

**ABUDHABI INDIAN SCHOOL**

**BRANCH-1, AL WATHBA**

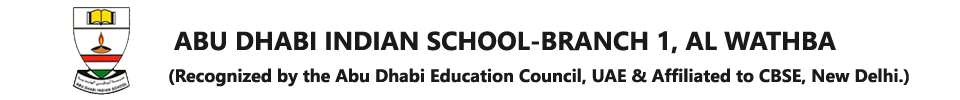
**COMPUTER SCIENCE**

**PRACTICAL FILE**

**NAME\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_KULDEEP\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**CLASS \_\_12 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ SECTION \_\_\_\_\_\_\_\_\_\_\_\_\_A\_\_\_\_\_\_\_\_\_\_\_\_**

**EXAM NO. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

 **ABUDHABI INDIANSCHOOL,**

**BRANCH-1, AL WATHBA**

**CERTIFICATE**

Certified that the work entered in this file is the bonafide work of Master/Miss.\_KULDEEP\_ of Class \_\_\_\_\_12\_\_\_ Section\_\_\_\_\_\_\_A\_\_\_\_, Examination N0. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**,** during theacademic year 2020-2021.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Principal Teacher In-Charge

Date of Examination:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Internal Examiner :\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

External Examiner :\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sl No**  **INDEX** | **Programs** | **Page No** | **Date** | **Signature** |
| **PROGRAMMING WITH PYTHON** | | | | |
| 1 | Program using Conditional Statements | 1 | 9:04:2020 |  |
| 2 | Program using Random Module | 5 | 23-04-2020 |  |
| 3 | Program using String and its functions | 7 | 30-04-2020 |  |
| 4 | Program using List / Array | 10 | 7-05-2020 |  |
| 5 | Program using Dictionary | 12 | 21-05-2020 |  |
| 6 | Program using Tuple | 14 | 28-05-2020 |  |
| 7 | Program using Function | 16 | 4-06-2020 |  |
| 8 | Program using Text File | 18 | 11-06-2020 |  |
| 9 | Program using Text File | 21 | 11-06-2020 |  |
| 10 | Program using Text File | 23 | 18-06-2020 |  |
| 11 | Program using Text File | 26 | 18-006-2020 |  |
| 12 | Program using Binary File | 29 | 25-06-2020 |  |
| 13 | Program using Binary File | 34 | 6-08-2020 |  |
| 14 | Program using CSV File | 37 | 13-08-2020 |  |
| 15 | Program using CSV File | 39 | 20-08-2020 |  |
| 16 | Stack as List- PUSH, POP | 42 | 27-08-2020 |  |
| 17 | Database Connectivity programs: Python with SQL | 45 | 3-09-2020 |  |
| 18 | Database Connectivity programs: Python with SQL | 48 | 10-09-2020 |  |
| 19 | Database Connectivity programs: Python with SQL | 51 | 17-09-2020 |  |
| 20 | Database Connectivity programs: Python with SQL | 54 | 24-09-2020 |  |

**PROGRAM 1**

**DATE: 9:04:2020**

**AIM:**

|  |
| --- |
| **Create a menu driven program to calculate the area of the following shapes based on user’s choice:**  **a. RECTANGLE**  **b. SQUARE**  **c. TRIANGLE**  **d. PARALLELOGRAM**  **e. CIRCLE** |

Concept used:

Conditional Statements

(if..elif..)

Program code:

ch='yes'

print('''to calculate area:

A.RECTANGLE

B.SQUARE

C.TRIANGLE

D.PARALLELOGRAM

E.CIRCLE''')

#area of rectangle

def rectangle():

l=float(input('enter the length'))

b=float(input('enter the breadth'))

return('area of the rectangle is',l\*b)

#AREA OF SQUARE

def square():

s=float(input('enter the side of the square'))

return('the area of the square is',s\*s)

#area of triangle

def triangle():

h=float(input('enter the height'))

b=float(inpu('enter the breadth'))

return('the area of triangle is ',(1/2)\*(l\*b))

#area of parallelogram

def parallelogram():

b=float(input('enter the length of the base'))

h=float(input('enter the height '))

return('the area of parallelogram is',b\*h)

#area of circle

def circle():

r=float(input('enter the radius'))

return('the area of circle is',3.14\*r\*r)

while ch=='yes':

print('which shapes area do want to calculate??' )

a=input('enter the option ')

if a=='A':

print(rectangle())

elif a=='B':

print(square())

elif a=='C':

print(triangle())

elif a=='D':

print(parallelogram())

elif a=='E':

print(circle())

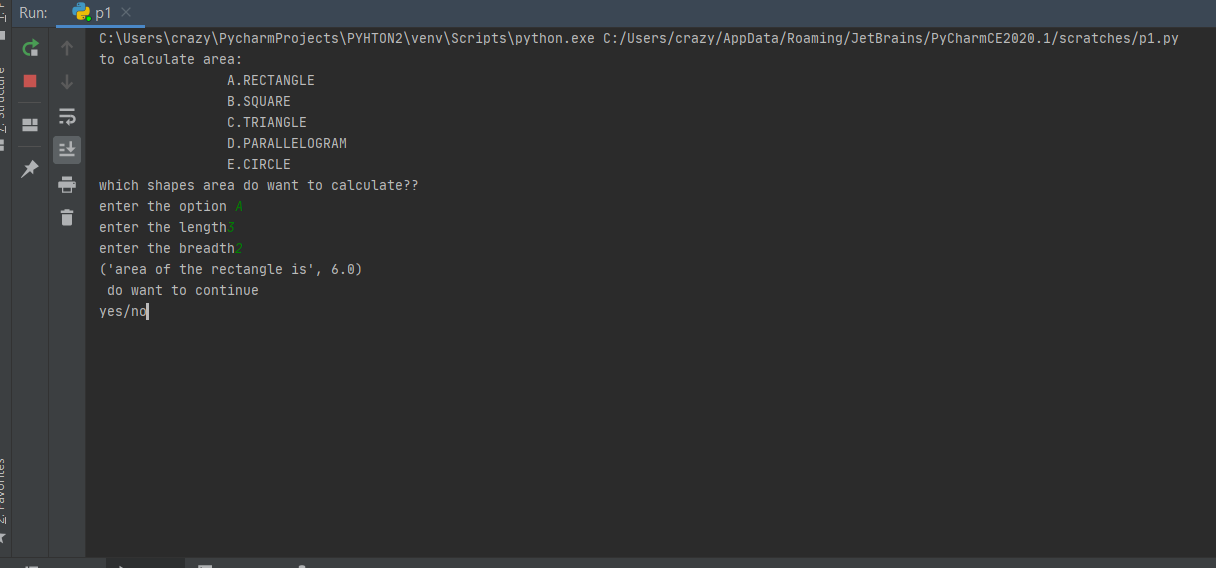
print(' do want to continue ')

ch=input('yes/no')

if ch=='no':

print('thank you for using this , you have now exited the menu')

Sample output



**Program no. 2**

**DATE: 23:04:2020**

**Aim:**

Create Number Guessing Game in Python using random module.

Concept used:Random module

**Program code:**

import random

def game():

a=int(input('enter your guess from 1-15'))

x=random.randint(1,15)

if x==a:

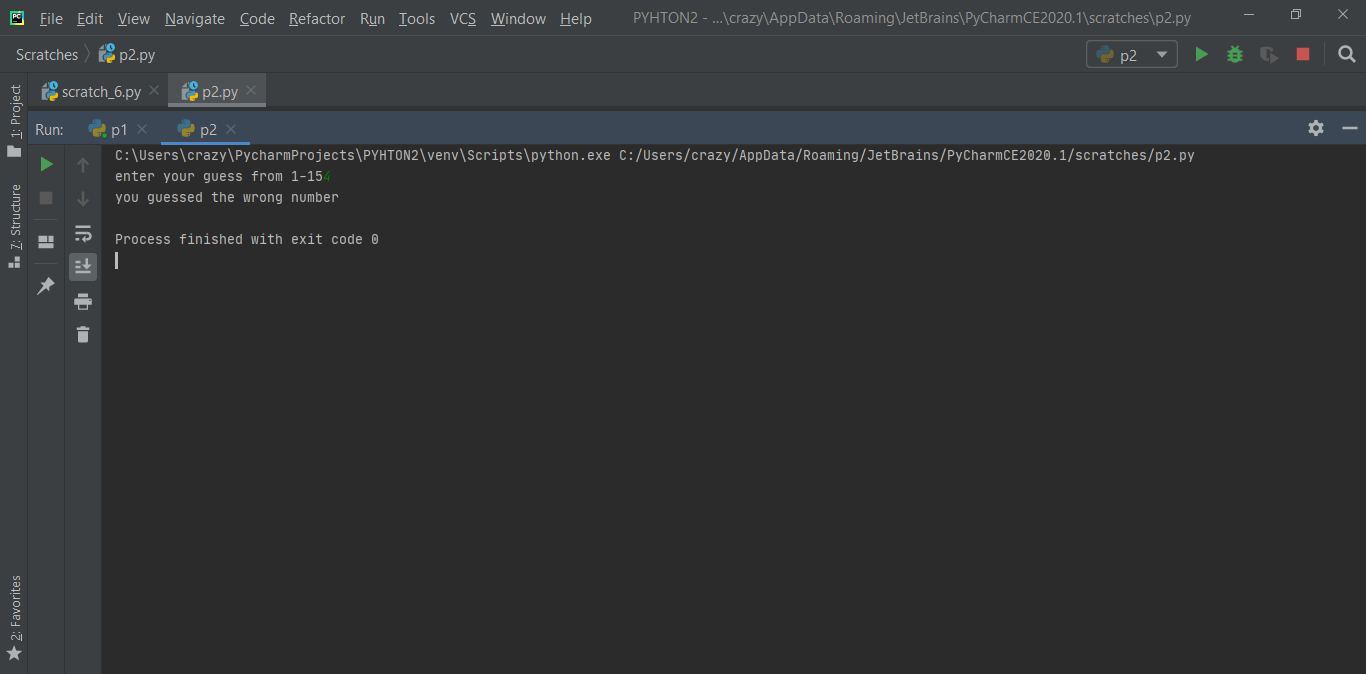
print('you guessed the right number')

else:

print('you guessed the wrong number')

game()

SAMPLE OUTPUT



**Program 3:**

**DATE:30:04:2020**

Aim:

Create a Python program (menu driven) to accept a string from the user and check whether it is palindrome or not. Also display number of characters, no of words, no. of lower case letters, no. of uppercase letters and no. of digits in the string.

