**Chapter 7. Lists and Tuples**

Solutions for Programming Questions

Submitted by: 19251A0421

-------------------------------------------------------------------

1. **Valid Number Information**

**Program:**

'''

Problem Statement:

Design a program that uses a loop to build a

list named valid\_numbers that contains only

the numbers between 0 and 100 from the numbers list below.

The program should then determine

and display the total and average of the values

in the valid\_numbers list.

numbers = [74, 19, 105, 20, −2, 67, 77, 124, −45, 38]

'''

import **math**

numbers=[74,19,105,20,-2,67,77,124,-45,30]

**print**("List of Numbers:",\*numbers)

valid\_numbers=[]    *#output list*

*# Traverse 'numbers' list*

*# if any element is in between 0 to 100, \*

*# add that element into 'valid\_numbers' list*

for i in **range**(**len**(numbers)):

    if numbers[i]>=0 and numbers[i]<=100:

        valid\_numbers.**append**(numbers[i])

*#computing total of all valid numbers*

total=**sum**(valid\_numbers)

*#computing average of all valid numbers*

average=**sum**(valid\_numbers)/**len**(valid\_numbers)

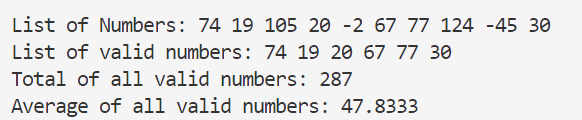
*#display output*

**print**("List of valid numbers:",\*valid\_numbers)

**print**("Total of all valid numbers:",total)

**print**("Average of all valid numbers:",**round**(average,4))

**Output:**

****

**2. Lottery Number Generator**

**Program:**

'''

Problem Statement:

Design a program that generates a seven-digit lottery number.

 The program should generate seven random numbers,

 each in the range of 0 through 9,

 and assign each number to a list element.

 (Random numbers were discussed in Chapter 5.)

Then write another loop that displays the contents of the list.

'''

import **random**

*#randint(m,n) will generate integers from m to n*

lottery\_number=[]

for i in **range**(7):

    ith\_digit=**random**.randint(0,9)

    lottery\_number.**append**(ith\_digit)

**print**("Generated 7 digit Lotter Number is: ",\*lottery\_number)

**Output**:



**3. Rainfall Statistics**

**Program:**

'''

Problem Statement:

Design a program that lets the user enter the total rainfall

for each of 12 months into a list.

The program should calculate and display

the total rainfall for the year,

the average monthly rainfall,

the months with the highest and lowest amounts.

'''

rainfall=[]

months=['January','February','March','April','May',\

    'June','July','August','September',\

        'October','November','December']

**print**("Enter rainfall in mms")

for i in **range**(12):

**print**("Enter rainfall measure in",months[i],end=": ")

    level=**int**(**input**())

    rainfall.**append**(level)

*#the total annual rainfall = sum of rainfall in all twelve months*

total=**sum**(rainfall)

*#the average monthly rainfall = the total annual rainfall / 12*

average=total/12

*#the month with the highest rainfall*

highest\_rainfall=rainfall[0]

highest\_rainfall\_month=months[0]

for i in **range**(1,12):

    if rainfall[i]>highest\_rainfall:

        highest\_rainfall=rainfall[i]

        highest\_rainfall\_month=months[i]

*#the month with lowest rainfall*

lowest\_rainfall=rainfall[0]

lowest\_rainfall\_month=months[0]

for i in **range**(1,12):

    if rainfall[i]<lowest\_rainfall:

        lowest\_rainfall=rainfall[i]

        lowest\_rainfall\_month=months[0]

