Charles Sturt University

**Assessment item 2**

**Data Analysis**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Student ID: 11706271

Student Name: Alina Maksumova

Lecturer Name: Dr Thair Al-Dala'in

Subject Name: Programming Principles

Subject Code: ITC558

Sydney 2020

Contents

[**Task 1** 3](#_Toc41250682)

[Flowchart: Step 2-6 3](#_Toc41250683)

[**Task 2** 4](#_Toc41250684)

[Testing 4](#_Toc41250685)

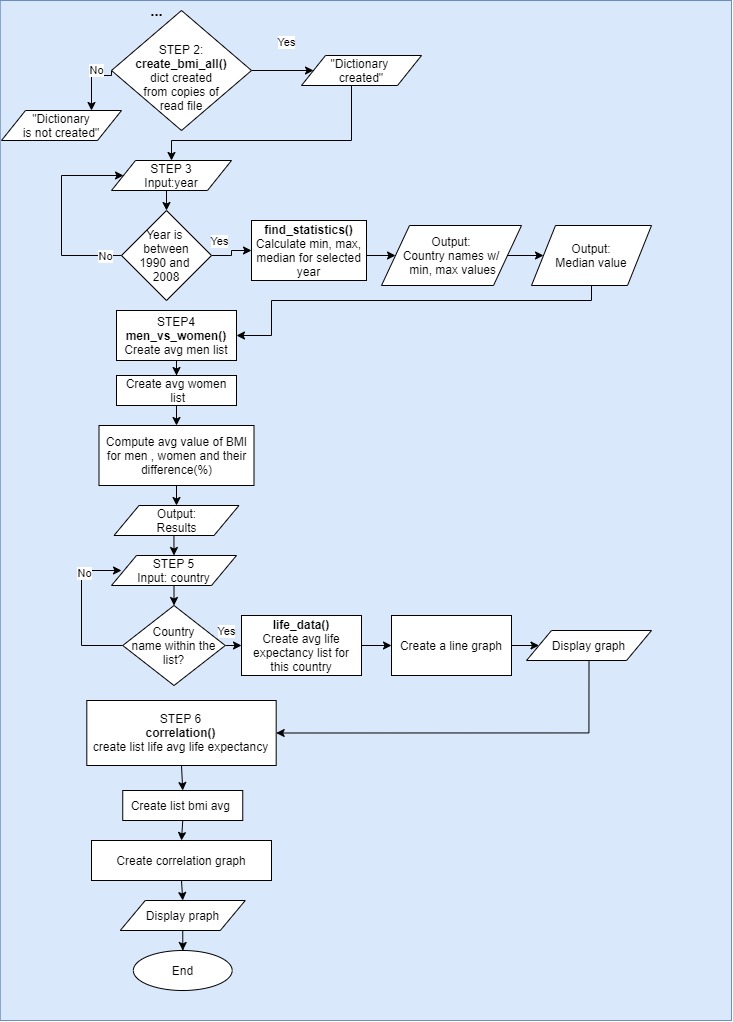
[**Task 3** 24](#_Toc41250686)

[Source Code 24](#_Toc41250687)

[Text Code 24](#_Toc41250688)

# **Task 1**

## Flowchart: Step 2-6

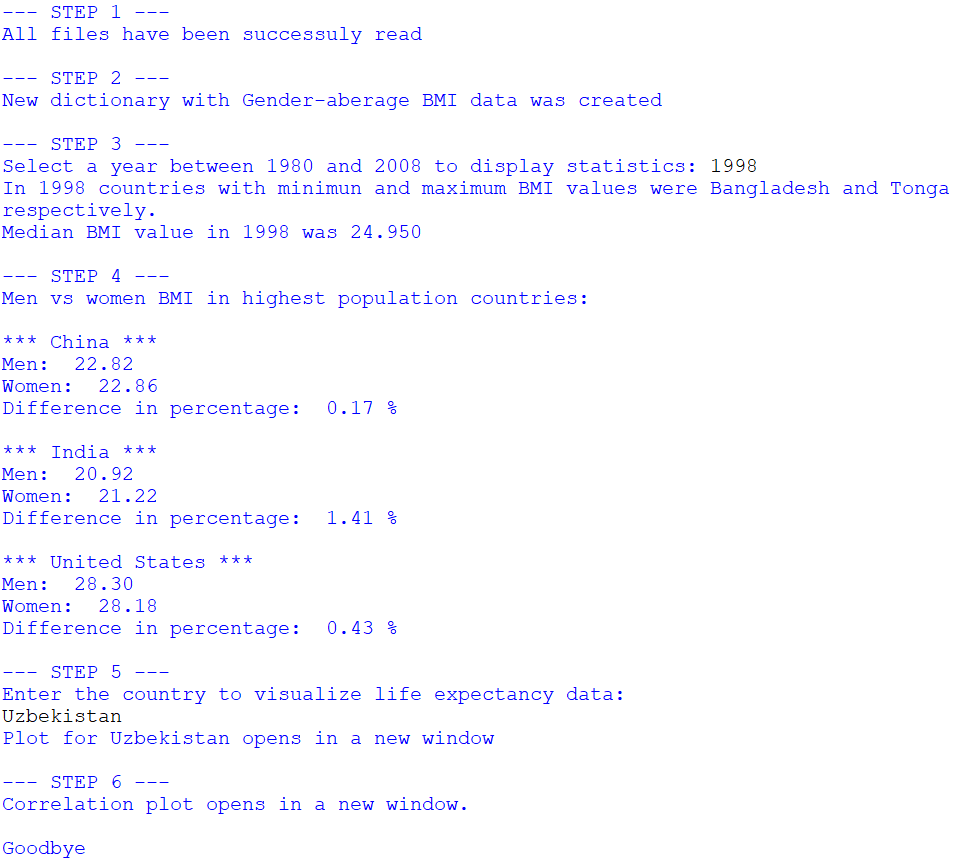


# **Task 2**

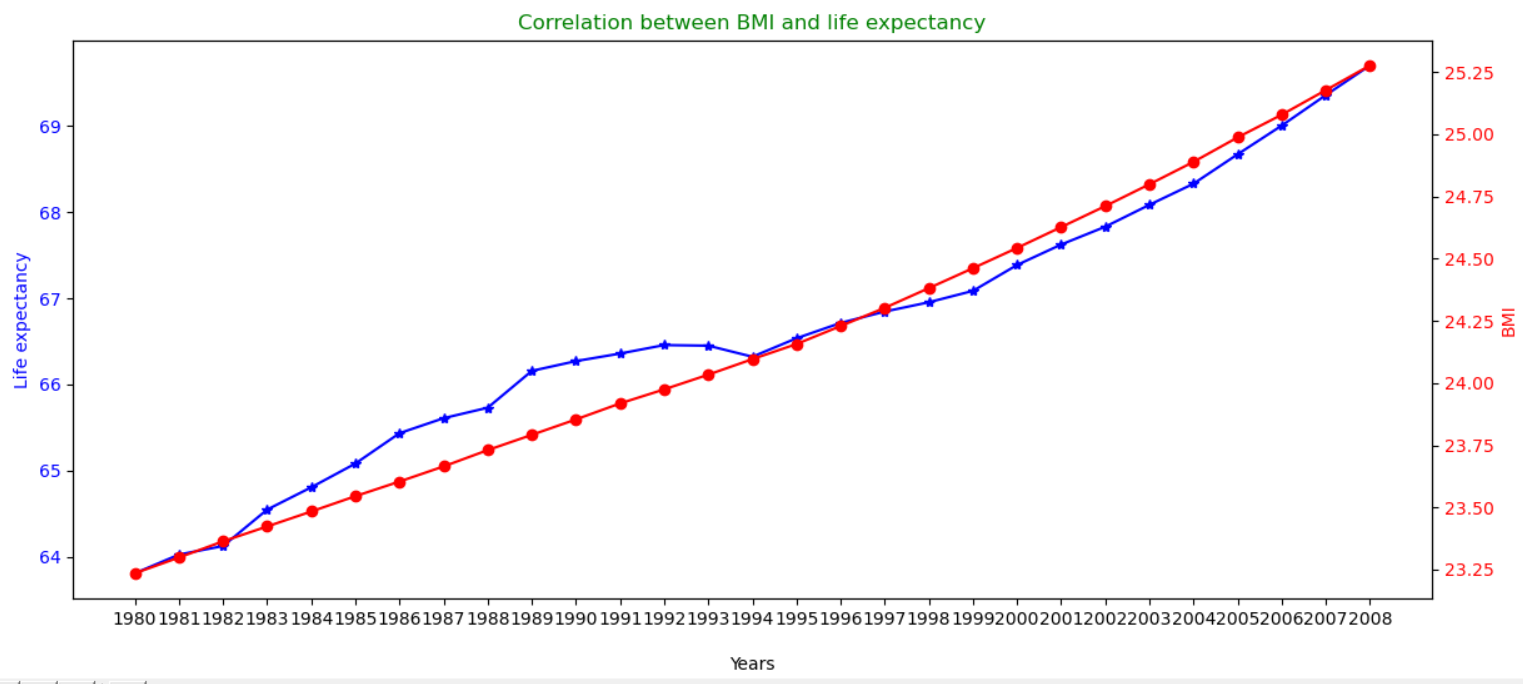
## Testing

| **Test Data Table** | | | |
| --- | --- | --- | --- |
| **Test data type** | **Test data** | **The reason it was selected** | **The output expected due to the use of the test data** |
| 1. Normal | Year: 1998  Country:  Uzbekistan | To test program operation for selected year and country | Successful completion from step 1 to step 6 and 2 graphs |