**Concept used:**

String and its functions

**Program code:**

dh='yes'

#palindrome

def palindrome(a):

if a[0::]==a[::-1]:

return('it is a palindrome')

else:

return('it is not a palindrome')

while dh=='yes':

a=input('enter the string')

print(palindrome(a))

u=0

l=0

n=0

w=0

s=0

c=0

for i in a:

if i.isupper():

u=u+1

elif i.islower():

l=l+1

elif i.isdigit():

n=n+1

print('the number of upper case letters are',u)

print('the number of lower case letters are',l)

print('the number of digits are',n)

print('no. of characters',len(a))

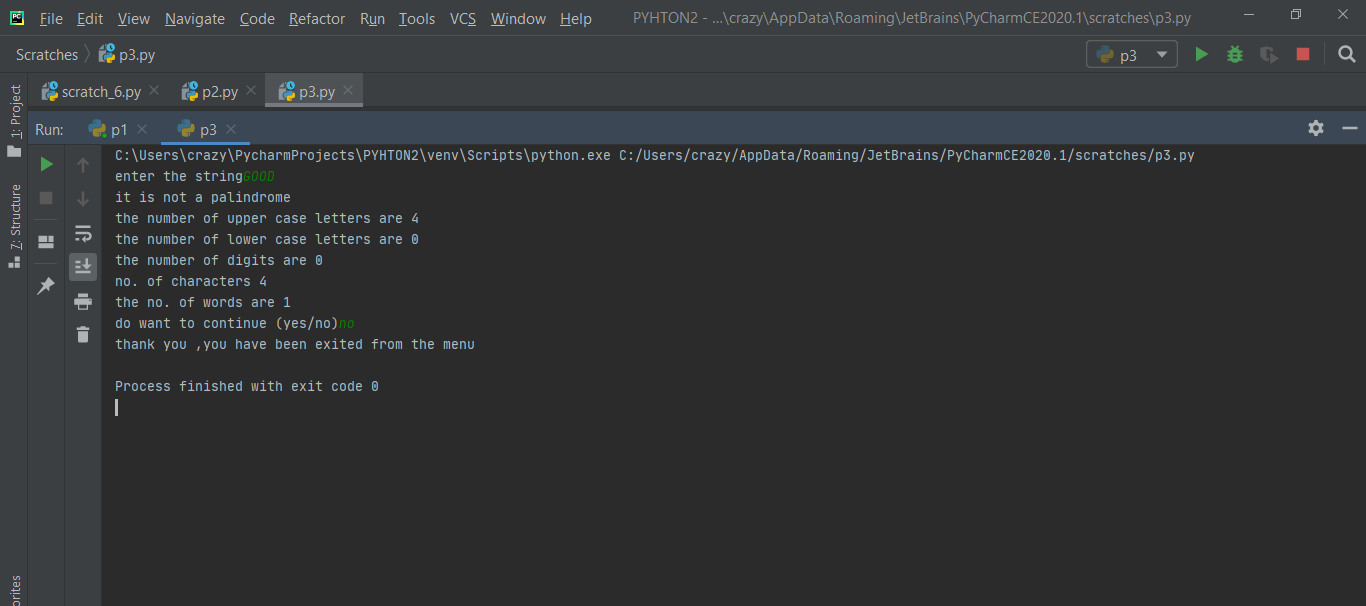
print('the no. of words are',len(a.split()))

dh=input('do want to continue (yes/no)')

if dh=='no':

print('thank you ,you have been exited from the menu'

SAMPLE OUTPUT;



**Program 4:**

**DATE: 7:05:2020**

**Aim:**

Create a Python program to accept a list of numbers from the user then display the highest number and lowest number. Also display the reverse of the list.

**Concept used:**

List / Array

**Program code:**

a=eval(input('enter a list of numbers'))

c=0

l=0

h=0

for i in a:

if i>h:

h=i

l=h

elif i<l:

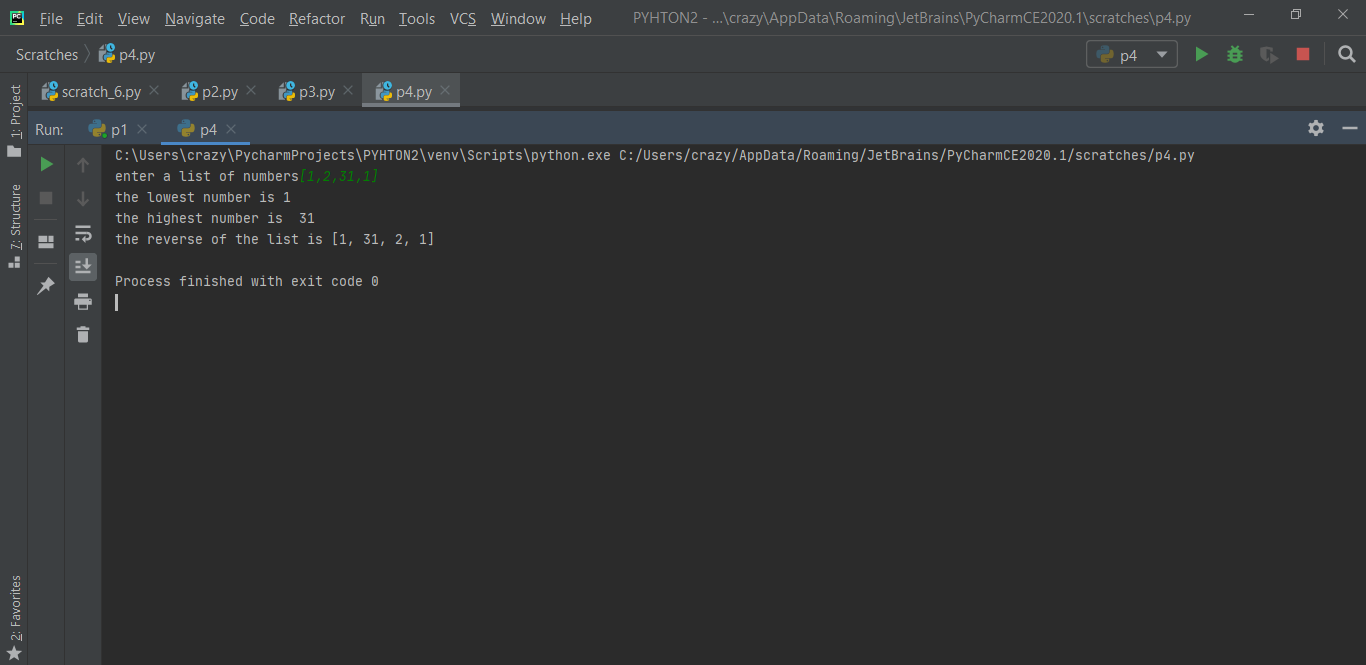
l=i

print('the lowest number is',l)

print('the highest number is ',h)

print('the reverse of the list is',a[::-1])

Sample output:



**Program 5:**

**DATE: 21:05:2020**

**Aim:**

Create a dictionary whose keys are month names and whose values are number of days in the corresponding months.

Ask the user to enter a month name and use the dictionary to display the no. of days in the month.

Display all the keys in Alphabetical order.

Display all the months with 31 days.

Display all key value pairs sorted by the number of days in each month.