**print**("The total annual rainfall: ",total,"mm")

**print**("The average monthly rainfall: ",**round**(average,2),"mm")

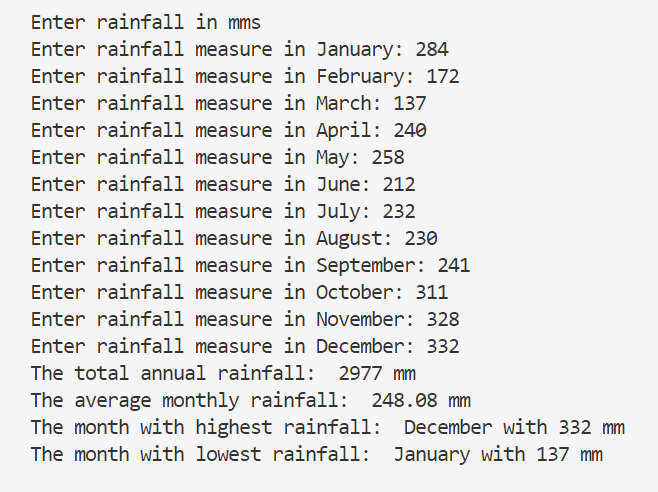
**print**("The month with highest rainfall: "\

    ,highest\_rainfall\_month,"with",highest\_rainfall,"mm")

**print**("The month with lowest rainfall: "\

    ,lowest\_rainfall\_month,"with",lowest\_rainfall,"mm")

**Output:**

****

1. **Number Analysis Program**

**Program:**

'''

Problem Statement:

Design a program that asks the user to enter a series of 20 numbers.

 The program should store the numbers in a list

 then display the following data:

• The lowest number in the list

• The highest number in the list

• The total of the numbers in the list

• The average of the numbers in the list

'''

**print**("Enter Series of 20 Numbers: ")

arr=[]

for i in **range**(20):

    arr.**append**(**int**(**input**()))

lowest=**min**(arr)

highest=**max**(arr)

total=**sum**(arr)

average=total/12

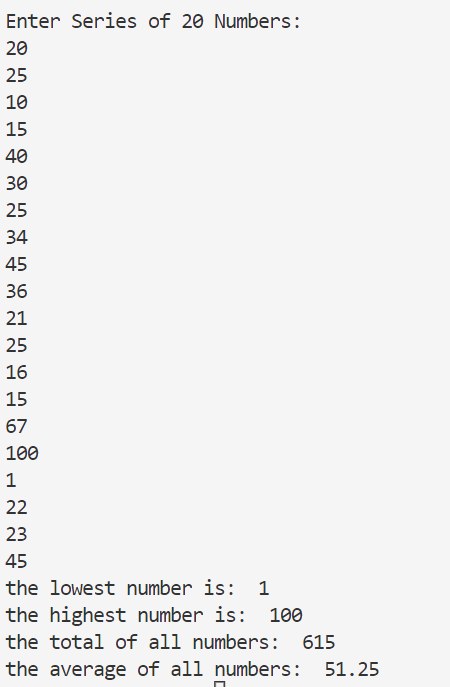
**print**("the lowest number is: ",lowest)

**print**("the highest number is: ",highest)

**print**("the total of all numbers: ",total)

**print**("the average of all numbers: ",average)

**Output:**

****

1. **Charge Account Validation**

**Program:**

'''

Problem Statement

If you have downloaded the source code from the Premium Companion Website you will find a file named charge\_accounts.txt in the Chapter 07 folder.

This file has a list of a company's valid charge account numbers.

Each account number is a seven-digit number, such as 5658845.

Write a program that reads the contents of the file into a list.

The program should then ask the user to enter a charge account number.

The program should determine whether the number is

valid by searching for it in the list.

If the number is in the list, the program should

display a message indicating the number is valid.

If the number is not in the list,

the program should display a message indicating the number is invalid.

(You can access the Premium Companion Website at www.pearsonglobaleditions.com/gaddis.)

'''

file1=**open**("5. charge\_accounts.txt",'r')

valid\_charge\_accounts=file1.**read**()

valid\_charge\_accounts=valid\_charge\_accounts.split(‘\n’)

file1.**close**()

while(1):

    charge\_account\_input=**input**("Enter a charge account number: ")

    if charge\_account\_input.**isnumeric**() and **len**(charge\_account\_input)==7:

        break

    else:

**print**("Invalid Input")

**print**("Please enter correct input")

if **int**(charge\_account\_input) in valid\_charge\_accounts:

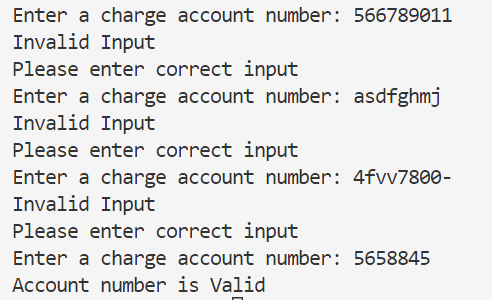
**print**("Account number is Valid")

else:

**print**("Account number is Invalid")

**Output:**

****

****

**6. Dice Rolling Function**

**Program:**

'''

In a program, write a function named roll

that accepts an integer argument number\_of\_throws.