**Output:**

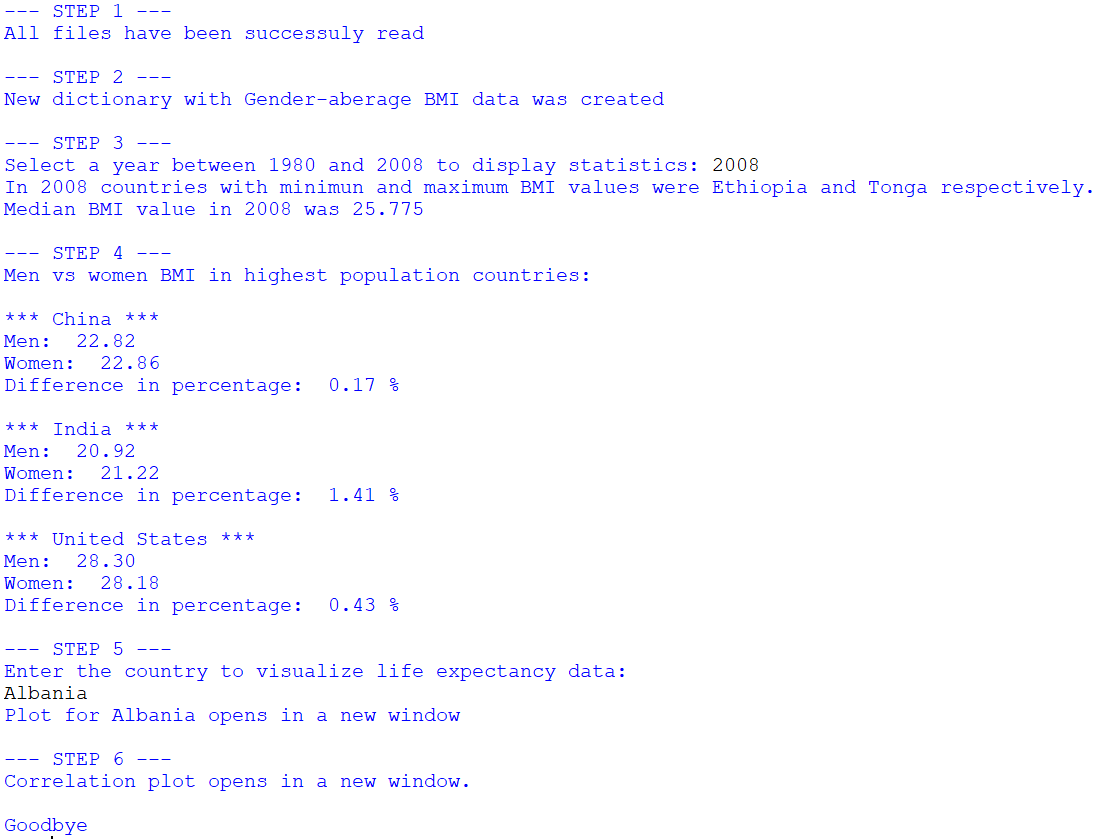


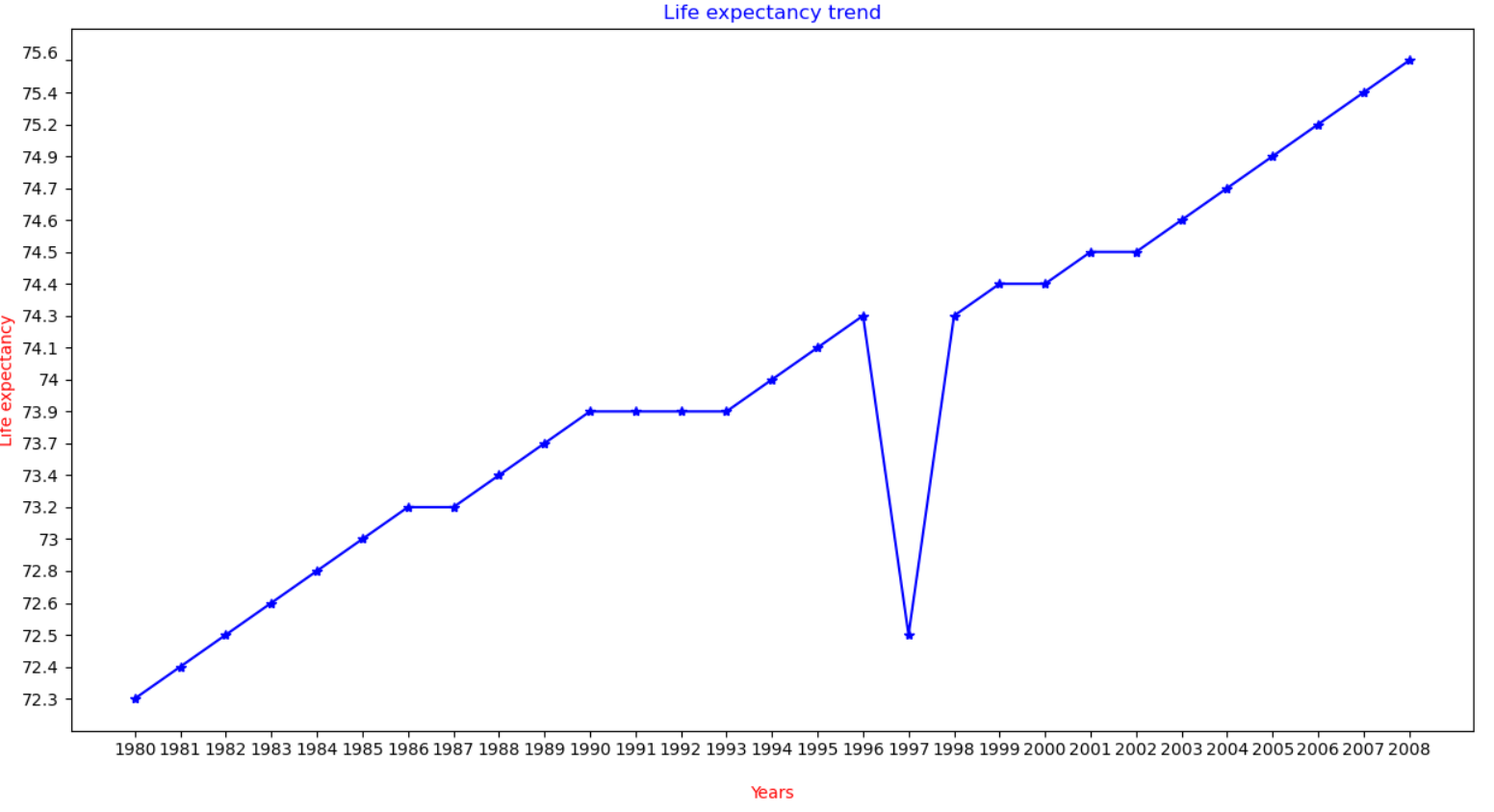


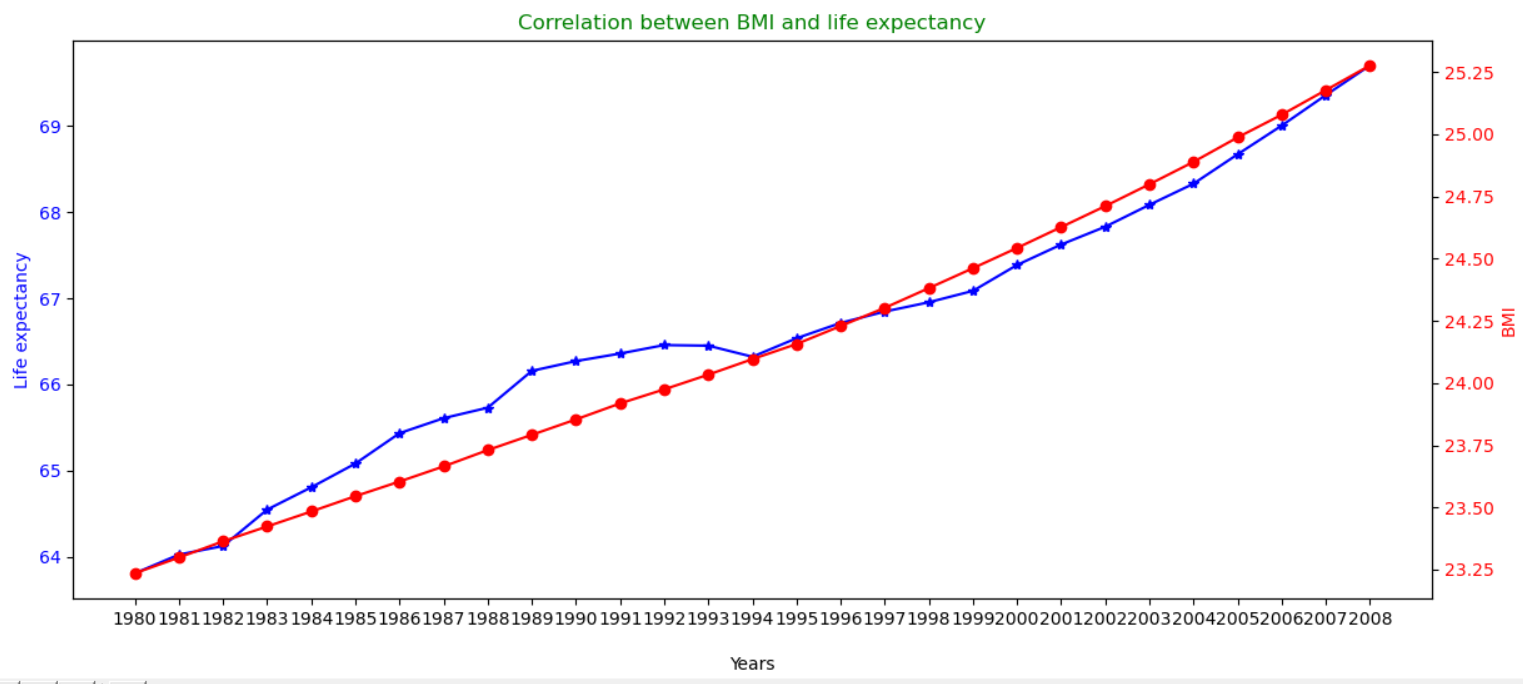


| **Test Data Table** | | | |
| --- | --- | --- | --- |
| **Test data type** | **Test data** | **The reason it was selected** | **The output expected due to the use of the test data** |
| 1. Normal | Year: 2008  Country:  Albania | To test program operation for last year and new country | Successful completion from step 1 to step 6 and 2 graphs |

**Output:**

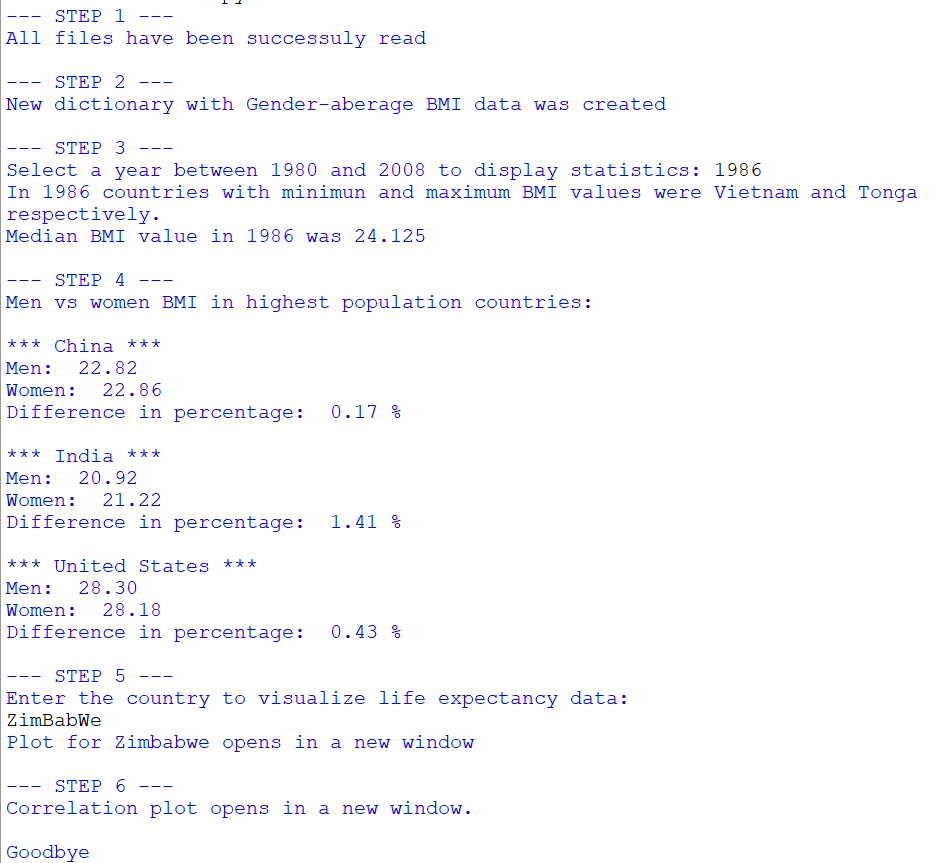


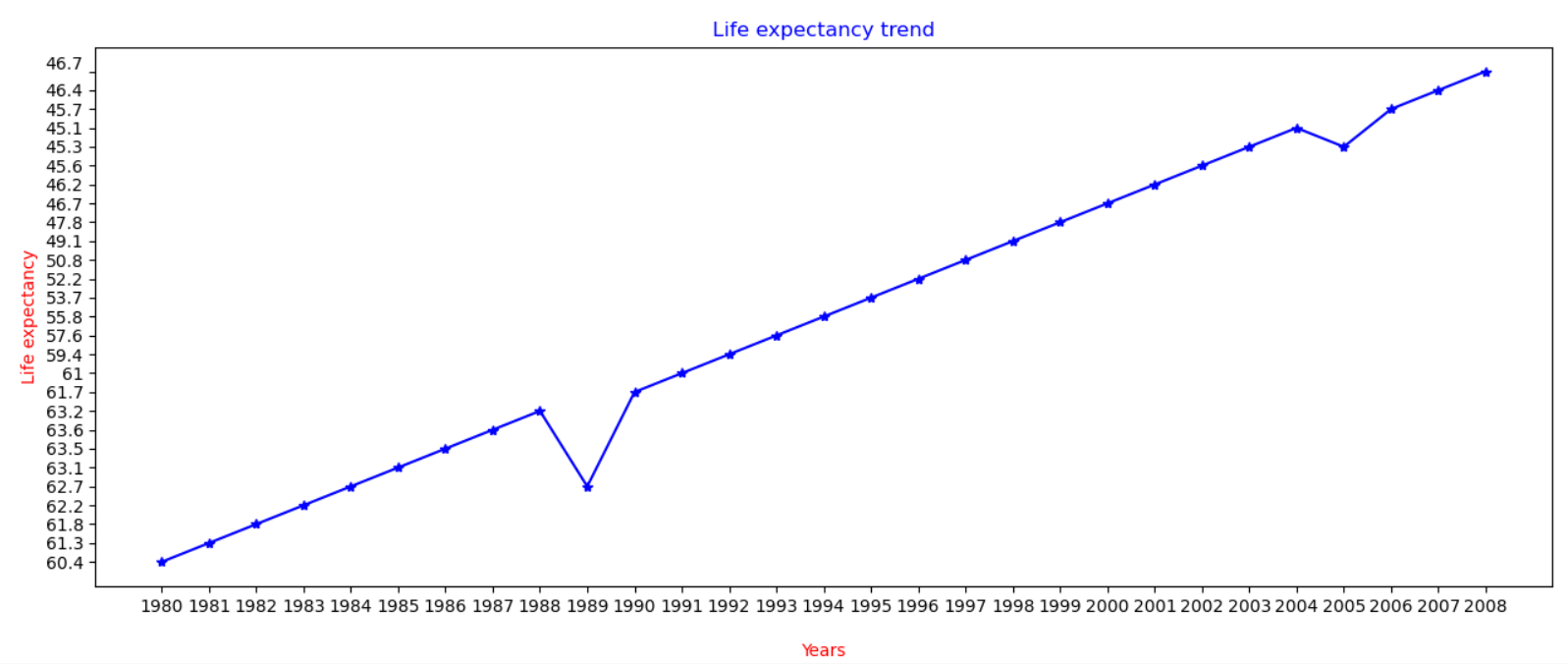


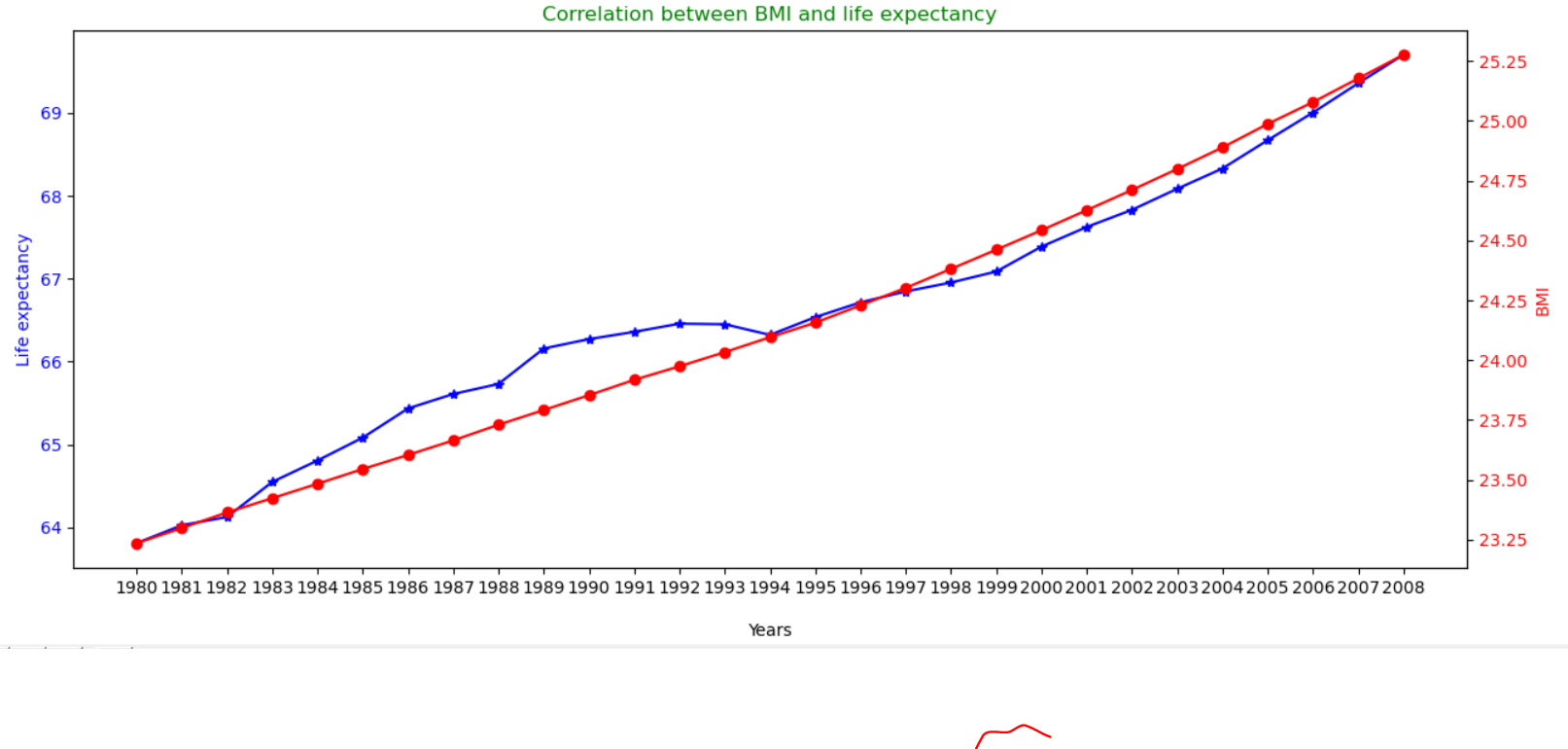


| **Test Data Table** | | | |
| --- | --- | --- | --- |
| **Test data type** | **Test data** | **The reason it was selected** | **The output expected due to the use of the test data** |
| 1. Normal | Year: 1986  Country:  ZimBabWe | To test program operation for selected year and country typed with lower and upper cases | Successful completion from step 1 to step 6 and 2 graphs |