**Concept used:**

Dictionary

**Program code:**

dict={'january':31,'february':28,'march':31,'april':30,'may':31,'june':30,'july':31,"august":31,"september":30,'october':31,'november':30,'december':31}

a=input('enter the month')

print(dict[a])

print('THE ALPHABETICALLY SORTED KEYS ARE',sorted(dict))

for i in dict:

if dict[i]==31:

print(i)

b=[]

c=[]

d=[]

for i in dict:

if dict[i]==31:

b.append(i)

elif dict[i]==30:

c.append(i)

else:

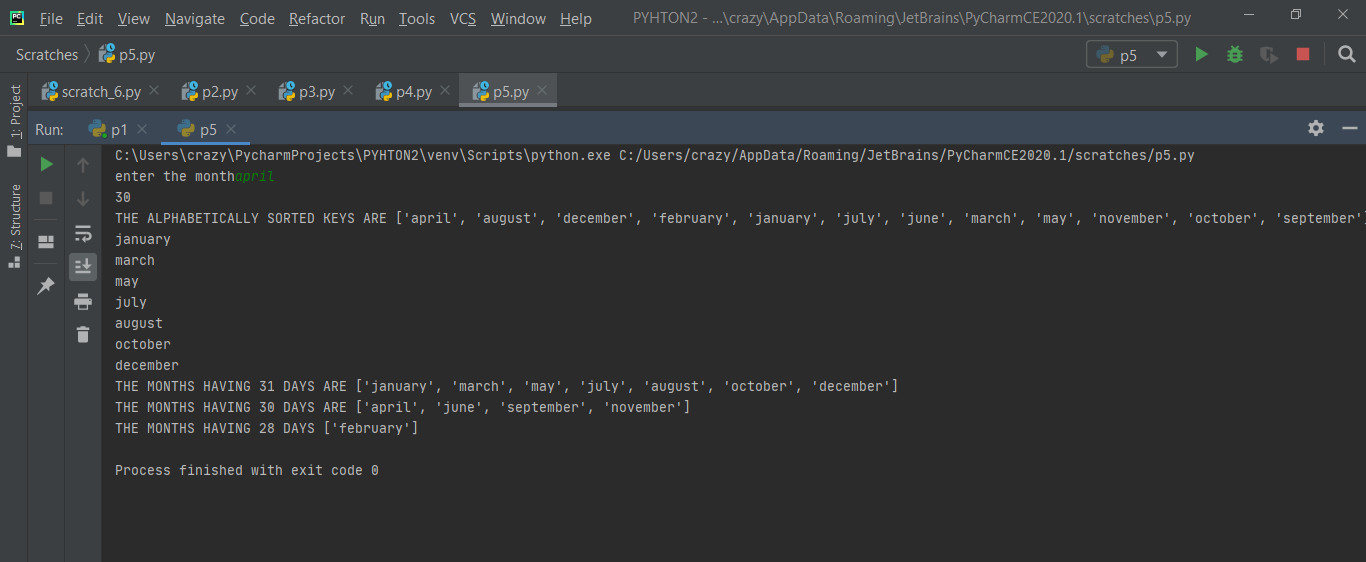
d.append(i)

print('THE MONTHS HAVING 31 DAYS ARE',b)

print('THE MONTHS HAVING 30 DAYS ARE',c)

print('THE MONTHS HAVING 28 DAYS',d)

SAMPLE OUTPUT:



**PROGRAM NO.6**

**DATE: 28:05:2020**

**AIM:**

**Write a Python program to create a tuple for storing the first 12 terms of**

**Fibonacci series.**

**CONCEPT USED:**

**Tuple**

**PROGRAM CODE:**

a=0

b=1

o=()

d=[]

for i in range(1,13):

t=a+b

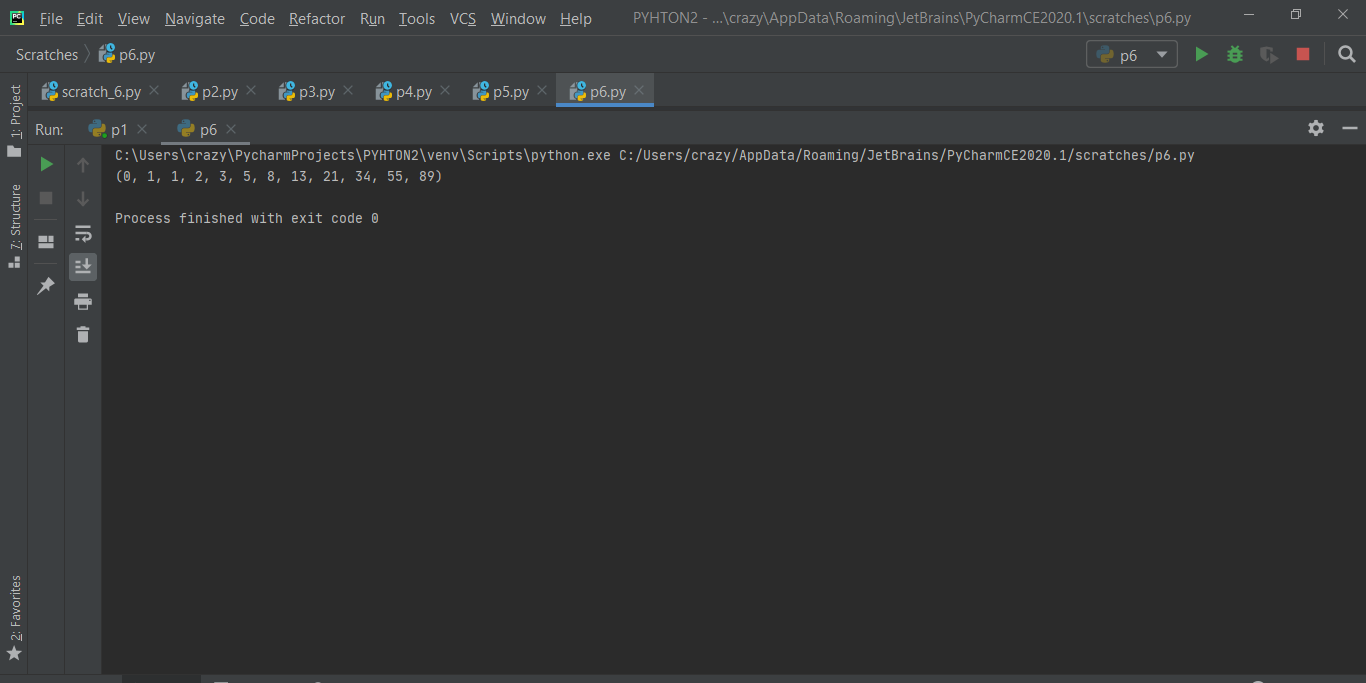
d.append(a)

a=b

b=t

print(tuple(o)+tuple(d))

SAMPLE OUTPUT:



**PROGRAM NO.7**

**DATE: 4:06:2020**

**AIM:**

**Create a Python program to have the following functions:**

**a. A function that takes a number as an argument and calculates its cube.**

**If there is no value passed to the function, it should calculate the cube of**

**5.**

**b. A function that receives two strings and checks whether they are the same**

**length string or not. It should return True if both are the same length string**

**otherwise False.**

**CONCEPT USED:**

Function

**PROGRAM CODE:**

def cube():

b=input('enter the number whose cube is required')

if b=='':

b='5'

return(int(b)\*\*3)

def string():

a=input('enter a string')

c=input('enter a string')

if len(a)==len(c):

return('true')

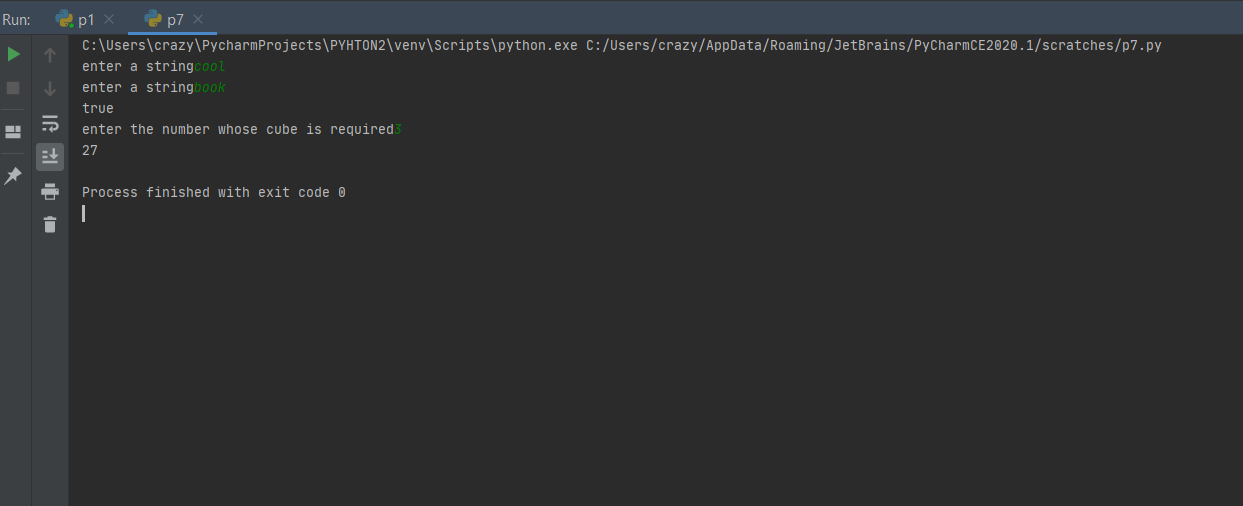
else:

return('false')

print(string())

print(cube())