The function should generate and return

a sorted list of number\_of\_throws random numbers

between 1 and 6.

The program should prompt the user to enter a positive integer

that is sent to the function, and then print the returned list

'''

import **random**

def **checkvalid**(n):

    for i in **range**(**len**(n)):

        if n[i]<"0" or n[i]>"9":

            return 0

    return 1

def **roll**(no\_of\_throws):

    result=[]

    for i in **range**(no\_of\_throws):

        face\_value=**random**.randint(1,6)

        result.**append**(face\_value)

    result.**sort**()

    return result

number\_of\_throws=**input**("Enter no of throws you want to roll a die (Note: Enter only positive integers): ")

while(True):

    if **checkvalid**(number\_of\_throws)==0:

**print**("You have a entered a invalid number")

**print**("Please enter a valid input")

        number\_of\_throws=**input**("Enter no of throws you want to roll a die (Note: Enter only positive integers): ")

    else:

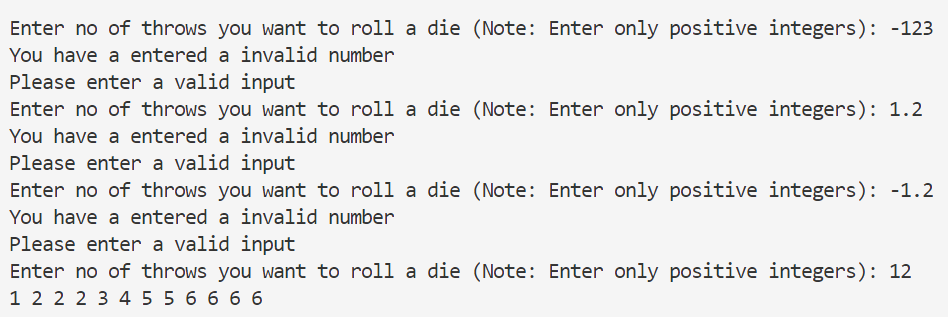
        number\_of\_throws=**int**(number\_of\_throws)

        break

rolled\_values=**roll**(number\_of\_throws)

**print**(\*rolled\_values)

**Output:**

****

**7. Driving License Exam**

**Program:**

'''

The local driver’s license office has asked you to create an application

 that grades the written portion of the driver’s license exam.

  The exam has 20 multiple-choice questions. Here

are the correct answers:

1. A

2. C

3. A

4. A

5. D

6. B

7. C

8. A

9. C

10. B

11. A

12. D

13. C

14. A

15. D

16. C

17. B

18. B

19. D

20. A

Your program should store these correct answers in a list.

 The program should read the student’s answers

 for each of the 20 questions from a text file and

 store the answers in another list.

 (Create your own text file to test the application.)

  After the student’s answers

have been read from the file, the program should display a

 message indicating whether the

student passed or failed the exam.

(A student must correctly answer 15 of the 20 questions to pass the exam.)

 It should then display the total number of correctly answered questions,

the total number of incorrectly answered questions,

and a list showing the question numbers of the incorrectly answered question

'''

file1=**open**("7. driving licence mcqs.txt",'r')

**print**(file1.**read**())

file1.**close**()

**print**("Enter your answers ")

**print**("Options lie in [A,B,C,D]")

student\_answers=[]

key='ACAADBCACBADCADCBBDA'

wrongly\_answeres\_qns=[]

no\_of\_crct\_answers=0

for i in **range**(20):

    while(1):

**print**(i+1,end=". ")

        ans=**input**()

        if ans=='A' or ans=='B' or ans=='C' or ans=='D':

            break

        else:

**print**("Enter valid option (Note:Correct Option is one of A,B.C and D)")

    student\_answers.**append**(ans)

    if ans==key[i]:

        no\_of\_crct\_answers+=1

    else:

        wrongly\_answeres\_qns.**append**(i+1)