**Output:**

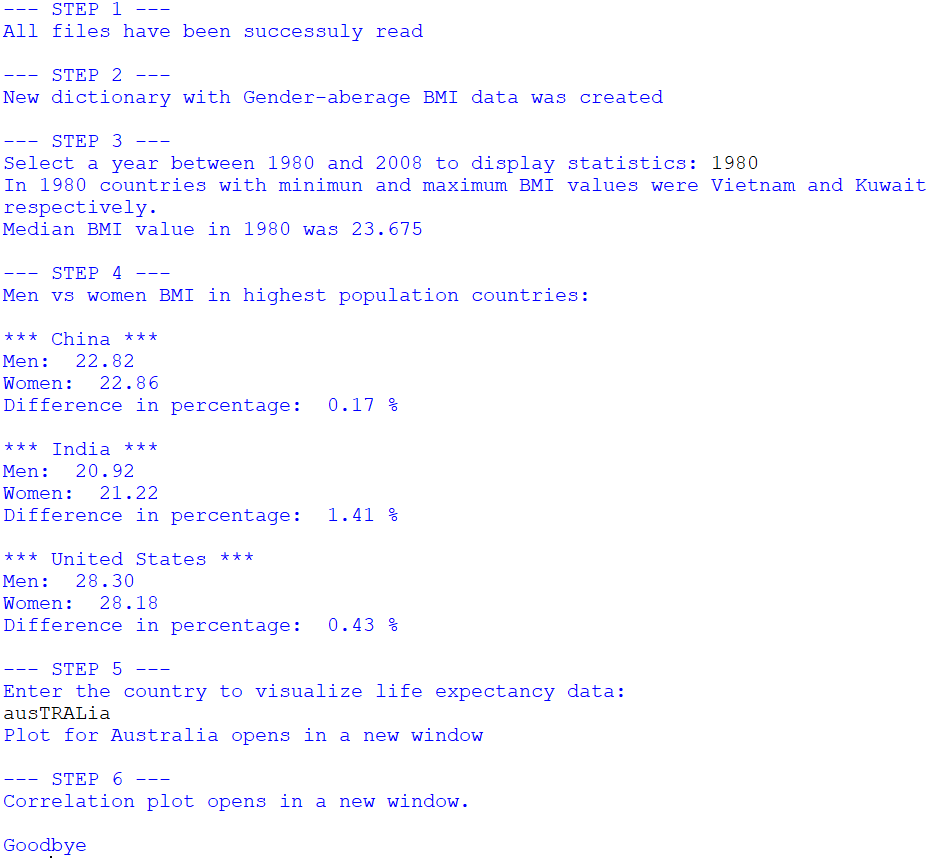


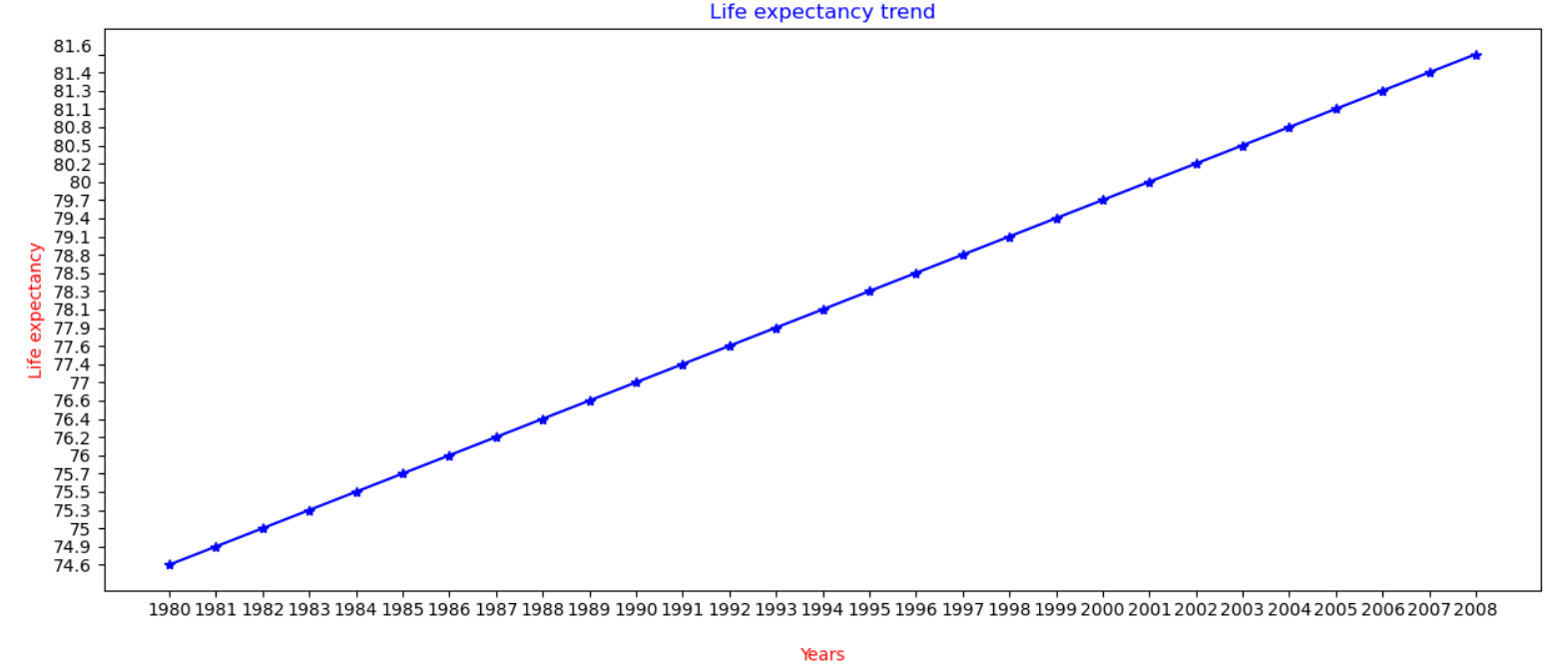


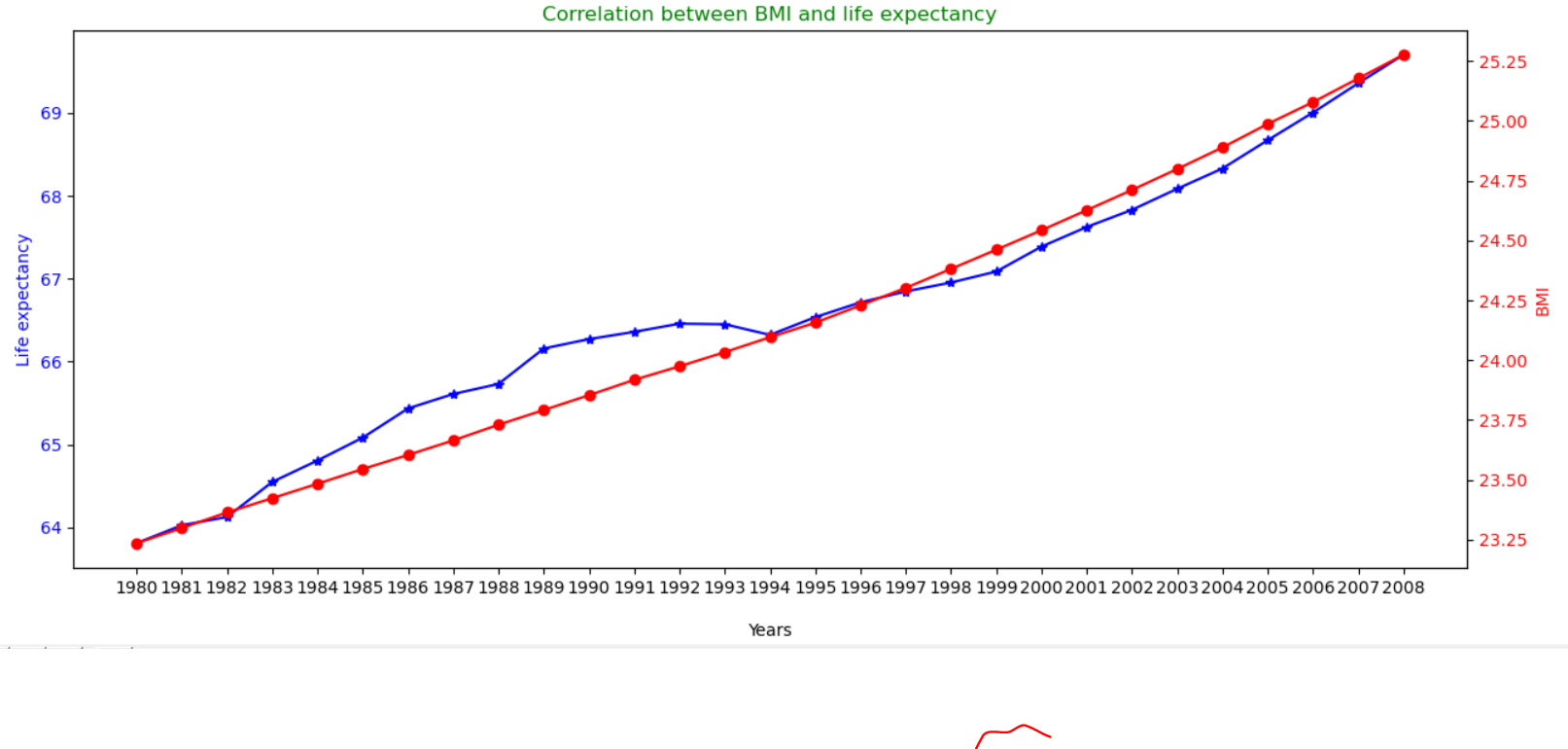


| **Test Data Table** | | | |
| --- | --- | --- | --- |
| **Test data type** | **Test data** | **The reason it was selected** | **The output expected due to the use of the test data** |
| 1. Normal | Year: 1980  Country:  ausTRALia | To test program operation for the first year and country typed with lower and upper cases | Successful completion from step 1 to step 6 and 2 graphs |