SAMPLE OUTPUT



**PROGRAM 8:**

**DATE: 11:6:2020**

**AIM:**

**Write a program to create a text file “Story.txt” and do the following:-**

**a) Display the number of blanks present in the file.**

**b) Display the number of lines present in the file.**

**c) Display the number of capital letters present in the file.**

**d) Display the number of words present in the file.**

**e) Display the number of lowercase letters present in the file.**

**Concept used:**

Text File

**PROGRAM CODE:**

try:

file = open('C:\\Users\\crazy\\Desktop\\Stories.txt', 'r')

u = 0

lo = 0

b = 0

s = file.read()

file.seek(0)

d = file.readlines()

for i in s:

if i == ' ':

b = b + 1

elif i.isupper():

u = u + 1

elif i.islower():

lo = lo + 1

print('the number of words are', len(s.split()))

print('the number of blank space are', b)

print('the number of lines are', len(d))

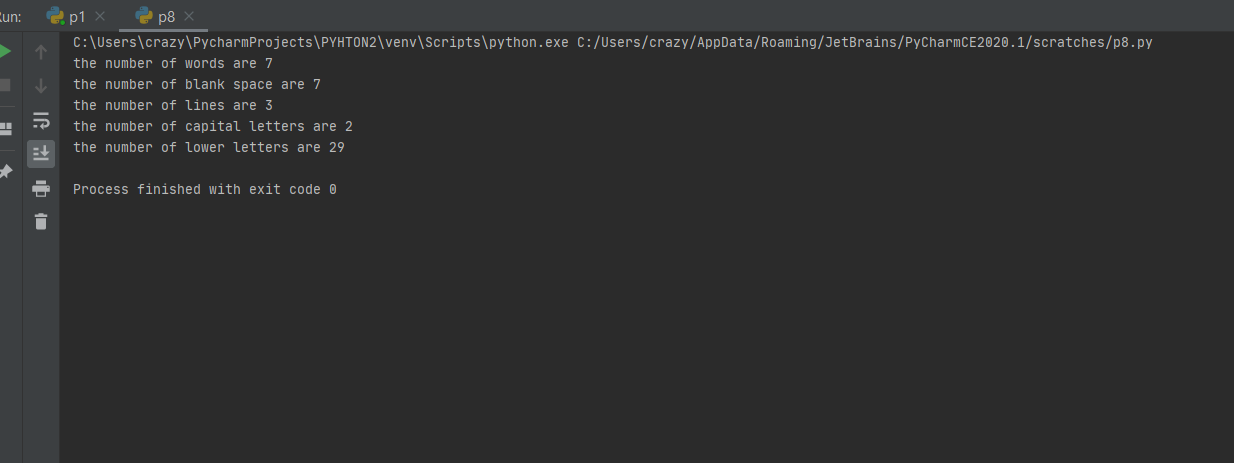
print('the number of capital letters are', u)

print('the number of lower letters are', lo)

except FileNotFoundError:

print('file not found')

Sample output



**PROGRAM CODE 9**

**DATE: 11:06:2020**

**AIM:**

**Write a program to read the text file “Poem.txt” and display it content after**

**replacing every space by #.**

**CONCEPT USED**

TEXT FILE

**PROGRAM CODE:**

try:

with open('C:\\Users\\crazy\\Desktop\\poem.txt', 'r') as f:

a = f.read()

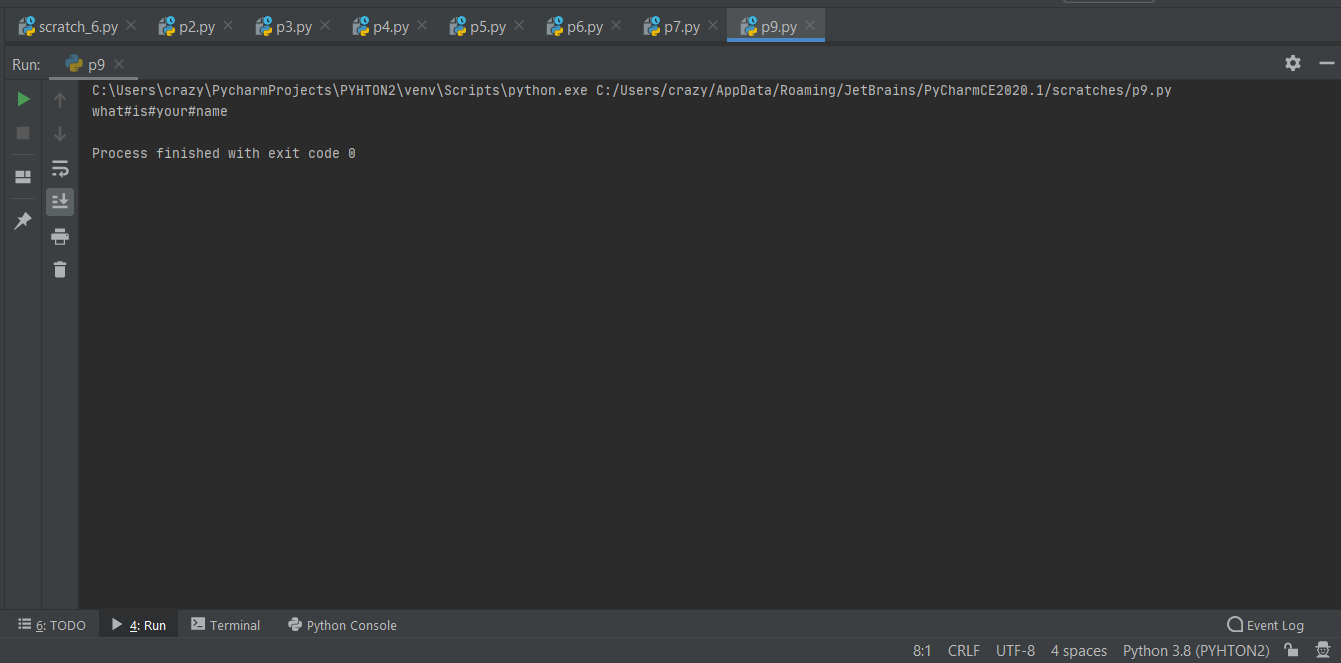
d = a.split()

print('#'.join(d))

except FileNotFoundError:

print("file not found")

**SAMPLE OUTPUT:**

****

**PROGRAM 10:**

**DATE: 18:06:2020**

**AIM:**

Create a text file “COVID.TXT” with some instructions to prevent yourself

from COVID virus. Write a function/method in python to read the content of

the file “COVID.TXT” to count and display the occurrence of those words

which are having 5 or more alphabets.

**CONCEPT USED:**

TEXT FILE

**PROGRAM CODE:**

try:

file = open('C:\\Users\\crazy\\Desktop\\COVID.txt', 'r')

e = []

def read():

print(file.read())

file.seek(0)

d = file.read()

for i in d.split():

if len(i) >= 5:

e.append(i)

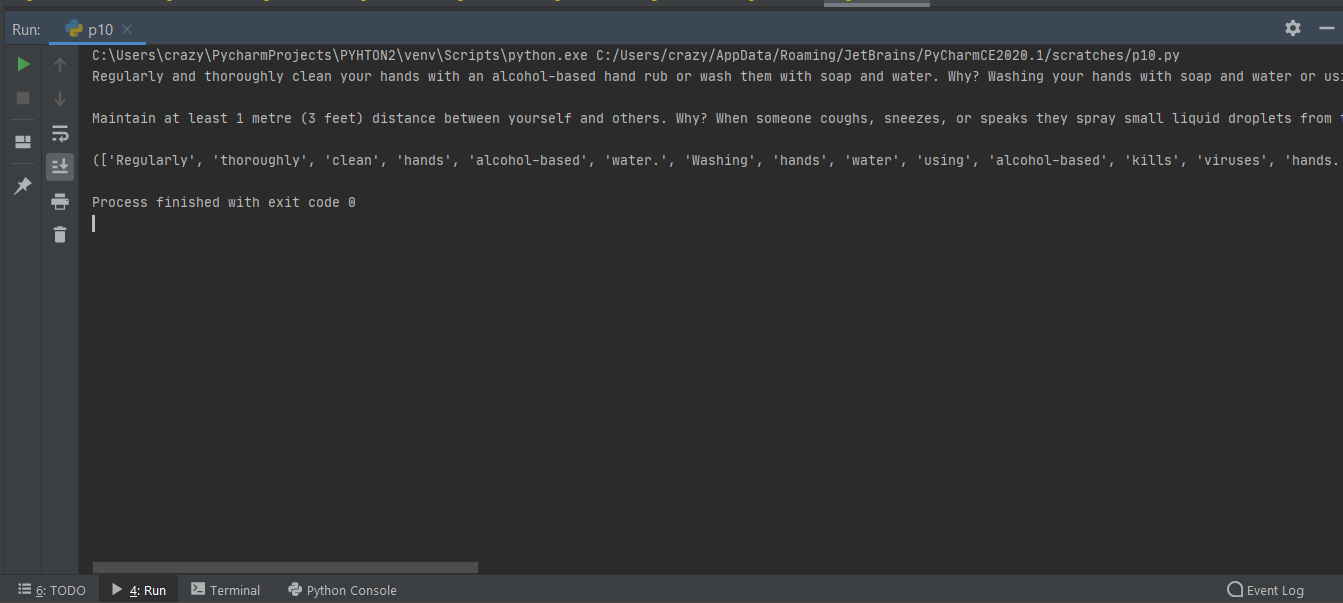
return (e, 'the number of words are', len(e))

print(read())

except FileNotFoundError:

print('file is not found')

Sample output:



**Program no.11**

**Date:18:06:2020**

**Aim:**

Create a text file “My School.TXT” with some information regarding your

school. Read the content of the file “My School.TXT” and display the

following:

No. of characters in the file.