**print**("You have Scored",no\_of\_crct\_answers,"marks")

if no\_of\_crct\_answers>=15:

**print**("You have passed this exam")

else:

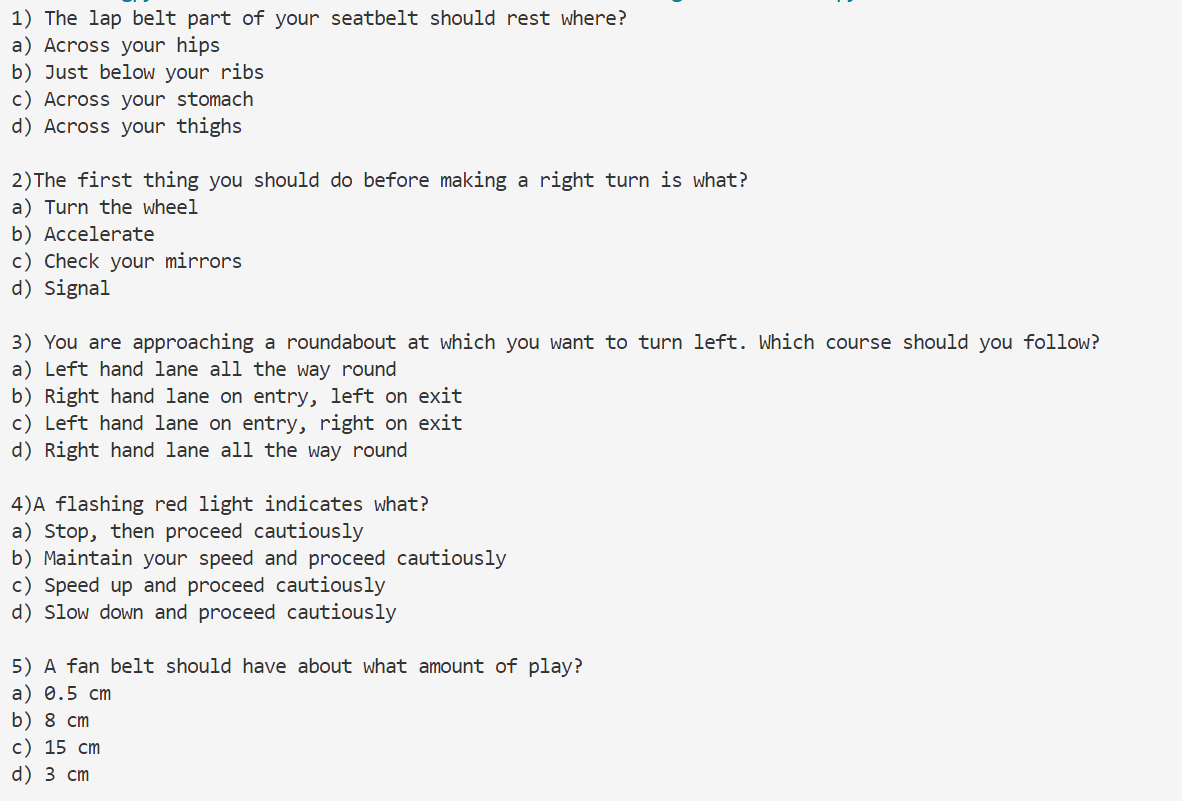
**print**("You have failed in this exam")