**Output:**



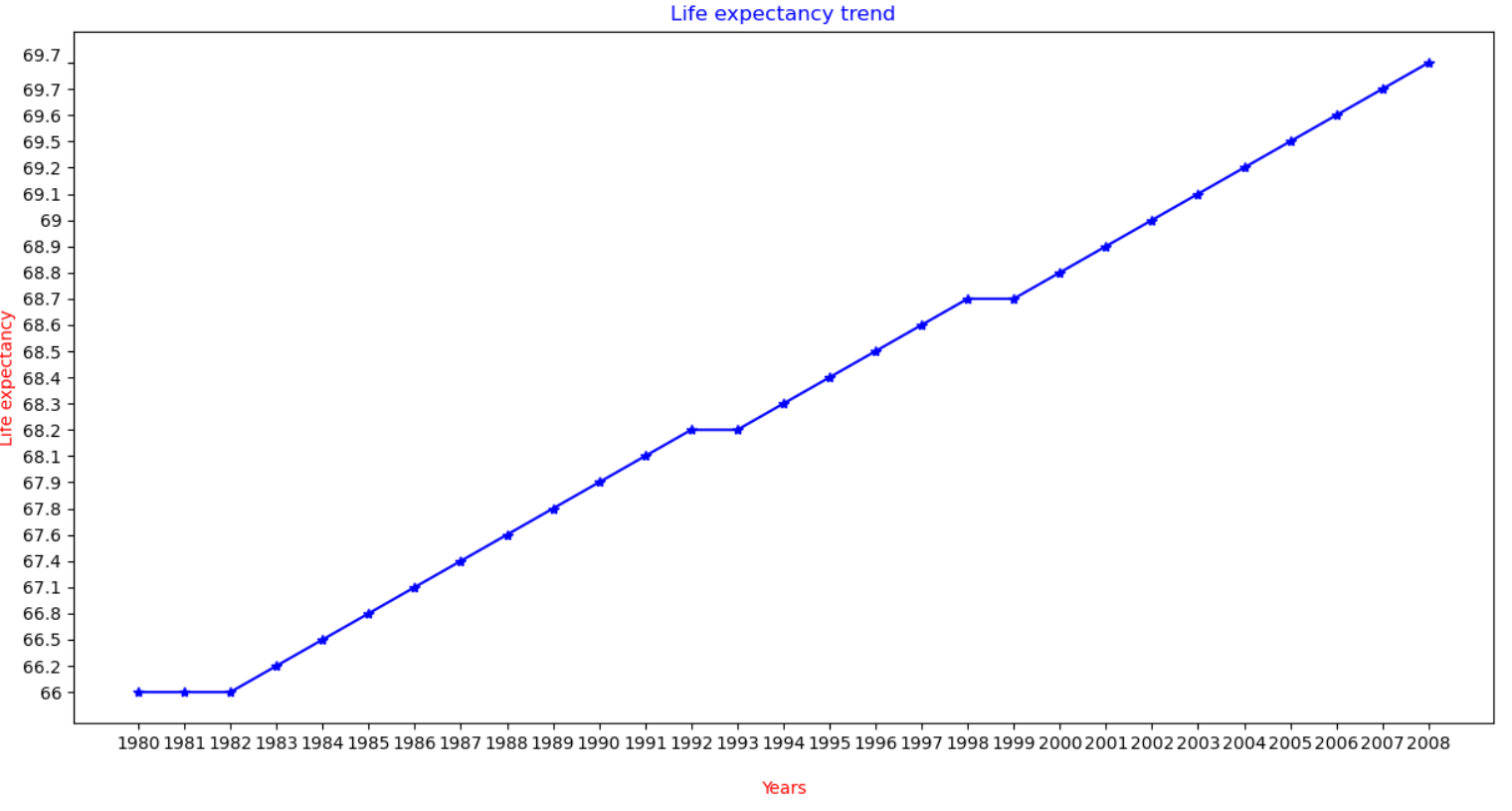


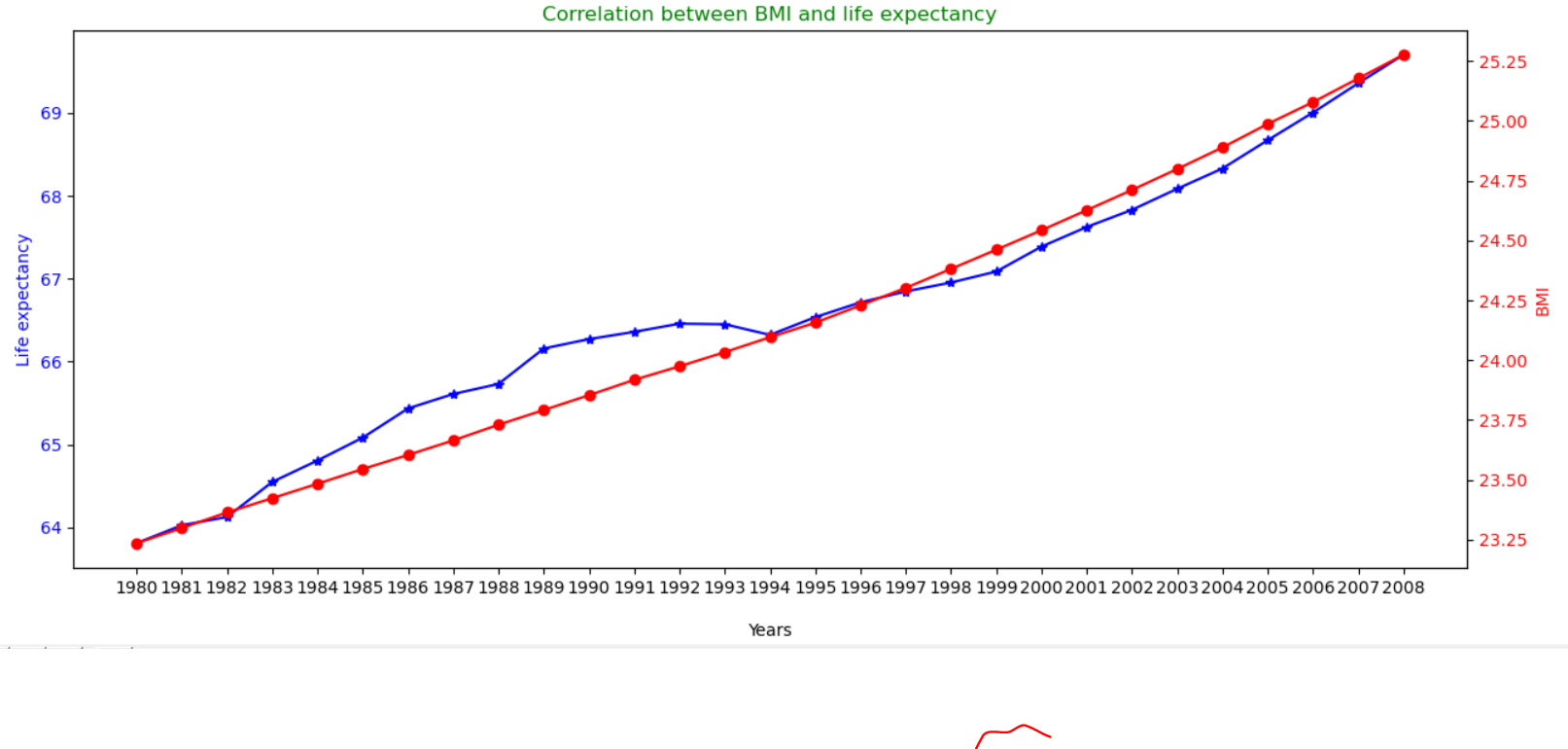


| **Test Data Table** | | | |
| --- | --- | --- | --- |
| **Test data type** | **Test data** | **The reason it was selected** | **The output expected due to the use of the test data** |
| 5) Normal | Year: 2003  Country:  tonga | To test program operation for selected year and country typed with lower cases only | Successful completion from step 1 to step 6 and 2 graphs |

**Output:**

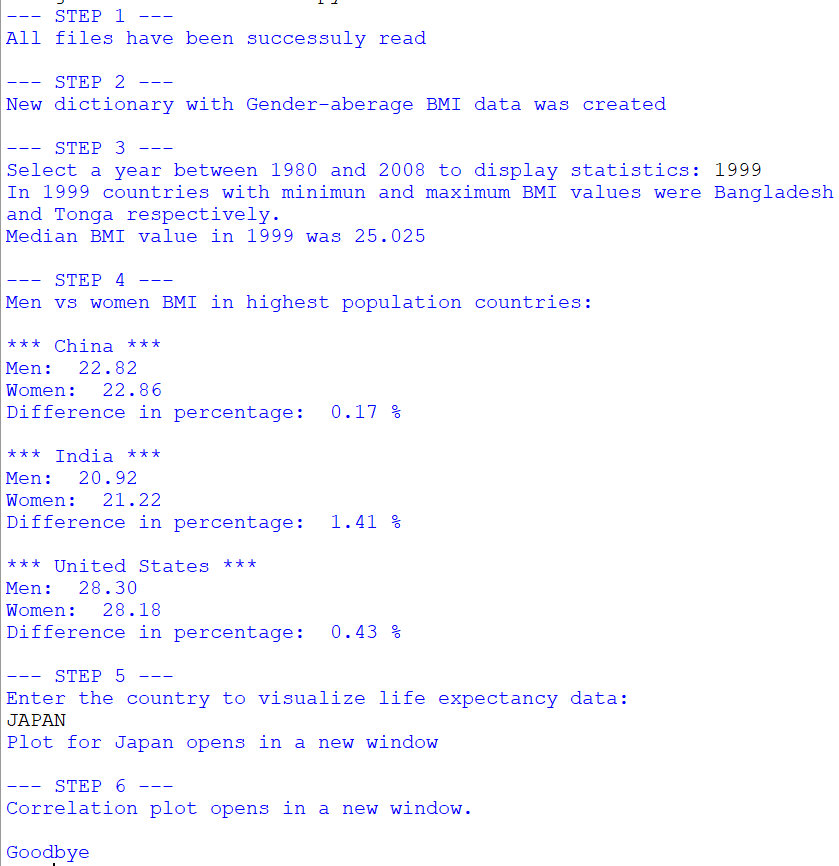


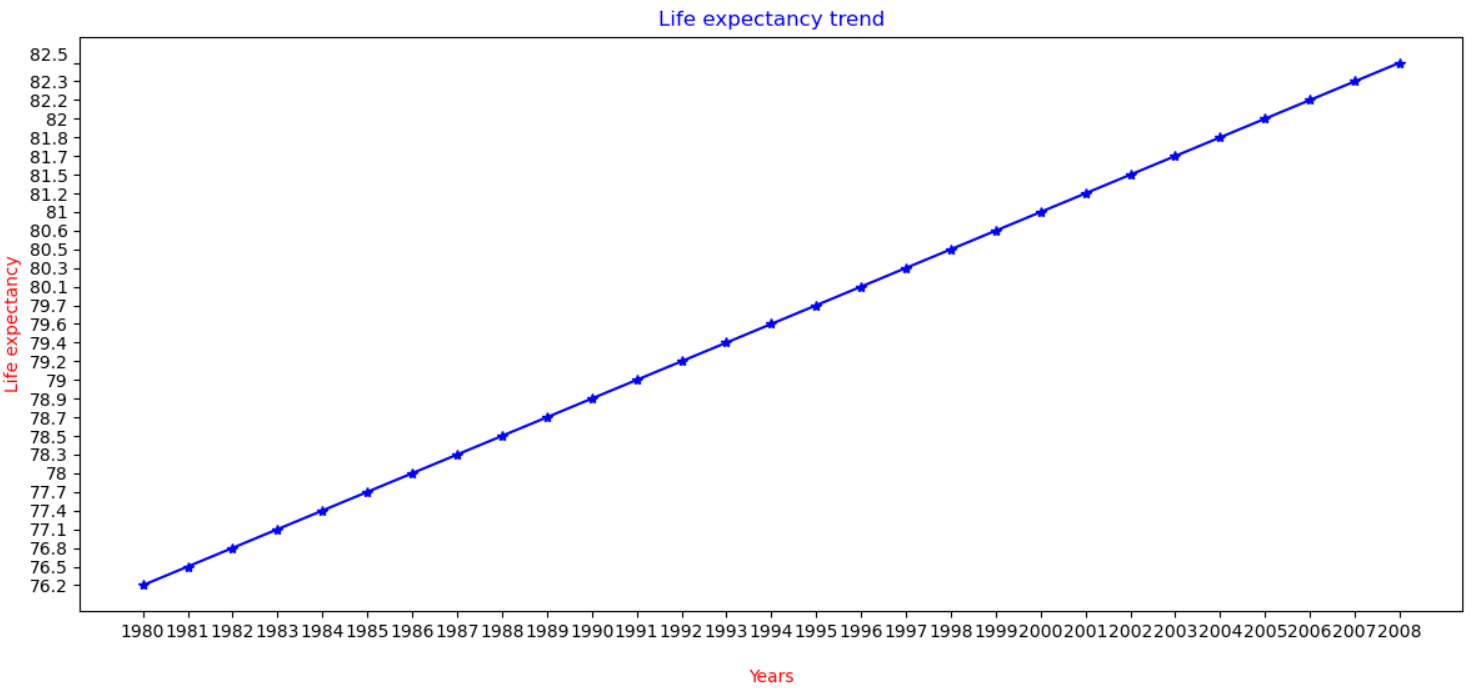


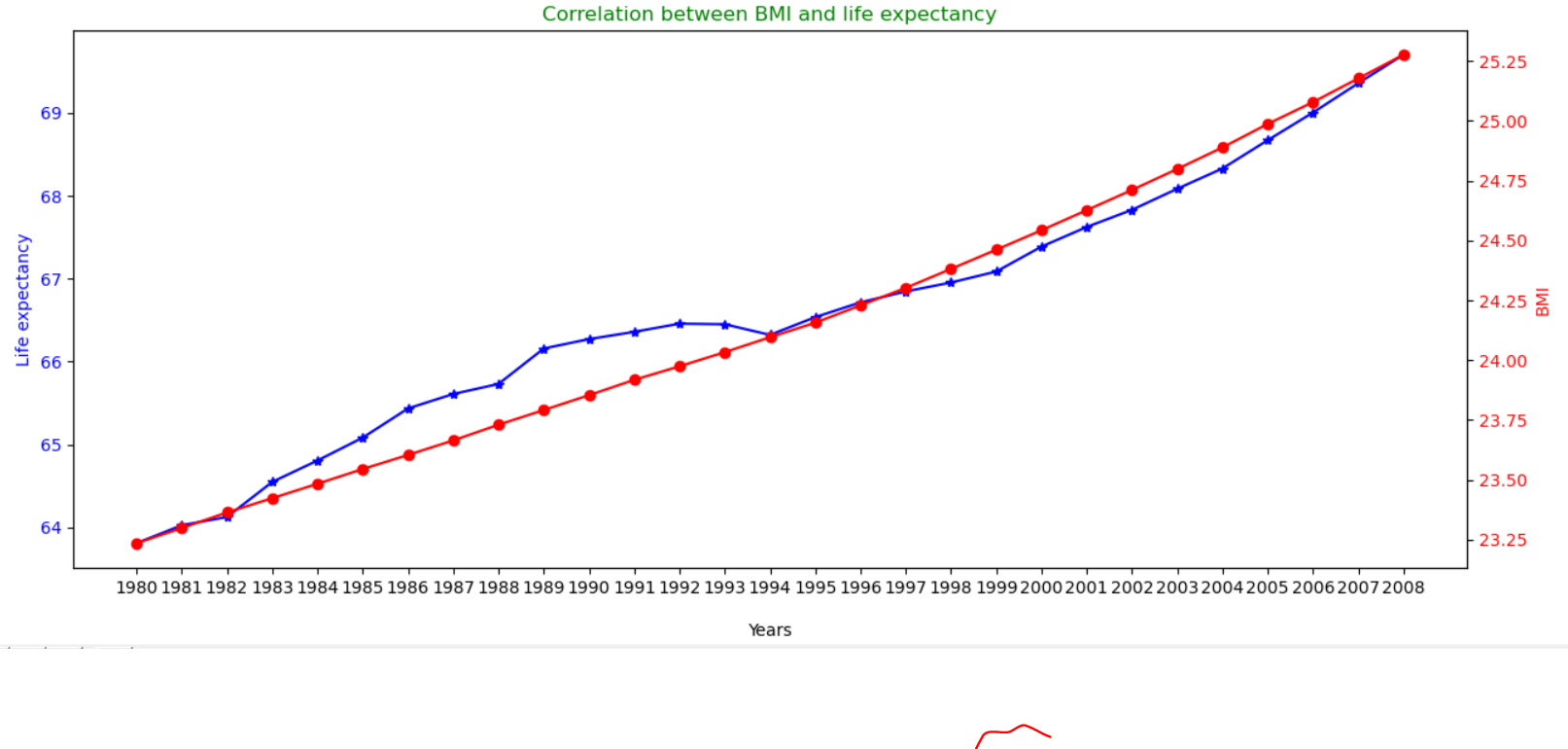


| **Test Data Table** | | | |
| --- | --- | --- | --- |
| **Test data type** | **Test data** | **The reason it was selected** | **The output expected due to the use of the test data** |
| 6) Normal | Year: 1999  Country:  JAPAN | To test program operation for selected year and country typed with upper cases only | Successful completion from step 1 to step 6 and 2 graphs |