Number of vowels present in the file.

Number of consonants present in the file.

Number of words start with “A”

**Concept used:**

Text file

**Program code:**

f= open('C:\\Users\\crazy\\Desktop\\MySchool.txt','r')

def read():

print(f.read())

f.seek(0)

d=f.read()

s=d.split()

V=0

c=0

co=0

for i in s:

for j in i:

if j in ('a','e','i','o','u','A','E','O','U','I'):

V=V+1

elif j not in ('a','e','i','o','u','A','E','O','U','I'):

c=c+1

if i[0]=='A':

co=co+1

print('the total number of characters are',len(d))

print('the number of vowels are ',V)

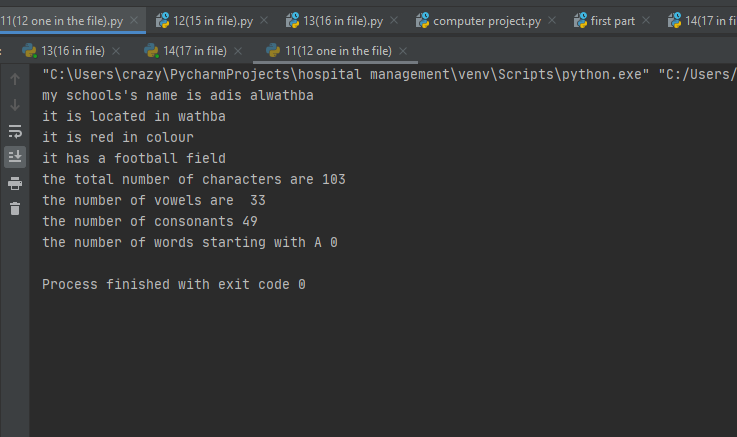
print('the number of consonants',c)

print('the number of words starting with A',co)

**read()**

**f.close()**

**Sample output:**

****

**PROGRAM 12**

**DATE: 25:06:2020**

**AIM**

Write an interactive menu driven program to perform the following

operations on a data file “UAE.dat” which stores the information such as

Emirates\_Id, Name, and Visa Number.

1. Add a record

2. Modify record for a given Emirates\_ID

3. Search record using a particular Emirates\_ID

4. Display records

**CONCEPT USED:**

BINARY FILES

**PROGRAM CODE:**

import pickle

def arecord():

try:

with open("UAE.dat",'rb+') as f:

eid=int(input('enter the emirates id'))

nm=input("enter the name ")

v\_id=int(input('enter the visa id'))

l=[eid,nm,v\_id]

pickle.dump(l,f)

except FileNotFoundError:

print('file not found')

def meid():

e=int(input('enter the emirates id'))

pos=0

with open("UAE.dat",'rb+') as g:

try:

while True:

rec=pickle.load(g)

if rec[0]==e:

g.seek(pos)

rec[0]=int(input('enter the new emirates id'))

rec[1]=input("enter the name")

rec[2]=int(input('enter the new visa id'))

pickle.dump(rec,g)

print('record modified')

break

else:

pos=g.tell()

except EOFError:

print('...record not found')

def search():

e=int(input('enter the emirates id'))

with open("UAE.dat",'rb') as k:

try:

while True:

rec=pickle.load(k)

if rec[0]==rec:

print('record found',rec)

break

except EOFError:

print('record not found')

def display():

with open('UAE.dat','rb') as y:

rec=pickle.load(y)

for i in rec:

print(i)

print(' menu ')

print('''

what do you want to do:

1.add record

2.modify record for a give emirates\_id

3.search record using a particular emirates id

4.display record

5.exit

''')

while True:

ch=int(input("enter your choice"))

if ch==1:

arecord()

elif ch==2:

meid()

elif ch==3:

search()

elif ch==4:

display()

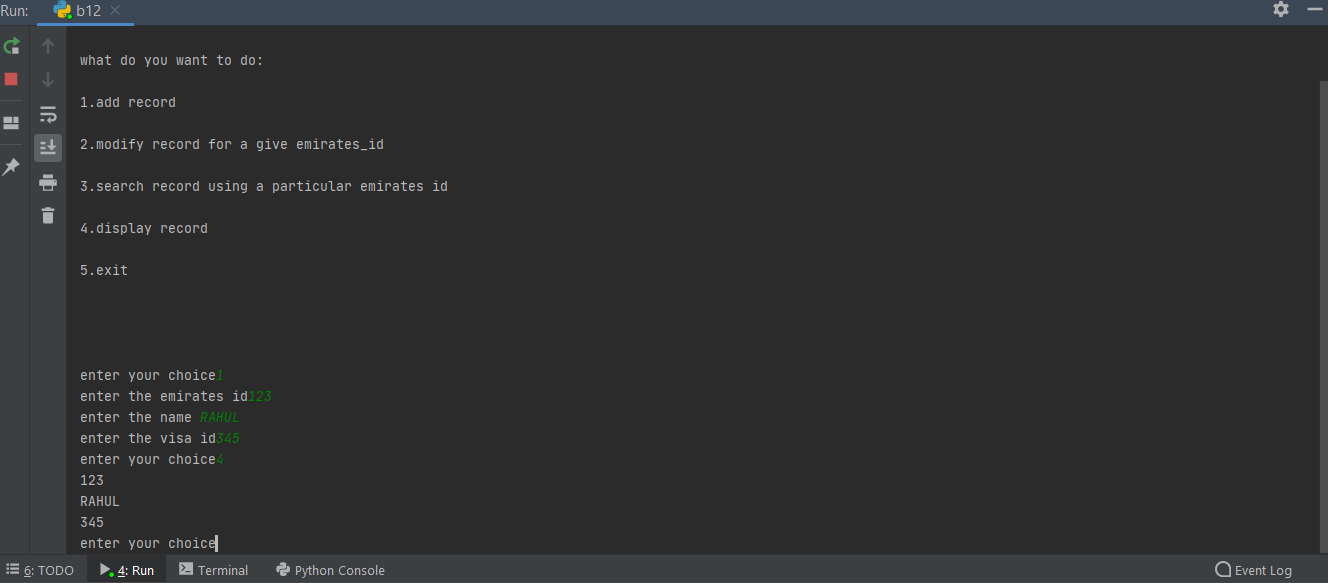
elif ch==5:

break

else:

print('invalid option')

**SAMPLE OUTPUT:**

****

**PROGRAM 13:**

**DATE:6-08-2020**

**AIM**

Create a binary file “Sports.dat” to store the information of sports event in

the following format : Event – Participant Name.

Write a function that would read the content from the file “Sports.dat” and

create a file named “Athletics.dat” copying only those record from

Sports.dat where the event name is “Athletics”.

**CONCEPT USED**

Binary files

**PROGRAM CODE:**

import pickle

def t():

h=0

with open('Sports.dat','rb') as f:

with open('Athletics.dat','wb') as g:

try:

while True:

rec=pickle.load(f)

if rec[0]=="Athletics":

h=1

pickle.dump(rec,g)

print('record transferred')

except EOFError:

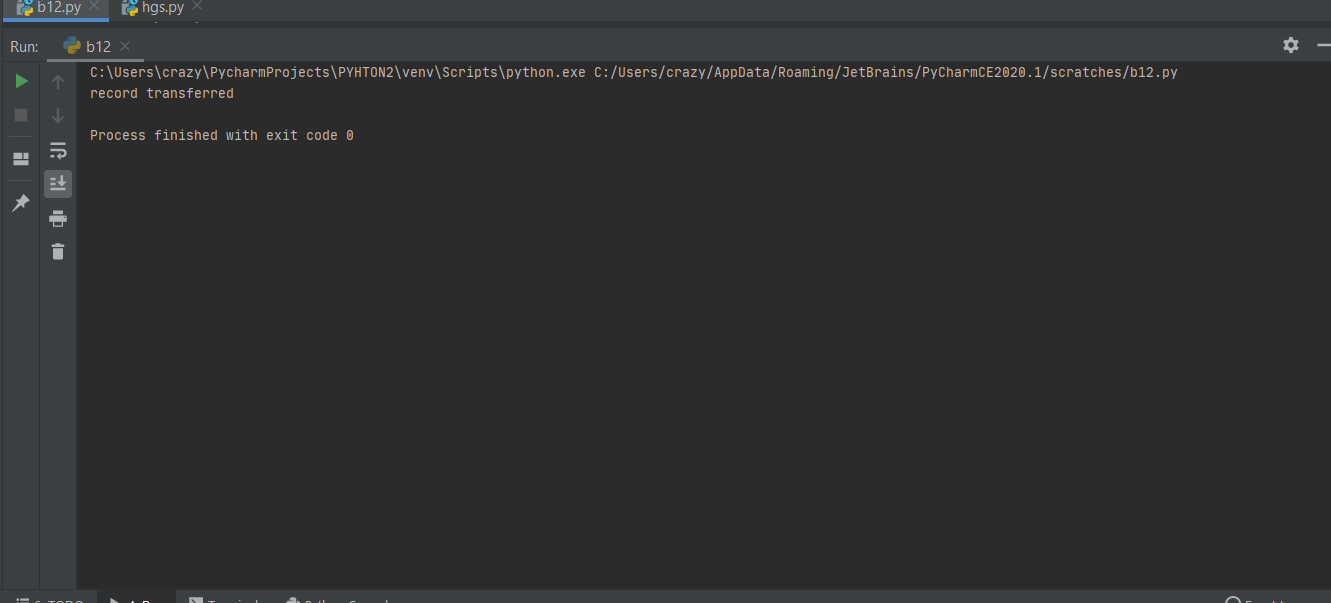
pass

if h == 0:

print('the given record is not found')

t()

**SAMPLE OUTPUT:**

****

**Program no.14**

**DATE:13-08-2020**

**Aim:**

Create a CSV file called “contact.csv” to store name and contact number of

any 10 students. Write a function to display the content of the file.