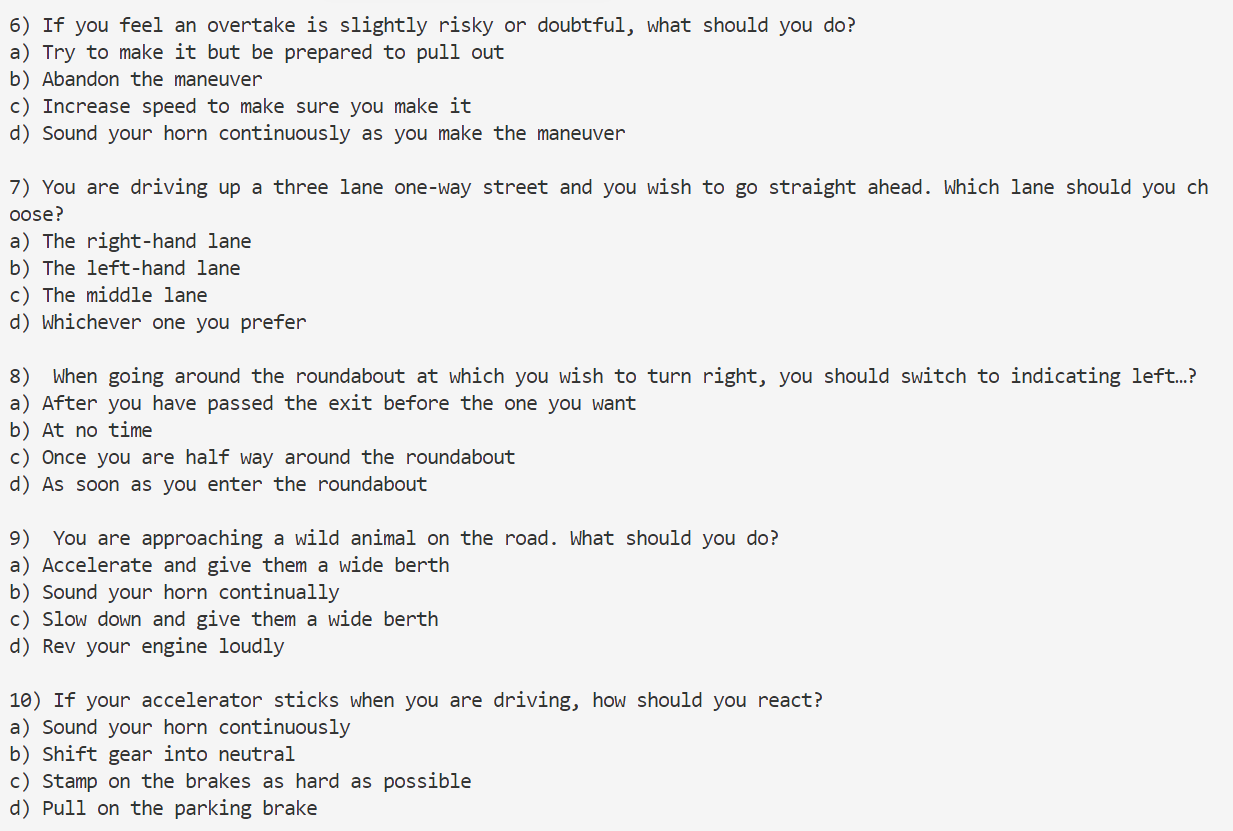
if no\_of\_crct\_answers!=20:

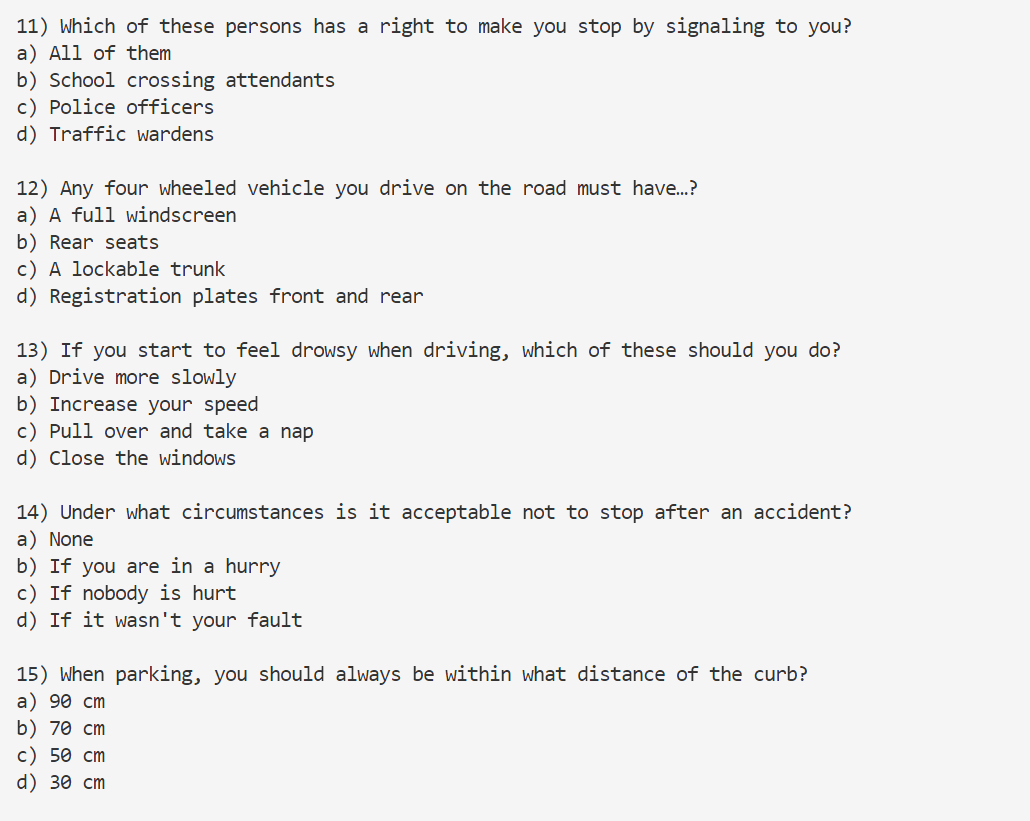
**print**("You answers are incorrect for following questions: ")