**Output:**

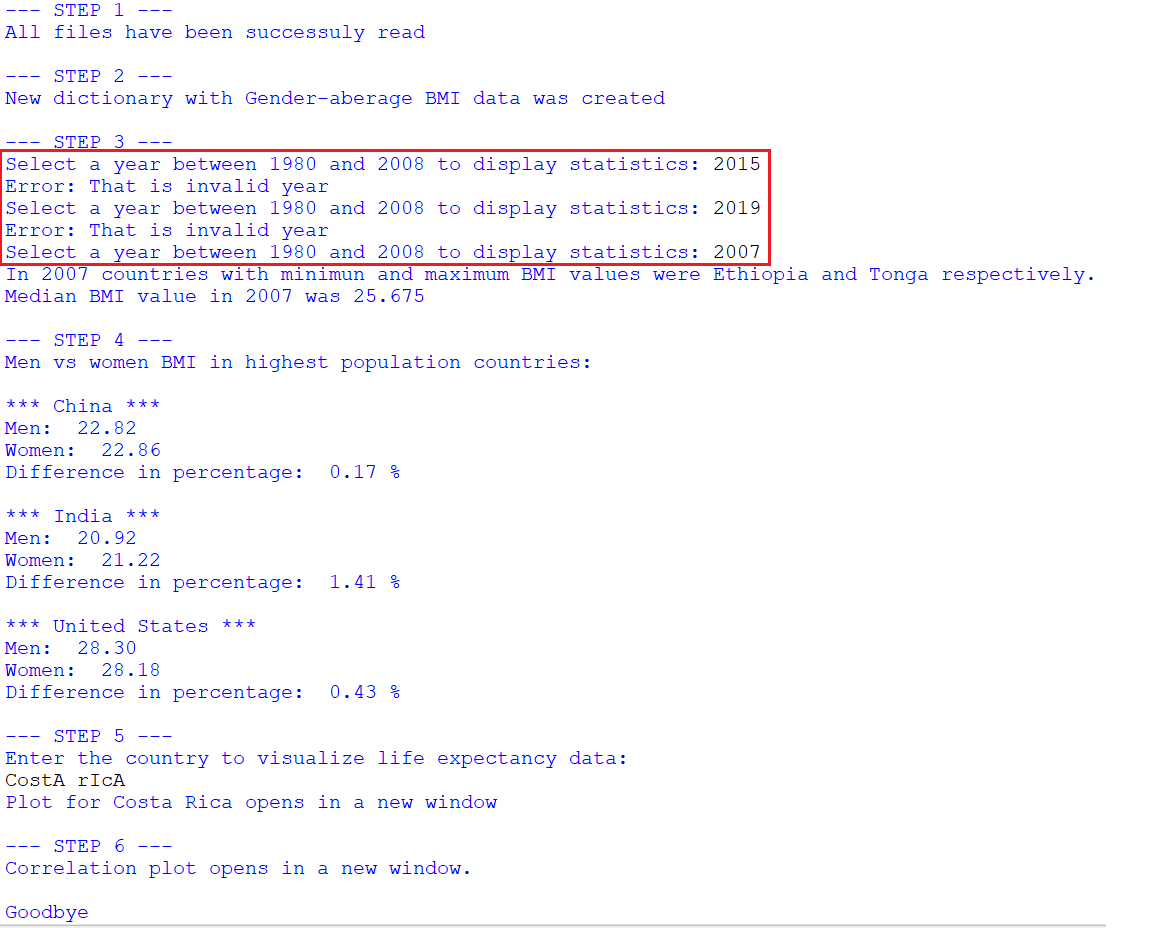


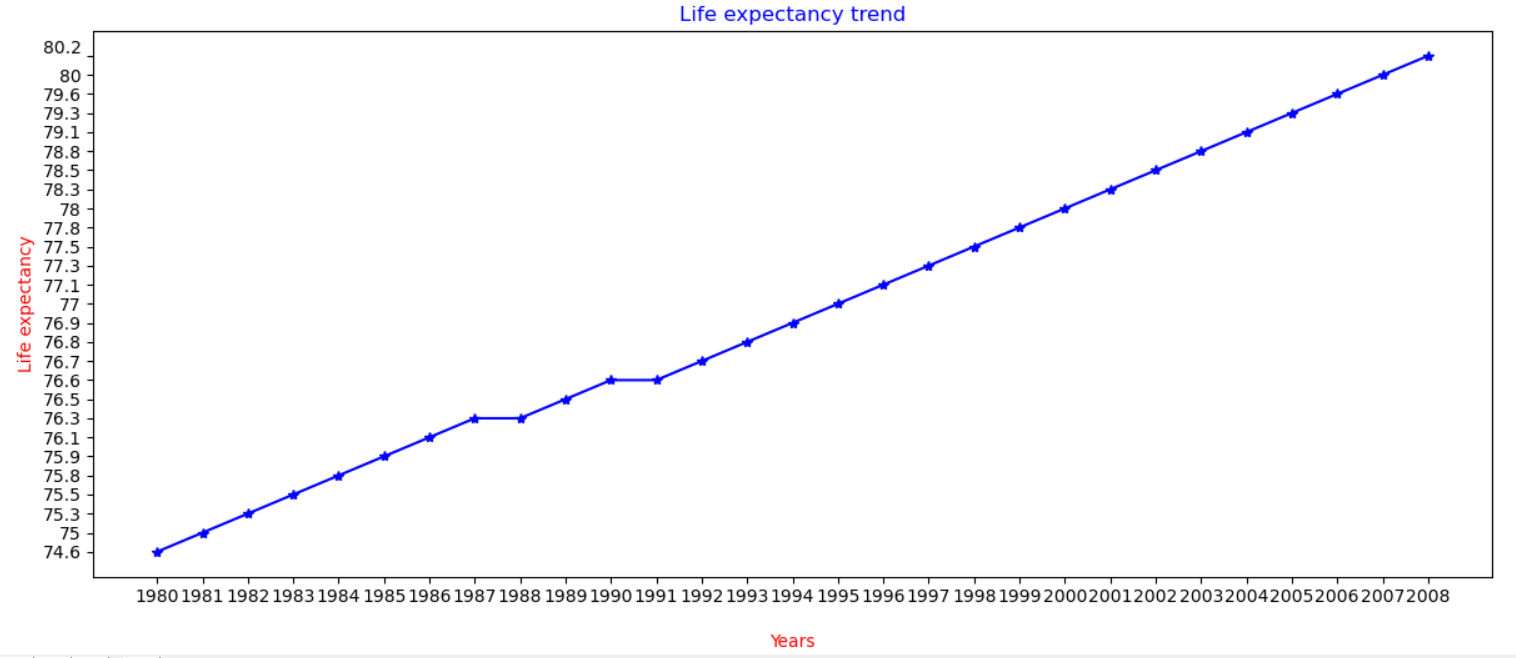


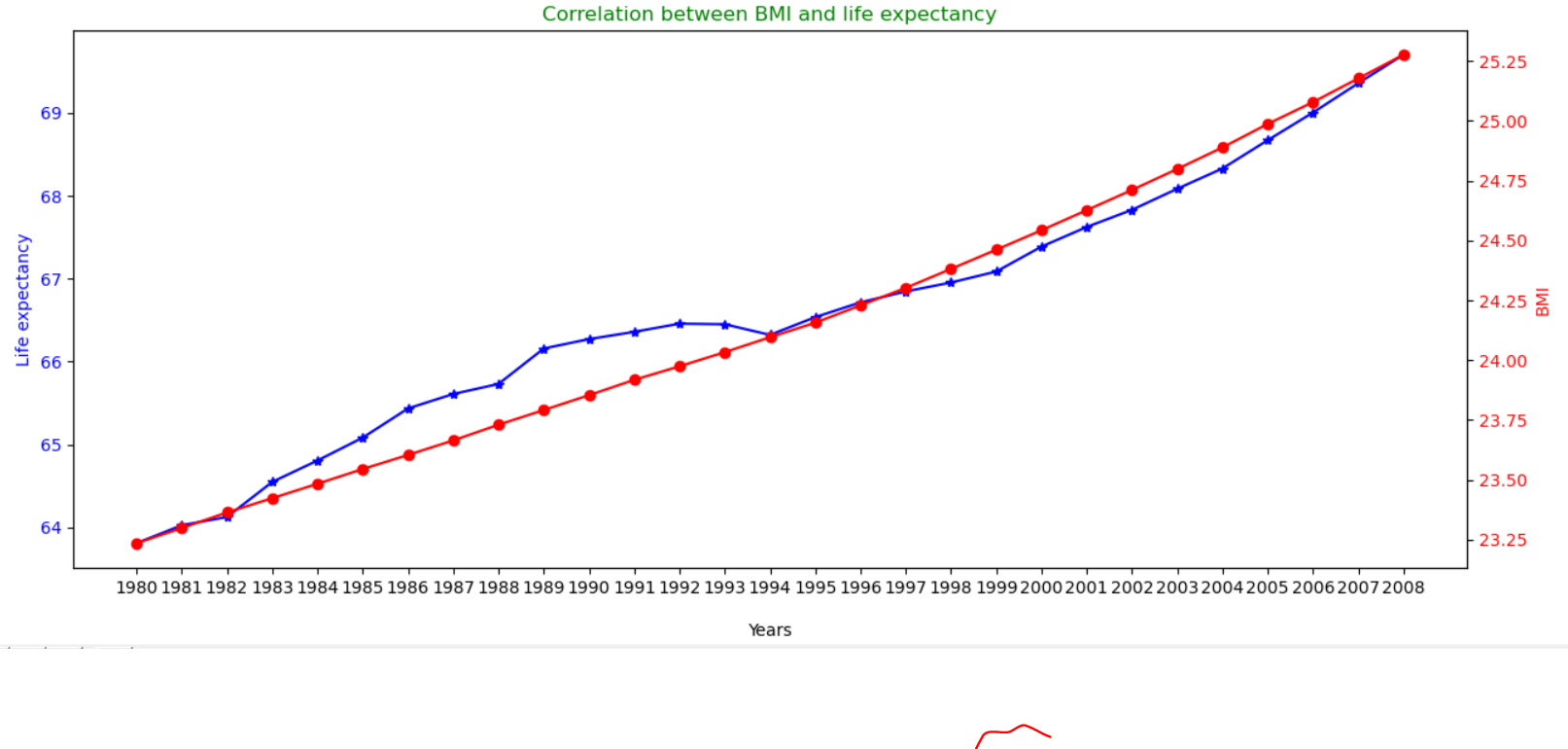


| **Test Data Table** | | | |
| --- | --- | --- | --- |
| **Test data type** | **Test data** | **The reason it was selected** | **The output expected due to the use of the test data** |
| 1. Abnormal | Year1: 2015  Year2: 2019  Year3: 2007  Country:  CostA rIcA | To test program operation for year, which is greater than expected for range; and country typed with upper and lower cases | Exception will be raised in Step 3, there should appear error message. After that, new input should be required |