**CONCEPT USED**

Csv files

**Program code:**

import csv

def dispcontact():

with open('contact.csv','r',newline='') as f:

s=csv.reader(f)

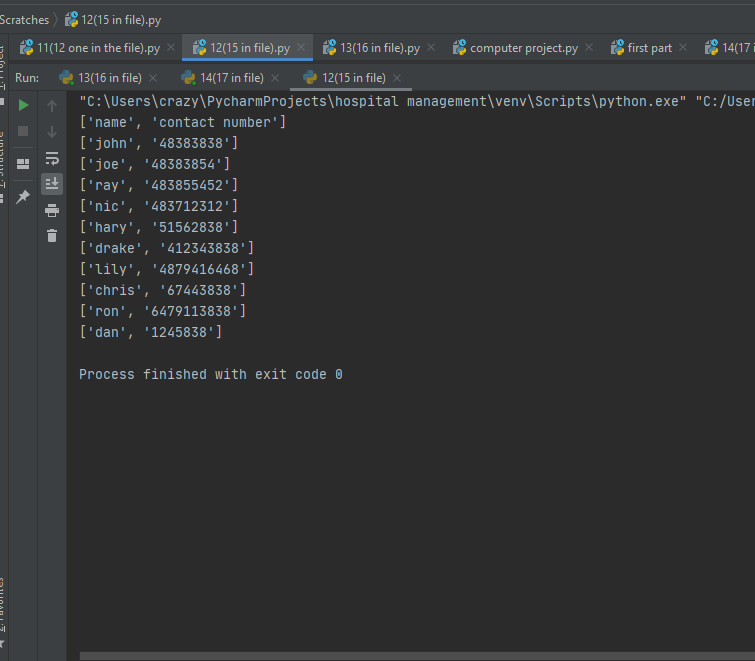
for i in s:

print(i)

next(f)

dispcontact()

**Sample output:**

****

**Program no.15**

**Date:20-08-2020**

**Aim:**

Create a CSV file “books.csv” to store BOOKID, BOOKNAME, AUTHOR and

PRICE as a tabular form: Write it as a menu driven program to do the

following:-

1. Write new row

2. Read all rows

3. Search a book information based on given bookid

**CONCEPT USED:**

CSV files

**Program code:**

while ch=='y':

a=input('enter your option')

if a=='1':

with open('C:\\Users\\crazy\\Desktop\\Books.csv','a',newline='') as c:

cf=csv.writer(c)

q=eval(input('enter the new row in form of a list'))

cf.writerow(q)

elif a=='2':

with open('C:\\Users\\crazy\\Desktop\\Books.csv', 'r') as k:

s=csv.reader(k)

next(k)

for i in s:

print(i)

elif a=='3':

i=input('enter the bookid')

with open('C:\\Users\\crazy\\Desktop\\Books.csv', 'r') as h:

g=csv.reader(h)

for j in g:

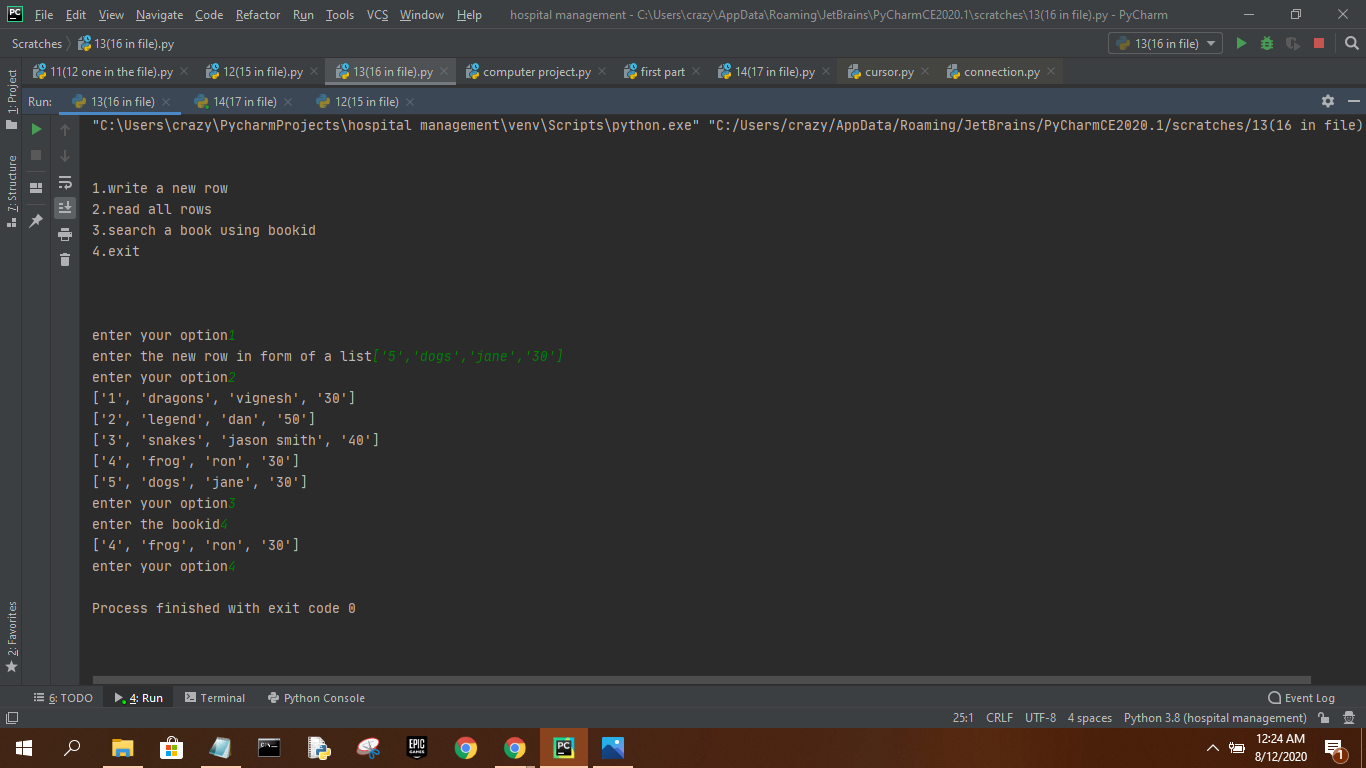
if j[0]==i:

print(j)

elif a=='4':

ch='no'

**Sample output**

****

**PROGRAM 16:**

**Date:27-08-2020**

**AIM**

Write a menu-driven program to implement the following operations on a stack. Each node of the stack is used to store details of items such as Icode, Name and quantity.

1.Push

2. Pop

3. Display

4.Exit

**CONCEPT USED**

STACKS

**PROGRAM CODE**

def push():

i=int(input("enter the icode"))

n=input("enter the name")

q=int(input('enter the quantity'))

n=[i,n,q]

st.append(n)

print('the data is pushed....')

def pop():

if len(st)==0:

print('...underflow...')

else:

print('the data tHATt is popped is',st.pop())

def display():

if len(st)==0:

print('the give stack is empty')

else:

for i in range(len(st)-1,-1,-1):

print(st[i])

st=[]

a='y'1

while a == 'y':

g=int(input('enter your option 1.push 2.pop 3.display 4.exit'))

if g==1:

push()

elif g==2:

pop()

elif g==3:

display()

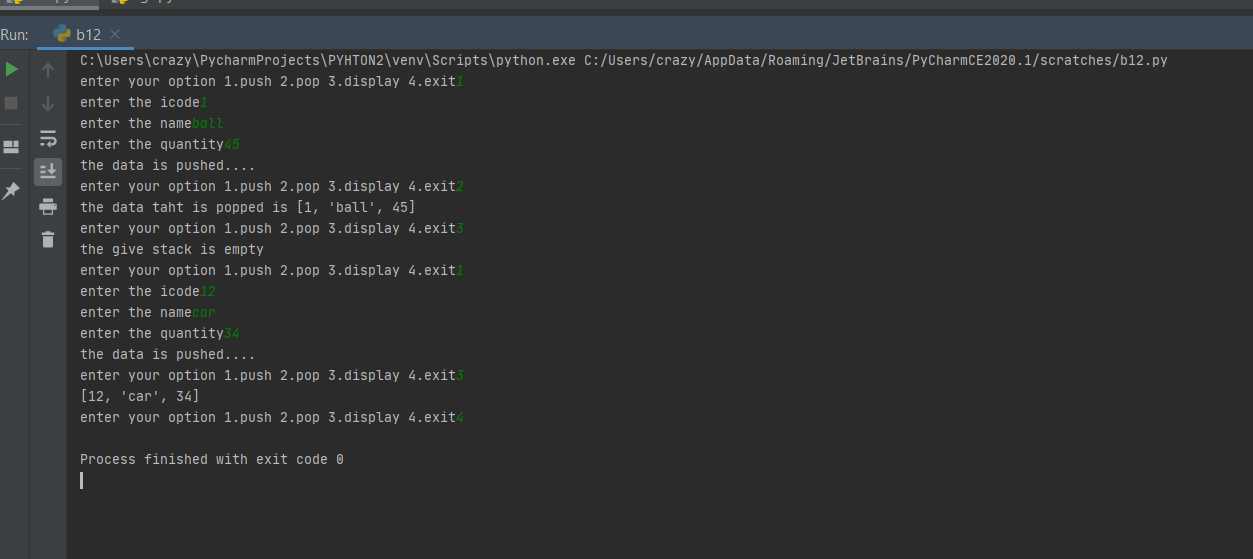
elif g==4:

break

else:

print('invalid option')