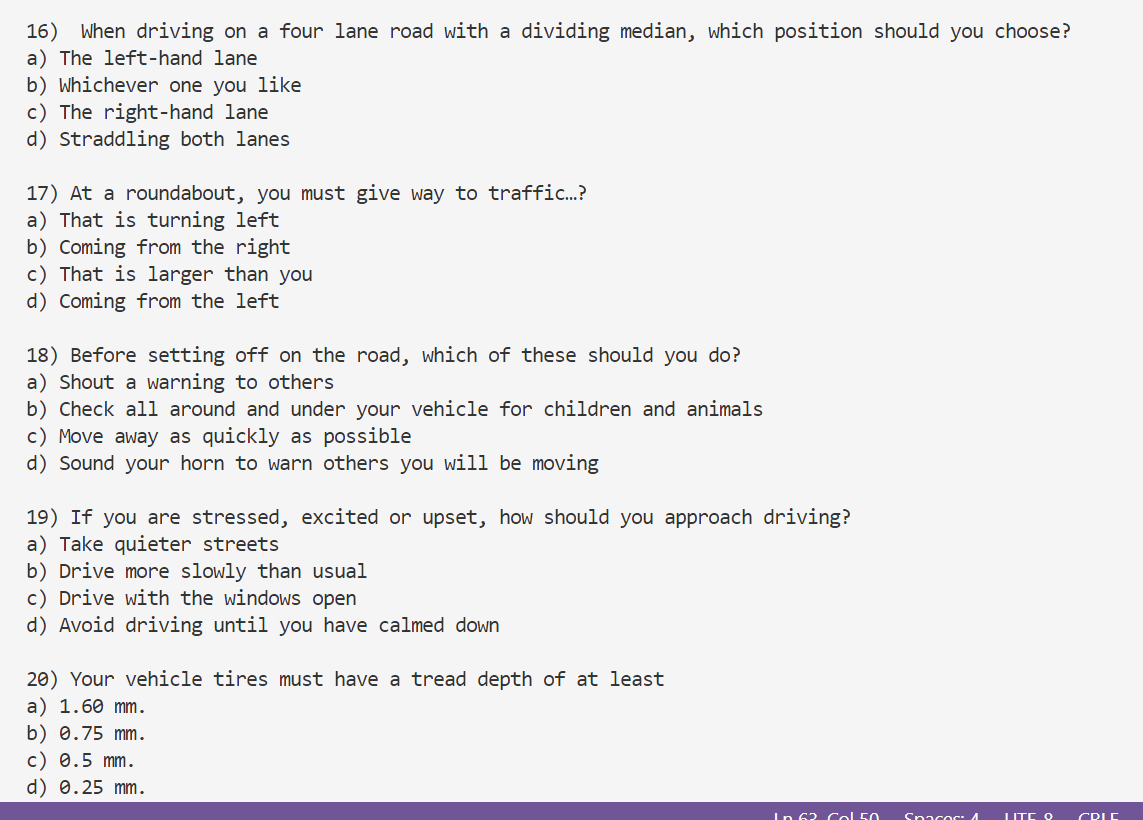
**print**(\*wrongly\_answeres\_qns)

**Output:**

****







**Text

Description automatically generated**

**Text

Description automatically generated**

**8. Name Search**

**Program:**

'''

If you have downloaded the source code

you will find the following files in the Chapter 07 folder:

•    GirlNames.txt This file contains a list

of the 200 most popular names given to girls

born in the United States from the year 2000 through 2009.