**Output:**

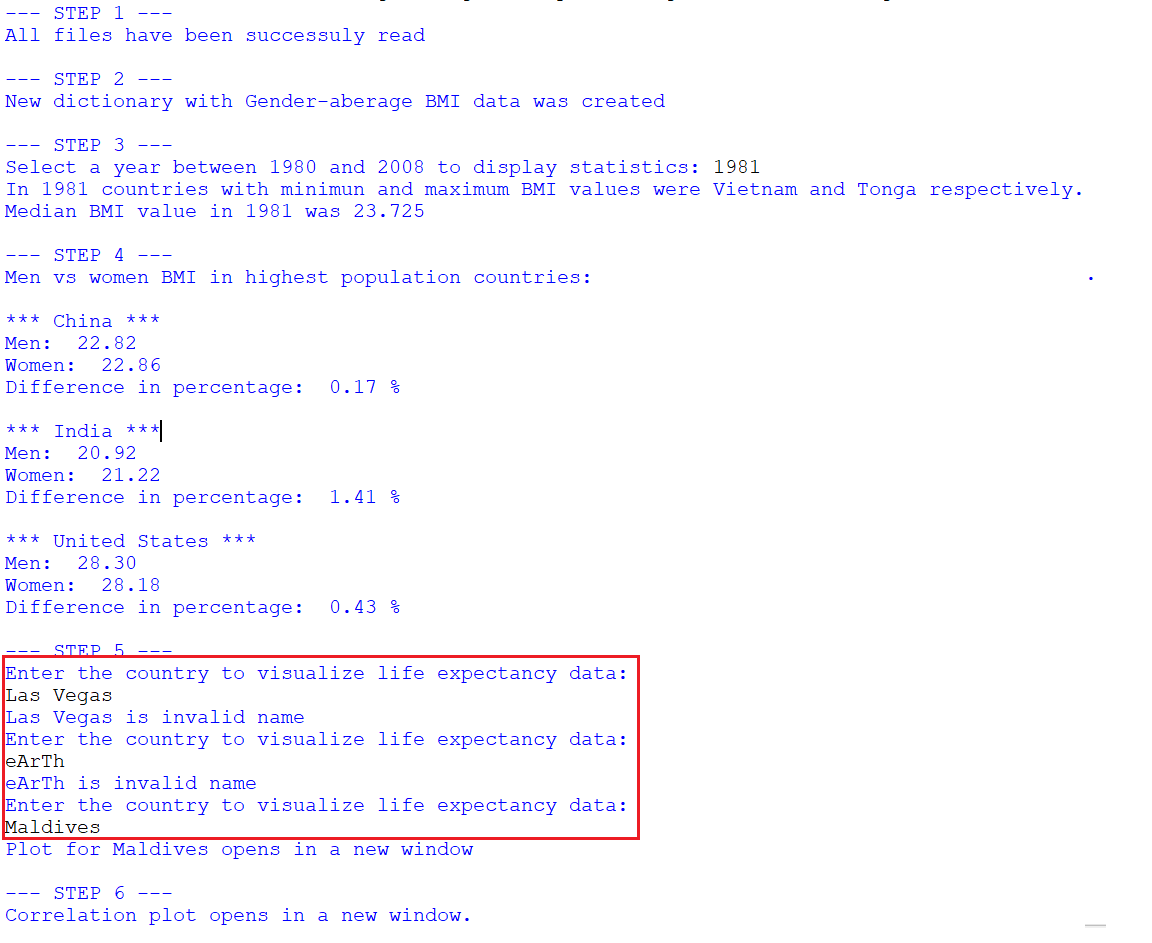
****

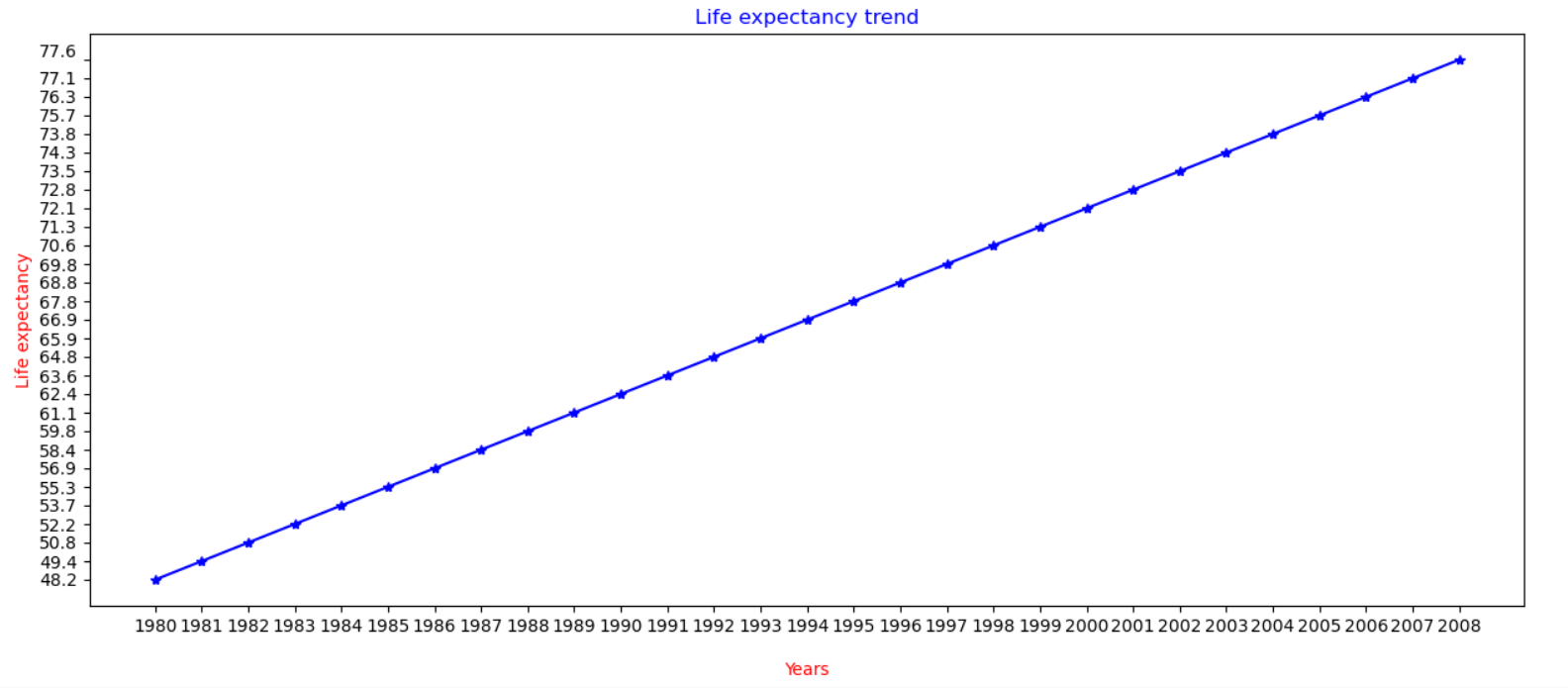


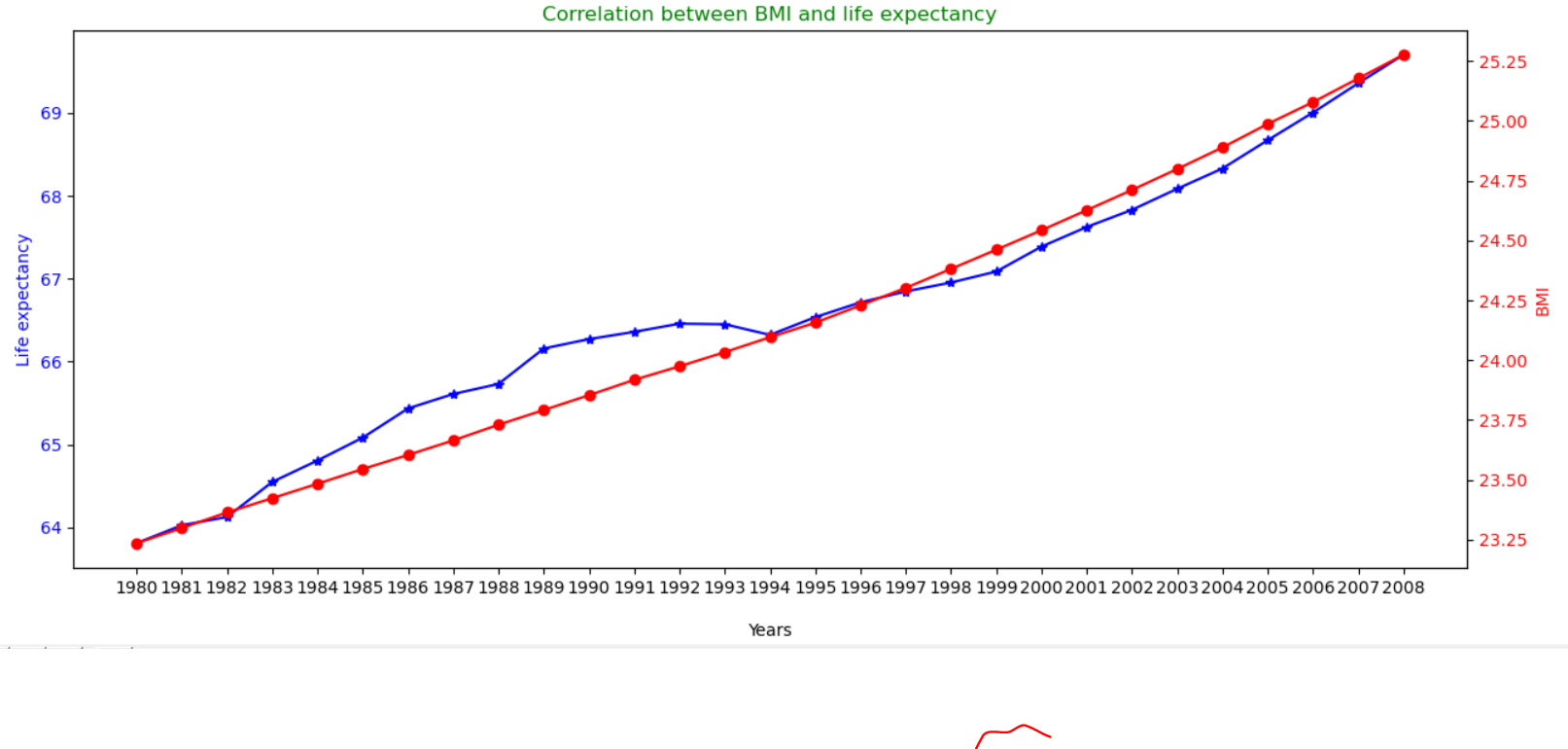


| **Test Data Table** | | | |
| --- | --- | --- | --- |
| **Test data type** | **Test data** | **The reason it was selected** | **The output expected due to the use of the test data** |
| 1. Abnormal | Year: 2002  Country1:  Las Vegas  Country2:  eArTh  Country3:  Maldives | To test program operation for selected year and input of invalid country names typed with upper and lower cases | Exception will be raised in Step 5, there should appear error message. After that, new input should be required |

