class: "))
                  sql = f"update STUDENT set Class = {u_class} where RollNo =
{roll_no}"
                  cursor.execute(sql)
            elif user_input.lower() == 'dob':
                  dob = input("Enter DOB in DD-MM-YYYY format: ")
                  sql = f"update STUDENT set DOB = '{dob}' where RollNo =
{roll_no}"
                  cursor.execute(sql)
            else:
                  gender = input("Enter Gender: ")
                  sql = f"update STUDENT set DOB = '{dob}' where RollNo =
{roll_no}"
                  cursor.execute(sql)
            print("Data succesfully updated")
      else:
```

```
      print("No such attribute in table")
connectMySQL.commit()
```

ACTIVITY 4

>>> Source Code

```
from tkinter import *
from tkinter import messagebox
import mysql.connector as mc
from tabulate import tabulate

connectMySQL = mc.connect(host = 'localhost', user = 'root', password =
'root', database = 'mydb')

cursor = connectMySQL.cursor(buffered = True)

def print_table():
      cursor.execute("select * from BUS")
      result = cursor.fetchall()
      return tabulate(result, headers = ['BusNo', 'Origin', 'Dest', 'Rate',
'Km'], tablefmt = 'psql')

def show_details():
      new = Toplevel(root)
      new.title("Details")
      textbox = Text(new, height = 20, width = 60)
      table = print_table()
      textbox.insert(INSERT, table)
      textbox.config(state = "disabled")
      textbox.grid(row = 0)

def add_record():
      new = Toplevel(root)
      new.title("Add Record")
      new.resizable(False, False)

      bus_no = StringVar(new)
      origin = StringVar(new)
      dest = StringVar(new)
      rate = StringVar(new)
      km = StringVar(new)

      label_bus_no = Label(new, text = "BusNo")
      entry_bus_no = Entry(new, textvariable = bus_no)
      label_origin = Label(new, text = "Origin")
      entry_origin = Entry(new, textvariable = origin)
      label_dest = Label(new, text = "Dest")
      entry_dest = Entry(new, textvariable = dest)
      label_rate = Label(new, text = "Rate")
      entry_rate = Entry(new, textvariable = rate)
      label_km = Label(new, text = "KM")
```

```
        entry_km = Entry(new, textvariable = km)

        def insert_value_db():
              nonlocal bus_no, origin, dest, rate, km
              b = int(bus_no.get())
              o = origin.get()
              d = dest.get()
              r = int(rate.get())
              k = int(km.get())
              sql = f"insert into BUS values({b}, '{o}', '{d}', {r}, {k})"
              cursor.execute(sql)
              connectMySQL.commit()
              messagebox.showinfo("Information", "Record added successfully!")
              new.destroy()

        label_bus_no.grid(row = 0, column = 0)
        entry_bus_no.grid(row = 0, column = 1)
        label_origin.grid(row = 1, column = 0)
        entry_origin.grid(row = 1, column = 1)
        label_dest.grid(row = 2, column = 0)
        entry_dest.grid(row = 2, column = 1)
        label_rate.grid(row = 3, column = 0)
        entry_rate.grid(row = 3, column = 1)
        label_km.grid(row = 4, column = 0)
        entry_km.grid(row = 4, column = 1)
        button_submit = Button(new, text = "Submit", command = insert_value_db)
        button_submit.grid(row = 5, columnspan = 2)

root = Tk()
root.title("Bus Records")

label1 = Label(root, text = "Bus Records")
button_show_details = Button(root, text = "Details", command = show_details)
button_add_record = Button(root, text = "New Record", command = add_record)

label1.grid(row = 0, columnspan = 2)
button_show_details.grid(row = 1, column = 0)
button_add_record.grid(row = 1, column = 1)
root.mainloop()
```