•    BoyNames.txt This file contains a list of the 200 most popular names given to boys

born in the United States from the year 2000 through 2009

Write a program that reads the contents of the two files into two separate lists.

The user should be able to enter a boy’s name, a girl’s name, or both, and the application will display

messages indicating whether the names were among the most popular.

(You can access the Premium Companion Website at www.pearsonglobaleditions.com/gaddis.

'''

girl\_names\_file=**open**("8. GirlNames.txt",'r')

girls\_list=girl\_names\_file.**read**()

girls\_list=girls\_list.**split**("\n")

girl\_names\_file.**close**()

boy\_names\_file=**open**("8. BoyNames.txt",'r')

boys\_list=boy\_names\_file.**read**()

boys\_list=boys\_list.**split**("\n")

boy\_names\_file.**close**()

**print**("Enter list of names: ")

names=**list**(**input**().**split**())

for i in **range**(**len**(names)):

    names[i]=names[i].**title**() *#capitaizes each word*

    if names[i] in girls\_list:

**print**(names[i],"is one of the most popular girl's name")

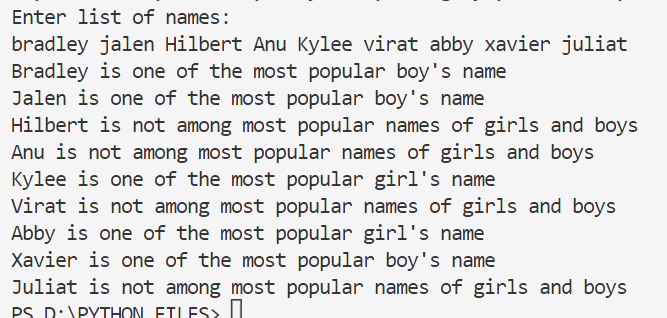
    elif names[i] in boys\_list:

**print**(names[i],"is one of the most popular boy's name")

    else:

**print**(names[i],"is not among most popular names of girls and boys")

**Output:**

****

**9. Population Data**

**Program:**

'''

 Population Data

If you have downloaded the source code

you will find a file named USPopulation.txt in the Chapter 07 folder.

 The file contains the midyear population of the United States,

 in thousands, during the years 1950 through 1990.

 The first line in the file contains the population for 1950,

 the second line contains the population for 1951, and so forth.

Write a program that reads the file’s contents into a list.

 The program should display the following data:

• The average annual change in population during the time period

• The year with the greatest increase in population during the time period

• The year with the smallest increase in population during the time period

'''

file1=**open**("9. USPopulation.txt",'r')

population\_list=file1.**read**()

population\_list=**list**(**map**(**int**,population\_list.**split**('\n')))

file1.**close**()

start\_year=1950

annual\_changes\_list=[]

for i in **range**(0,**len**(population\_list)-1):

    annual\_change=population\_list[i+1]-population\_list[i]

    annual\_changes\_list.**append**(annual\_change)

average\_change=**sum**(annual\_changes\_list)/**len**(annual\_changes\_list)

greatest\_increase=**max**(annual\_changes\_list)

gi\_index=annual\_changes\_list.**index**(greatest\_increase)

smallest\_increase=**min**(annual\_changes\_list)

si\_index=annual\_changes\_list.**index**(smallest\_increase)

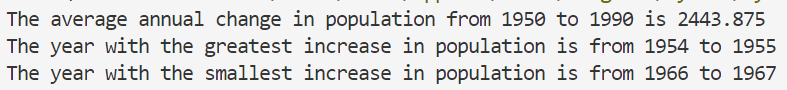
**print**("The average annual change in population from 1950 to 1990 is",average\_change)

**print**("The year with the greatest increase in population is from",gi\_index+1950,"to",gi\_index+1951)

**print**("The year with the smallest increase in population is from",\

    si\_index+1950,"to",si\_index+1951)

**Output:**

****

**10. World Series Champions**

**Program:**

'''

If you have downloaded the source code

you will find a file named WorldSeriesWinners.txt in the Chapter 07 folder.