**SAMPLE OUTPUT:**

****

**Program no.17**

**Date:3-09-2020**

**Aim:**

Write a menu driven database connectivity program to do the following

transactions between Python and MySQL. (Note: Table Name: STUDENT,

user=”root”, password=”adis123” and database name: ADIS.)

1. Create table STUDENT (GRNO, NAME, CLASS, AGE).

2. Insert a new student record.

3. Display all records

**CONCEPT USED:**

Database connectivity

**Program code:**

crl.execute("create table if not exists student(grno int(20) primary key,name varchar(20),class varchar(20),age int(3))")

def menu():

ch='y'

while ch=='y':

print('''

1.insert new data

2.display all the data

3.exit

''')

a=input('enter your choice')

if a=='1':

grno=int(input('enter the grno'))

name=input('enter the name')

c=input('enter class')

ag=int(input('enter the age'))

q="insert into student(grno,name,class,age)values({},'{}','{}',{})".format(grno,name,c,ag)

ms.commit()

crl.execute(q)

elif a=='2':

crl.execute('select \* from student')

row=crl.fetchall()

for i in row:

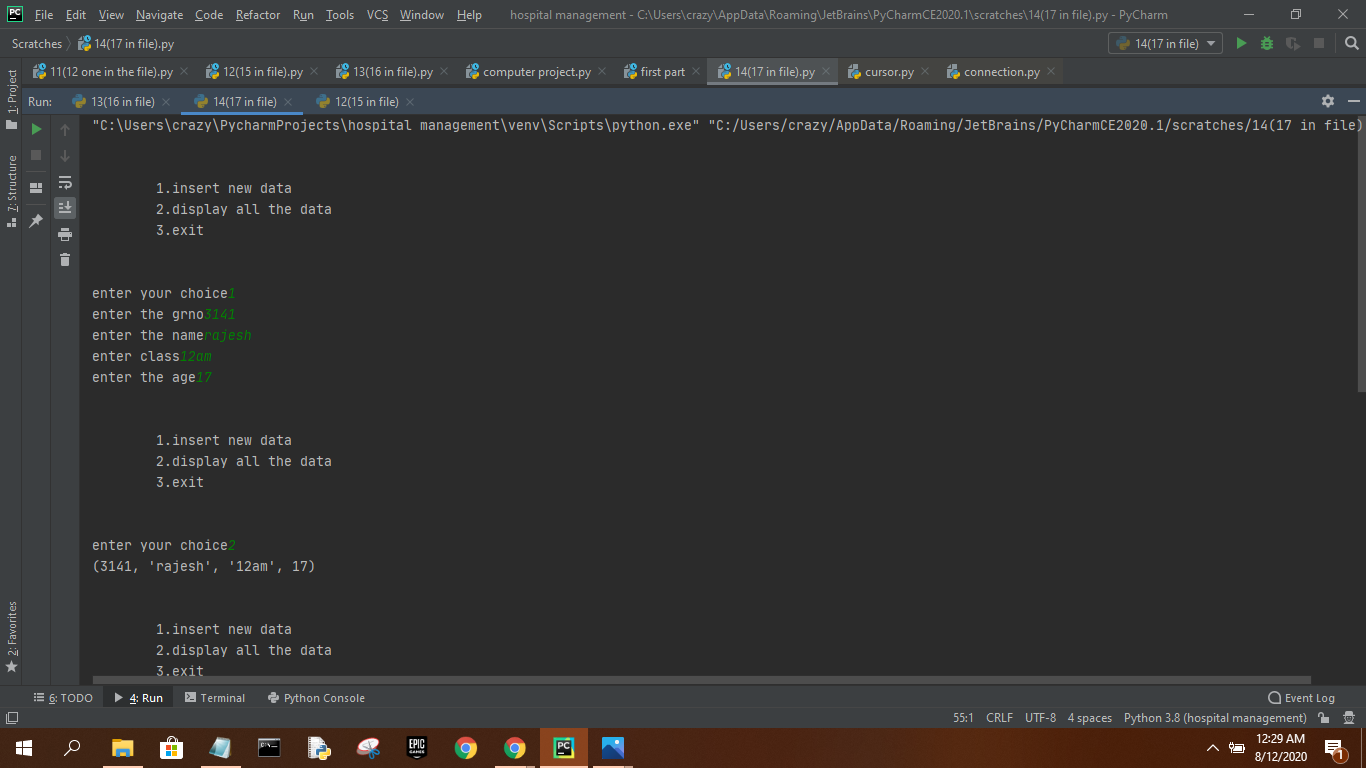
print(i)

else:

ch='no'

menu()

**SAMPLE OUTPUT:**

****

**Program no.18**

**Date:10-09-2020**

**Aim:**

Write a menu driven database connectivity program to do the following

transactions between Python and MySQL. (Note: Table Name: STUDENT,

user=”root”, password=”adis123” and database name: ADIS.)

1. Create table STUDENT(GRNO,NAME,CLASS,AGE).

2. Insert a new student record.

3. Search a student details based on given grno or name.

**Concept used:**

**Database Connectivity**

**Program code:**

import mysql.connector

ms=mysql.connector.connect(host='localhost',user='root',password='adis123',)

cr=ms.cursor()

cr.execute('create database if not exists ADIS')

cr.execute('use ADIS')

cr.execute('create table if not exists student(grno int(10) primary key,name varchar(30),class varchar(30),age int(3))')

g=int(input('enter the grno'))

n=input('enter the name')

c=input('enter the class')

a=int(input('enter the age'))

cr.execute("insert into student(grno,name,class,age)values({},'{}','{}',{})".format(g,n,c,a))

print('''1.search by grno.

2. search by name''')

i=int(input('enter your choice'))

if i == 1:

g=int(input('enter the grno'))

cr.execute('select \* from student where grno={}'.format(g))

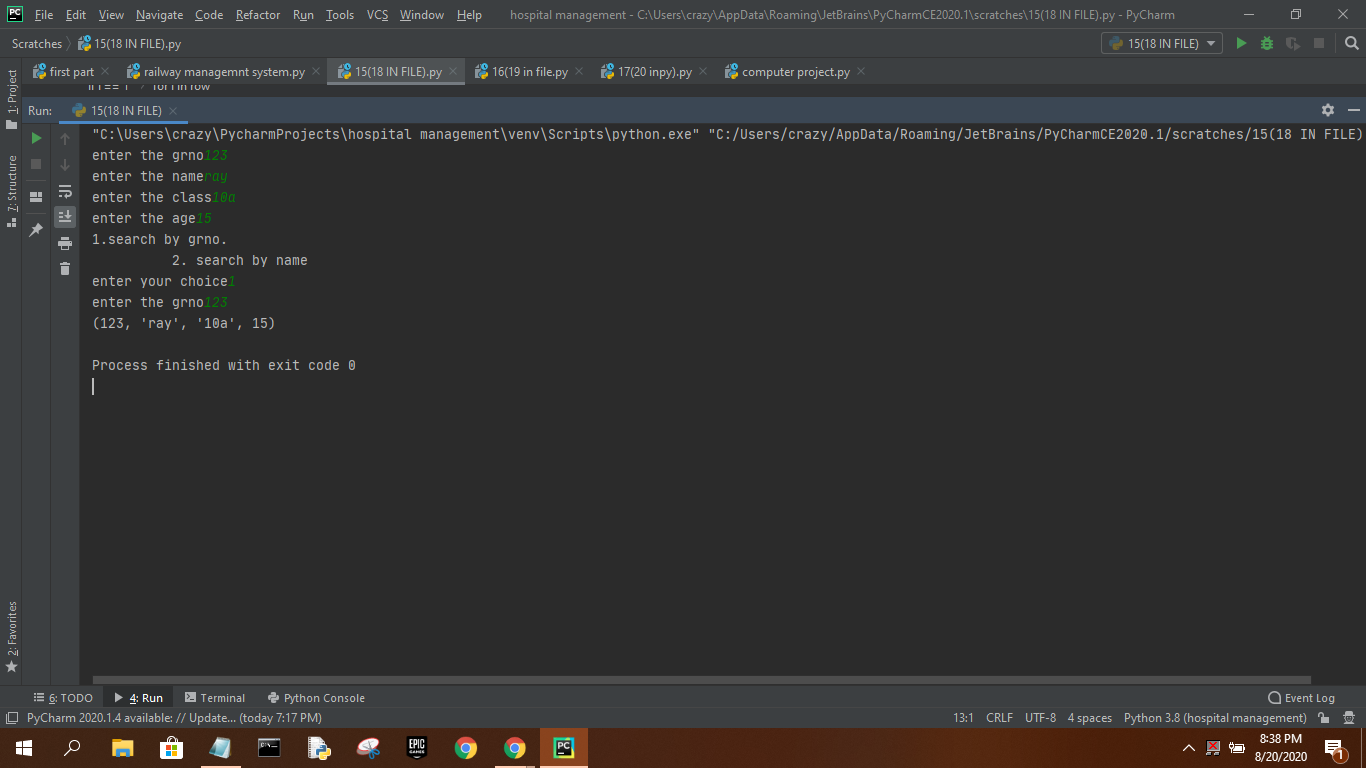
else:

na=input('enter the name')

cr.execute("select \* from student where name='{}'".format(na))

ms.commit()