**Output:**

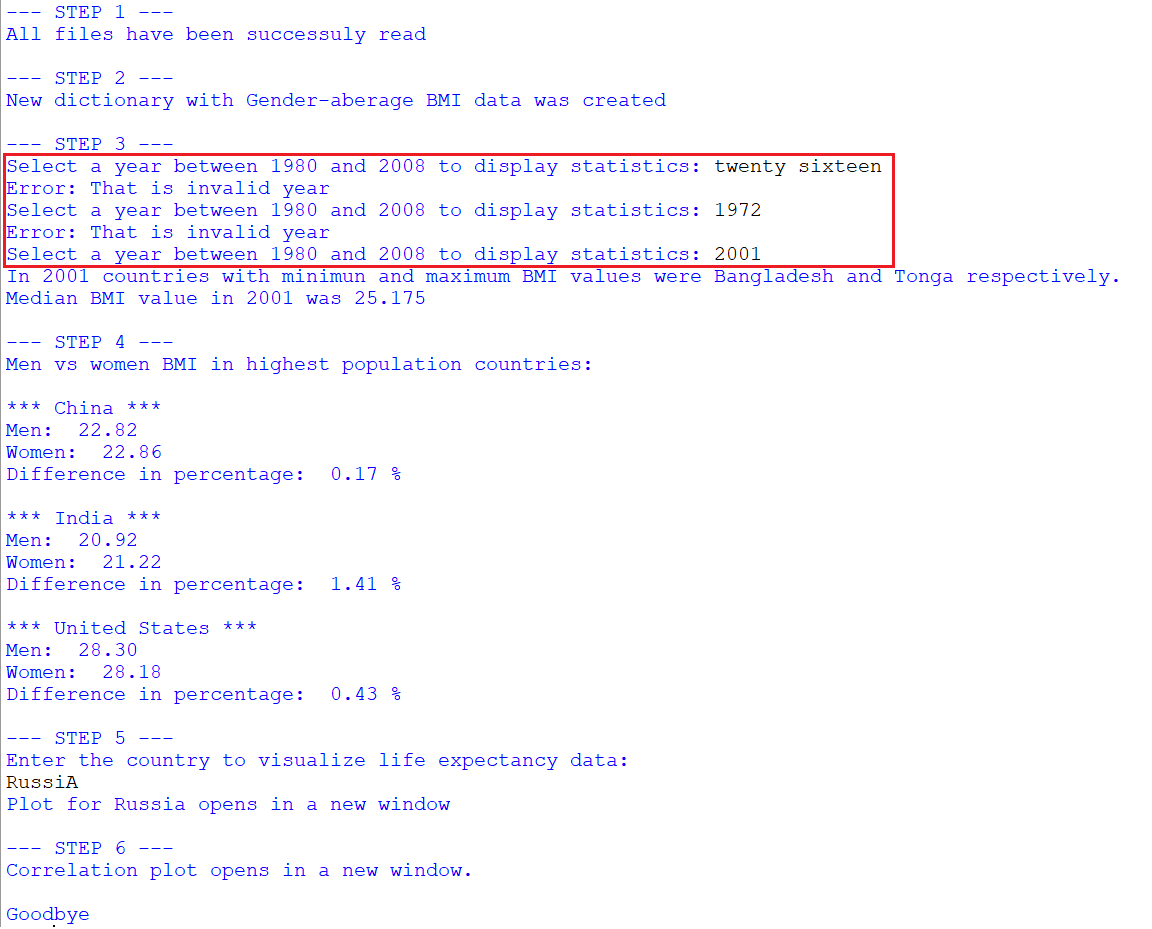
****

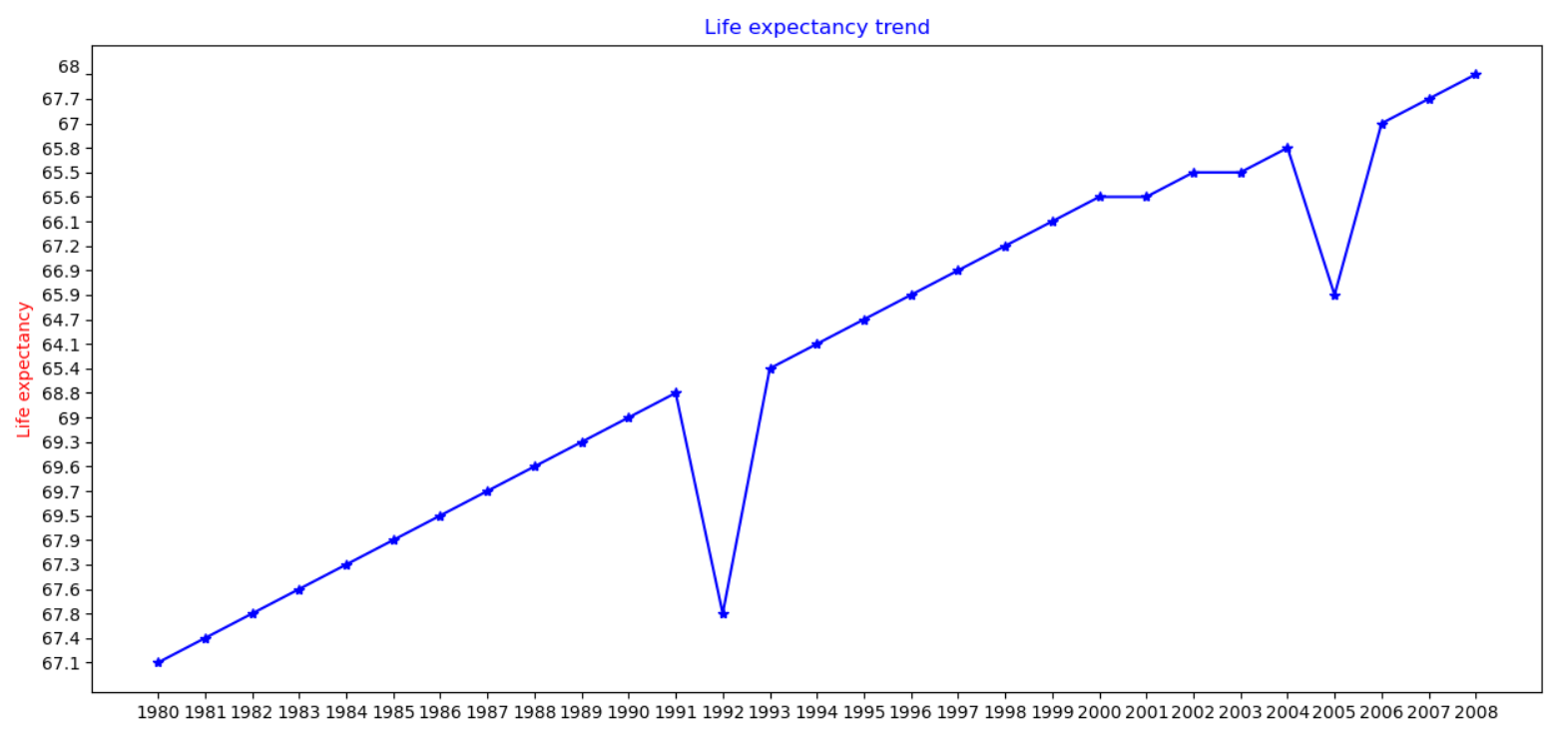


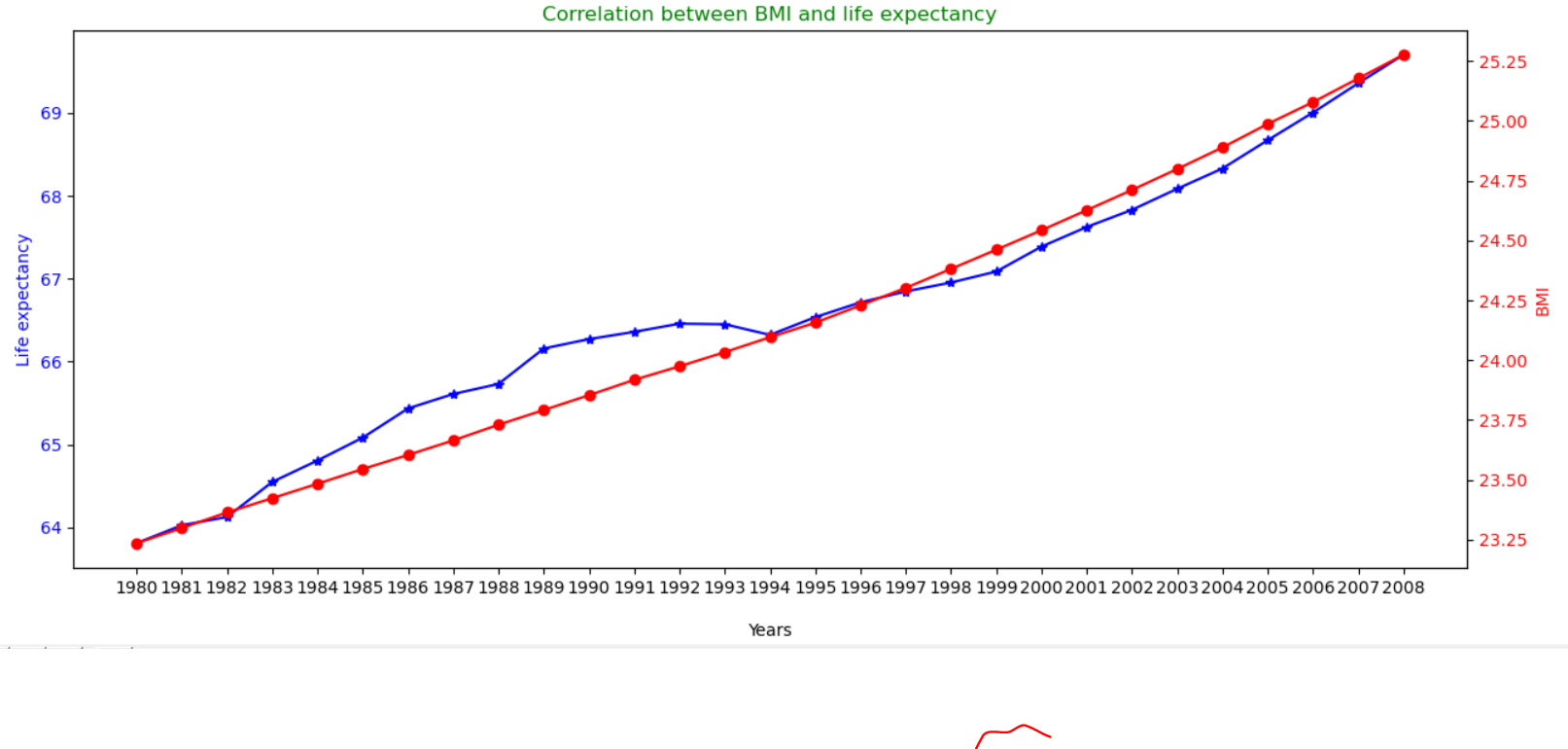


| **Test Data Table** | | | |
| --- | --- | --- | --- |
| **Test data type** | **Test data** | **The reason it was selected** | **The output expected due to the use of the test data** |
| 1. Abnormal | Year1: twenty sixteen  Year2:  1972  Year3:  2001  Country1:  RussiA | To test program operation for year typed by letters, then year greater than range and selected country | Exception will be raised in Step 3, there should appear error message. After that, new input should be required |

**Output:**

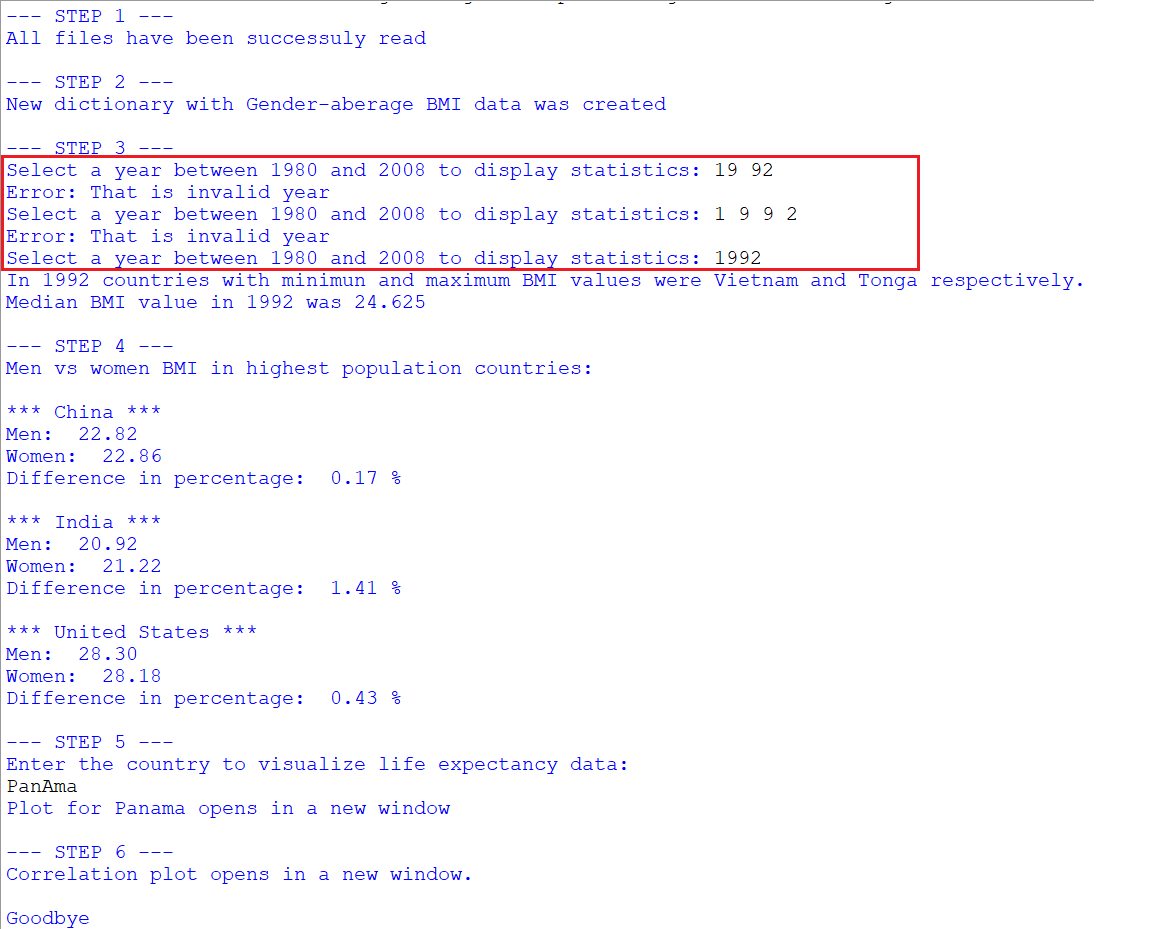
****

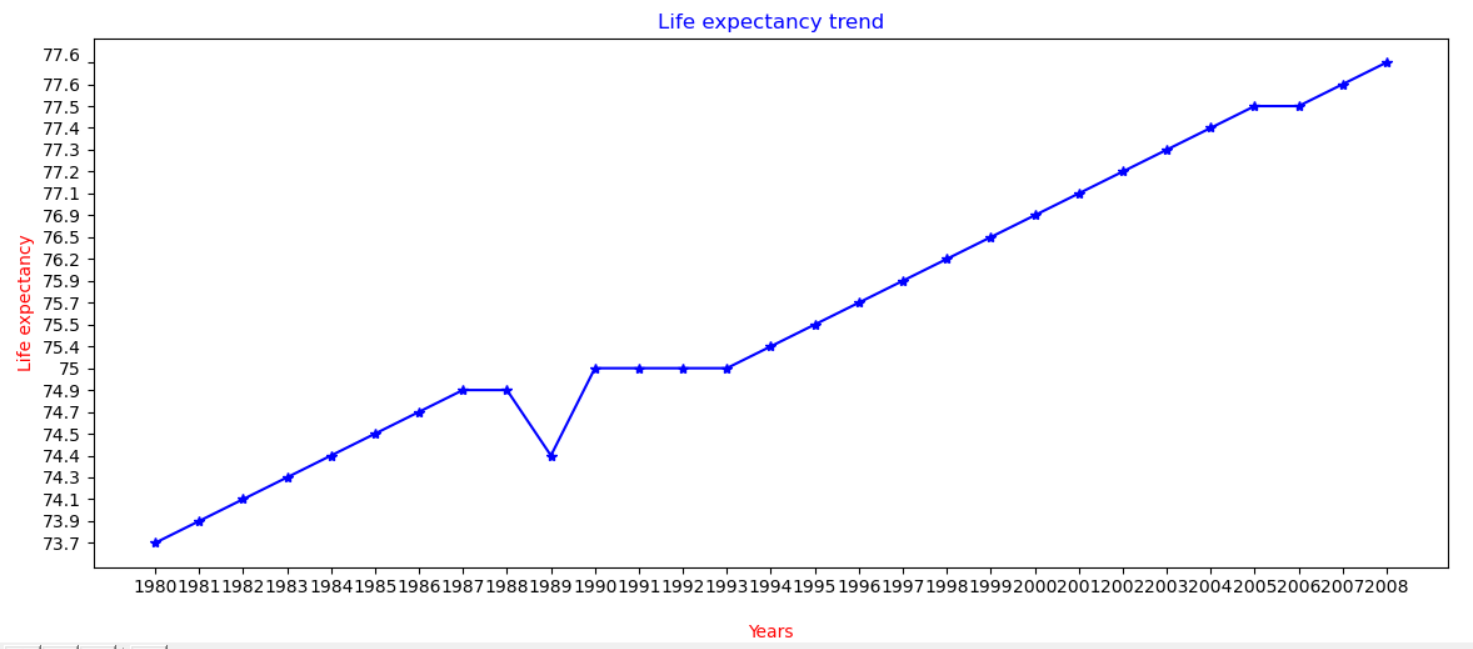


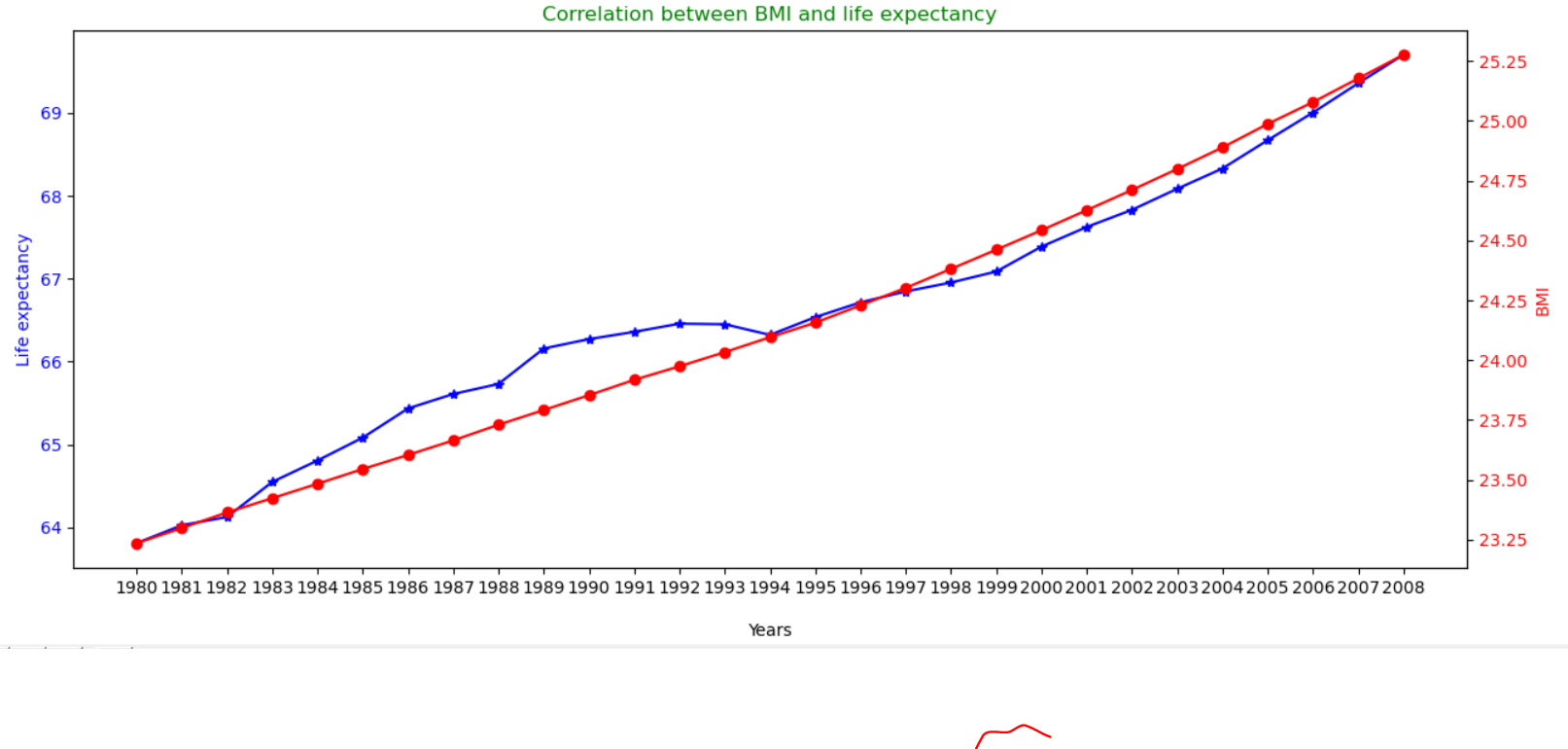


| **Test Data Table** | | | |
| --- | --- | --- | --- |
| **Test data type** | **Test data** | **The reason it was selected** | **The output expected due to the use of the test data** |
| 1. Abnormal | Year1: 19 92  Year2: 1 9 9 2  Year3: 1992  Country1:  PanAma | To test program operation for year typed by with white spaces and selected country | Exception will be raised in Step 3, there should appear error message. After that, new input should be required |