This file contains a chronological list of the

World Series winning teams from 1903 through 2009.

(The first line in the file is the name of the team that

won in 1903, and the last line is the name of the team that won in 2009.

 Note the World Series was not played in 1904 or 1994.)

Write a program that lets the user enter the name of a team, then displays the number of

times that team has won the World Series in the time period from 1903 through 2009.

TIP: Read the contents of the WorldSeriesWinners.txt file into a list. When the

user enters the name of a team, the program should step through the list, counting the

number of times the selected team appears

'''

file1=**open**("10. WorldSeriesWinners.txt",'r')

winners\_list=file1.**read**()

winners\_list=**list**(winners\_list.**split**('\n'))

file1.**close**()

if no\_of\_times>0:

    years=[]

    j=0

    for i in **range**(1903,2009):

        if i==1904 or i==1994:

            continue

        else:

            if name\_of\_team==winners\_list[j]:

                years.**append**(i)

            j=j+1

**print**(name\_of\_team,"won",no\_of\_times,"times","in years",end=" ")

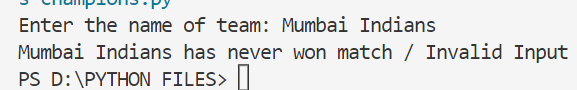
**print**(\*years)

else:

**print**(name\_of\_team,"has never won match / Invalid Input")

**Output:**

****

****

**11. Lo Shu Magic Square**

**Program:**

'''

The Lo Shu Magic Square is a grid with 3 rows and 3 columns, shown in Figure 7-18.

 The Lo Shu Magic Square has the following properties:

•    The grid contains the numbers 1 through 9 exactly.

•    The sum of each row, each column, and each diagonal all add up to the same number.

This is shown in Figure 7-19.

In a program you can simulate a magic square using a two-dimensional list.

Write a function that accepts a two-dimensional list as an argument and

determines whether the list is a Lo Shu Magic Square.

 Test the function in a program

'''

def **magicSquare**(grid):

    '''

    a00     a01     a02

    a10     a11     a12

    a20     a21     a22

    '''

    r0=**sum**(grid[0])

    r1=**sum**(grid[1])

    r2=**sum**(grid[2])

    dij=grid[0][0]+grid[1][1]+grid[2][2]

    din\_j=grid[2][0]+grid[1][1]+grid[0][2]

    c0=grid[0][0]+grid[1][0]+grid[2][0]

    c1=grid[0][1]+grid[1][1]+grid[2][1]

    c2=grid[0][2]+grid[1][2]+grid[2][2]

    if r0==r1==r2==dij==din\_j==c0==c1==c2:

        return 1

    else:

        return 0

**print**("Enter elements in 3-D Grid:  ")

**print**("Note: Enter numbers only between 1 and 9")

grid=[]

for i in **range**(3):

    row=**list**(**map**(**int**,**input**().**split**()))

    grid.**append**(row)

flag=**magicSquare**(grid)

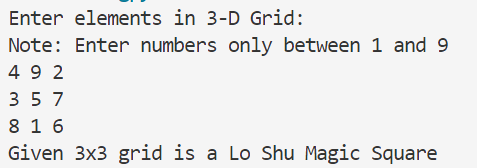
if flag==1:

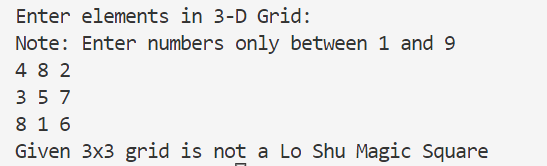
**print**("Given 3x3 grid is a Lo Shu Magic Square")

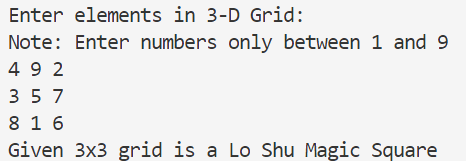
else:

**print**("Given 3x3 grid is not a Lo Shu Magic Square")

**Output:**

****

****

****