**Sample output:**

****

**Program no. 19**

**Date:17-09-2020**

**Aim:**

Write a menu driven database connectivity program to do the following

transactions between Python and MySQL. (Note: Table Name: EMP,

user=”root”, password=”adis123” and database name: ADIS.)

1. Create table EMP (EMPNO,ENAME,JOB, SALARY,DEPTNO).

2. Insert a new employee record.

3. Delete an employee record based on given empno.

**Concept used:**

Database Connectivity

**Program code:**

import mysql.connector

ms=mysql.connector.connect(host='localhost',user='root',password='adis123')

cr=ms.cursor()

cr.execute('use ADIS')

cr.execute('create table if not exists EMP(empno int(10) primary key,ename varchar(30),job varchar(30),salary float(30),deptno int(20))')

en=int(input('enter the empno'))

n=input('enter the name')

j=input('enter the job')

s=float(input('enter the salary'))

d=int(input('enter the deptno'))

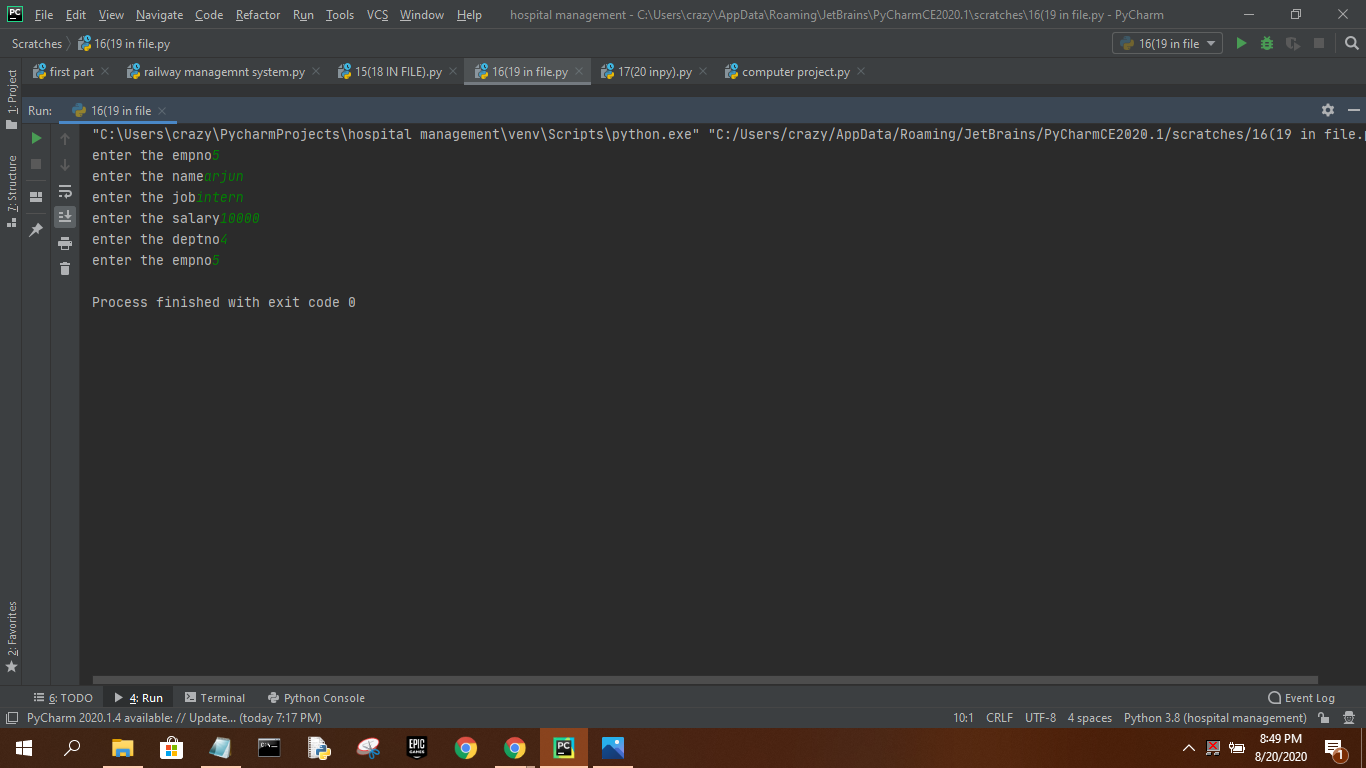
cr.execute("insert into EMP(empno,ename,job,salary,deptno)values({},'{}','{}',{},{})".format(en,n,j,s,d))

eno=int(input('enter the empno'))

cr.execute('delete from EMP where empno={}'.format(eno))

ms.commit()

**Sample output:**



**Program no. 20**

**Date:24-09-2020**

**Aim:**

Write a menu driven database connectivity program to do the following

transactions between Python and MySQL. (Note: Table Name: EMP,

user=”root”, password=”adis123” and database name: ADIS.)

1. Create table EMP (EMPNO,ENAME,JOB, SALARY,DEPTNO).

2. Insert a new employee record.

3. Update an employee record based on the given employee name.

**Concept used:**

Database Connectivity

**Program code:**

import mysql.connector

ms=mysql.connector.connect(host='localhost',user='root',password='adis123')

cr=ms.cursor()

cr.execute('use ADIS')

cr.execute('create table if not exists EMP(empno int(10) primary key,ename varchar(30),job varchar(30),salary float(30),deptno int(20))')

en=int(input('enter the empno'))

n=input('enter the name')

j=input('enter the job')

s=float(input('enter the salary'))

d=int(input('enter the deptno'))

cr.execute("insert into EMP(empno,ename,job,salary,deptno)values({},'{}','{}',{},{})".format(en,n,j,s,d))

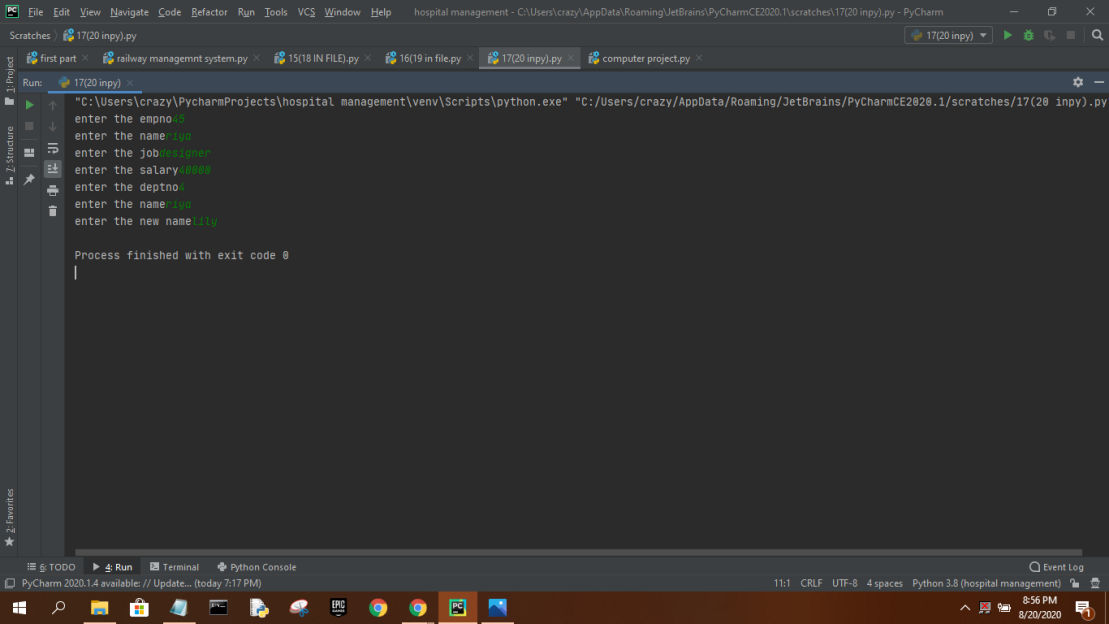
name=input('enter the name')

nname=input('enter the new name')

cr.execute("update EMP set ename='{}' where ename='{}'".format(nname,name))

ms.commit()

**Sample output:**

****