**Output:**

****





# **Task 3**

## Source Code



## Text Code

# import matplolib for graph

import matplotlib.pyplot as plt

#start main function

def main():

#step1

#call function to read files

print ("--- STEP 1 ---")

try:

women\_dic = read\_file\_dic("bmi\_women.csv")

men\_dic = read\_file\_dic("bmi\_men.csv")

life\_list = read\_file\_list("life.csv")

print ("All files have been successuly read", '\n')

except:

print("Error occured while reading some of the files")

#step2

#create bmi\_all dictionary

print ("--- STEP 2 ---")

try:

create\_bmi\_all()

print("New dictionary with Gender-aberage BMI data was created",'\n')

except:

print("Error occured while creating gender-average dictionary")

#step3

#compute statistics

print ("--- STEP 3 ---")

val = 0

#set loop to countinue till valid input is taken

while val ==0:

try:

#take input

year = int(input("Select a year between 1980 and 2008 to display statistics: "))

#call function

find\_statistics(year)

val =1 #finish loop

#except for invalid value of input

except ValueError:

print("Error: That is invalid year")

val=0 #continue loop

#step4

#statistics of BMI men vs women

print ("--- STEP 4 ---")

print("Men vs women BMI in highest population countries:", '\n')

men\_vs\_women()

#step5

#plot of life expectancy

print ("--- STEP 5 ---")

v=0

while v==0:

try:

country = input("Enter the country to visualize life expectancy data: \n")

life\_data(country.lower())

v=1

except:

print(country, "is invalid name")

v=0

#step6

#plot of corralation between BMI and Life expectancy

print ("--- STEP 6 ---")

print("Correlation plot opens in a new window.", '\n')

correlation()

print ("Goodbye")

#define function to read file and create dictionary

def read\_file\_dic(file\_path):

#call try/except statement

try:

#open file for reading

file\_obj = open (file\_path, "r")

except IOError:

#print IOerror message in case of wrong input

print("Err: File was not found")

#declare empty dictionary for data

data\_dic = {}

#declare empty dictionary for header

header\_dic = {}

#declare empty list for data

list = []

#declare empty list for header

header\_list = []

#define counter

counter = 0

#start for loop for a number of lines

for line in file\_obj:

#increase counter by one for each iteration

counter +=1

#condition for header

if counter ==1:

#seperate lines

values = line.rstrip('\n')

#split values with comma

values = line.split(',')

#identify value with 0 index as a key

key = values[0]

#identify values, starting with index 1, as set of data for header

header\_list = values[1::]

header\_float = [int(i) for i in header\_list]

header\_dic[key] = header\_float

#condition for the rest of data

else:

#seperate lines

values = line.rstrip('\n')

#split values with comma

values = line.split(',')

#identify value with 0 index as a key

key = values[0]

#identify values, starting with index 1, as set of data for list

list = values[1::]

data\_float = [float(i) for i in list]

data\_dic[key] = data\_float

#close file

file\_obj.close()

#return results of dicitionary

return header\_dic, data\_dic

#define function to read file and create 2d list

def read\_file\_list (file\_path):

try:

#open file for reading

file\_obj = open (file\_path, "r")

except IOError:

#print error message in case of wrong input

print("Err: File was not found")

##create emply lists for data and headers

new\_list = []

new\_list\_header = []

#set counter

counter = 0

#start loop for each line of a file

for line in file\_obj:

#increase counter by one for each iteration

counter +=1

#condition for header

if counter ==1:

values = line.strip('\n')

values = line.split(',')

new\_list\_header = values[:]

#condition for the rest of data

else:

#attach every line as a new set of values to a list

new\_list.append(line.split(','))

#close file

file\_obj.close()

#return results of 2dlist

return new\_list\_header, new\_list

#define function to create average bmi\_all dictionary

def create\_bmi\_all():

#reference dictionaries

bmi\_men\_header,bmi\_men\_data=read\_file\_dic("bmi\_men.csv")

bmi\_women\_header,bmi\_women\_data=read\_file\_dic("bmi\_women.csv")

#create bmi\_all dictionary

bmi\_all={}

bmi\_list=[]

#check for same key in men`s an women`s list

for key in list(set(bmi\_men\_data)|set(bmi\_women\_data)):

#get the keys if matched

if bmi\_men\_data.get(key) and bmi\_women\_data.get(key):

i=bmi\_men\_data.get(key)

j=bmi\_women\_data.get(key)

#create zipped list of men`s and women`s data

combined\_list=zip(i,j)

#create a list of combined keys and average values

bmi\_list=[round((x+y)/2,2) for (x,y) in combined\_list]

#set key

bmi\_all[key]=bmi\_list

bmi\_all\_header = bmi\_men\_header

return bmi\_all\_header, bmi\_all

#define function to find statistics of spesific year

def find\_statistics(year):

#reference dictionary

bmi\_all\_header, bmi\_all = create\_bmi\_all()

list = []

list = bmi\_all\_header['country']

#find index of entered year

year\_index = list.index(year)

all\_list = []

year\_list = []

counter = 0

total = 0

min\_value = 100

max\_value = 0

counter\_min = 0

counter\_max = 0

c = 0

for line in bmi\_all:

counter+=1

all\_list = bmi\_all[line]

#choose index of spesific year

value = all\_list[year\_index]

#create a list of values at that index

year\_list.append(value)

#find minimum value

if min\_value >= value:

min\_value = value

counter\_min = counter

#find maximum value

if max\_value < value:

max\_value =value

counter\_max = counter

for key in bmi\_all.keys():

c+=1

#find key of minimum value

if c == counter\_min:

country\_min = key

#find key of maximum value

if c== counter\_max:

country\_max = key

#find median value

year\_list.sort()

l = len(year\_list)

#for even number of values

if l%2 == 0:

med1 = year\_list[l//2]

med2 = year\_list[l//2-1]

med\_value = (med1+med2)/2

#for odd number of values

else:

med\_value = year\_list[l//2]

#print output

print("In", year, "countries with minimun and maximum BMI values were", country\_min, "and", country\_max, "respectively.")

print ("Median BMI value in", year, "was", format(med\_value, ".3f"), '\n')

#define function to compute men vs women ststistics of BMI

def men\_vs\_women():

#reference dictionaries

bmi\_men\_header,bmi\_men\_data=read\_file\_dic("bmi\_men.csv")

bmi\_women\_header,bmi\_women\_data=read\_file\_dic("bmi\_women.csv")

counter = 0

men\_list = []

#create empty lists for 3 countries

china\_list = []

india\_list = []

us\_list = []

##find values for men`s dictionary

for key in bmi\_men\_data.keys():

#find country China in dictionaty

if key=='China':

china\_list = bmi\_men\_data[key]

l = len(china\_list)

#get values for lat 5 years

val1 = china\_list[l-1]

val2 = china\_list[l-2]

val3 = china\_list[l-3]

val4 = china\_list[l-4]

val5 = china\_list[l-5]

china\_avg\_men = (val1+val2+val3+val4+val5)/ 5

#find country India in dictionaty

if key == 'India':

india\_list = bmi\_men\_data[key]

l = len(india\_list)

#get values for lat 5 years

val1 = india\_list[l-1]

val2 = india\_list[l-2]

val3 = india\_list[l-3]

val4 = india\_list[l-4]

val5 = india\_list[l-5]

india\_avg\_men = (val1+val2+val3+val4+val5)/ 5

#find country United States in dictionaty

if key == 'United States':

us\_list = bmi\_men\_data[key]

l = len(us\_list)

#get values for lat 5 years

val1 = us\_list[l-1]

val2 = us\_list[l-2]

val3 = us\_list[l-3]

val4 = us\_list[l-4]

val5 = us\_list[l-5]

us\_avg\_men = (val1+val2+val3+val4+val5)/ 5

##find values for women`s dictionary

for key in bmi\_women\_data.keys():

#find country China in dictionaty

if key=='China':

china\_list = bmi\_women\_data[key]

l = len(china\_list)

#get values for lat 5 years

val1 = china\_list[l-1]

val2 = china\_list[l-2]

val3 = china\_list[l-3]

val4 = china\_list[l-4]

val5 = china\_list[l-5]

china\_avg\_women = (val1+val2+val3+val4+val5)/ 5

#find country India in dictionaty

if key == 'India':

india\_list = bmi\_women\_data[key]

l = len(india\_list)

#get values for lat 5 years

val1 = india\_list[l-1]

val2 = india\_list[l-2]

val3 = india\_list[l-3]

val4 = india\_list[l-4]

val5 = india\_list[l-5]

india\_avg\_women = (val1+val2+val3+val4+val5)/ 5

#find country United States in dictionaty

if key == 'United States':

us\_list = bmi\_women\_data[key]

l = len(us\_list)

#get values for lat 5 years

val1 = us\_list[l-1]

val2 = us\_list[l-2]

val3 = us\_list[l-3]

val4 = us\_list[l-4]

val5 = us\_list[l-5]

us\_avg\_women = (val1+val2+val3+val4+val5)/ 5

#compute percentage difference for China

if china\_avg\_men == china\_avg\_women:

china\_diff = 100

if china\_avg\_men != china\_avg\_women:

china\_diff = (abs(china\_avg\_men - china\_avg\_women)/china\_avg\_women)\*100

#compute percentage difference for India

if india\_avg\_men == india\_avg\_women:

india\_diff = 100

if india\_avg\_men != india\_avg\_women:

india\_diff = (abs(india\_avg\_men - india\_avg\_women)/india\_avg\_women)\*100

#compute percentage difference for US

if us\_avg\_men == us\_avg\_women:

us\_diff = 100

if us\_avg\_men != us\_avg\_women:

us\_diff = (abs(us\_avg\_men - us\_avg\_women)/us\_avg\_women)\*100

#print out results

print("\*\*\* China \*\*\*")

print("Men: ", format(china\_avg\_men, '.2f'))

print("Women: ", format(china\_avg\_women, '.2f'))

print("Difference in percentage: ", format(china\_diff, '.2f'), "%", '\n')

print("\*\*\* India \*\*\*")

print("Men: ", format(india\_avg\_men, '.2f'))

print("Women: ", format(india\_avg\_women, '.2f'))

print("Difference in percentage: ", format(india\_diff, '.2f'), "%", '\n')

print("\*\*\* United States \*\*\*")

print("Men: ", format(us\_avg\_men, '.2f'))

print("Women: ", format(us\_avg\_women, '.2f'))

print("Difference in percentage: ", format(us\_diff, '.2f'), "%", '\n')

#define function to draw life expectancy plot

def life\_data(country):

#reference list

life\_header, life\_list = read\_file\_list("life.csv")

#get country name from user and set it for lower letters

for line in life\_list:

name = line[0]

name.lower()

#check if user`s input equals to country w/ lower letters within the list

if country==name.lower():

#get the list out of line

country\_list = line

print("Plot for", line[0], "opens in a new window", '\n')

##draw life expectancy plot

#set x,y

plt.plot(life\_header[1::], country\_list[1::], 'b\*-')

#name x axes

plt.xlabel("Years", color = 'red')

#name y axes

plt.ylabel("Life expectancy", color = 'red')

#name the graph

plt.title("Life expectancy trend", color='blue')

#display graph

plt.show()

#define function to draw correlation plot

def correlation():

#reference values

bmi\_all\_header, bmi\_all = create\_bmi\_all()

life\_header, life\_list = read\_file\_list("life.csv")

## create a list of average values for life expectancy

#set lists`s values as float numbers

lst = [[float(x) for x in line[1::]]for line in life\_list]

#compute total for each year

totals = [ sum(i) for i in zip(\*lst) ]

#find average for each year

avg\_life = []

for num in totals:

avg\_life.append(num/len(life\_list))

## create a list of average values for BMI

b\_list = []

bmi\_list = []

#compute total for each year

for line in bmi\_all:

b\_list = bmi\_all[line]

bmi\_list.append(b\_list)

total = [ sum(j) for j in zip(\*bmi\_list) ]

#find average for each year

avg\_bmi = []

for k in total:

avg\_bmi.append(k/len(bmi\_list))

##draw correlation plot

#set axes x, y1, y2

x\_data = life\_header[1::]

y1\_data = avg\_life

y2\_data = avg\_bmi

#create x and y1

fig = plt.figure()

ax1 = fig.add\_subplot()

ax1.set\_xlabel('Years')

ax1.plot(x\_data, y1\_data,'b\*-')

ax1.tick\_params(axis='y', labelcolor='blue')

ax1.set\_ylabel('Life expectancy', color='blue')

#create a second axes that shares the same x-axis

ax2 = ax1.twinx()

ax2.plot(x\_data, y2\_data, 'ro-')

ax2.tick\_params(axis='y', labelcolor='red')

ax2.set\_ylabel('BMI', color='red')

#set name for graph

plt.title("Correlation between BMI and life expectancy", color='green')

#display the graph

plt.show()

#call main function

main()