Survey Analyser Report

**Thomas Parker**

**Oxford Brookes**

Contents

[Introduction: 2](#_Toc503470500)

[Functions Explained: 2](#_Toc503470501)

[def calculateAverage(): 2](#_Toc503470502)

[def calculateAverageByDepartment(): 3](#_Toc503470503)

[def displayHighest(): 3](#_Toc503470504)

[def displayLowest(): 3](#_Toc503470505)

[def displayPoorPerformance(): 3](#_Toc503470506)

[def displayExcellentPerformance(): 3](#_Toc503470507)

[Test Cases: 3](#_Toc503470508)

[Source Code: 5](#_Toc503470509)

# Introduction:

A department store has installed devices throughout all its departments to obtain feedback from customers on whether their shopping experience was good, fair or poor. The results of which are stored on a text file ready to be read for further analysis.

The mission of this project will be to create a programme that will read the information stored in this text file and will then display a menu of options in which the user of this programme will be able to use to draw specific information about the votes that were made during the survey.

In this program, I have decided that I will store the data read from the text into a newly created list. Reason being that it is easy to manipulate and draw upon data stored in lists and apply certain commands to that data to meet almost any result. Should a specific piece of data from the list need to been analysed as an integer, this can be easily done using lists as I can dissect which piece of data I wish to use and then convert it accordingly.

# Functions Explained:

## def calculateAverage():

This function will cycle through every department in the list and calculate the good, fair and poor votes by multiplying them by their respective numbers (3, 2, 1). The total of this will then be divided by the total number of votes in the list to calculate the average.

## def calculateAverageByDepartment():

Like the previous function, this function will cycle through a department of the users choosing and then calculate the good, fair and poor votes. The total of this will then be divided by the total number of votes in the specified department to calculate the average.

## def displayHighest():

This function will cycle through each department in the list and calculate the good, fair and poor votes as a percentage of the whole. Those with good votes greater than the total of the poor and fair votes will then be displayed as having a high customer satisfaction rating.

## def displayLowest():

This is like the previous function but instead, if the good votes are either less than the poor votes or the fair votes per department, those departments will be displayed as having a low satisfaction rating.

## def displayPoorPerformance():

Any department with poor and fair votes accumulating over half the total number of votes will be displayed as have a low vote count and thus has a poor performance.

## def displayExcellentPerformance():

Any department with good votes accumulating over sixty percent of votes will be displayed as have a high survey rate and thus has an excellent performance. Although, no department has been found to have achieved this.

# Test Cases:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test ID | Test Case | Expected Result | Actual Result | Pass/Fail | Action Taken | Final Result |
| 1 | Text file is read correctly and data stored in memory | Text file is accurately read and stored in list without any unwanted spaces or characters | Text file is stored but includes “/n” and “/t” | FAIL | Each line is appended to the list individually using a loop to cycle through each one | PASS |
| 2 | Displaying the entered data works correctly | Full list is displayed with all data entries | Program displays the entire list with no complication | PASS | No action needed | PASS |
| 3 | Invalid menu option displays an error message | Program will inform user that an invalid option was entered | Program prints “Option does not exist, please try again." | PASS | No action needed | PASS |
| 4 | Program stops when it should | Program should stop upon users input | When user selects option 0, the program ends and prints “Goodbye.” | PASS | No action needed | PASS |
| 5 | Displaying the sum of the values entered works correctly | Program should sum up all values entered | Program successfully sums up values | PASS | No action needed | PASS |
| 6 | Displaying the average of the values entered works correctly | Program should average up all values entered | Program successfully averages all values entered | PASS | No action needed | PASS |
| 7 | Displaying that information about department works correctly | Program should display information about specific department based on users input | Program is notifying user that the department doesn’t exist on every input | FAIL | Action taken in test case 8 | PASS |
| 8 | Displaying that if department does not exist and error message appears and program asks again | Program should notify user if the entered department doesn’t exist and ask them to select another | Program is notifying user regardless of entry | FAIL | IF statements are now followed by ELIF’s and finally ended with an ELSE statement | PASS |
| 9 | Displaying highest rated works well | Program will display departments with high satisfaction rating | Program displaying less departments than expected as poor votes were calculating fair votes too | FAIL | Poor no longer calculating fair votes too | PASS |
| 10 | Displaying lowest rated works well | Program will display departments with lowest satisfaction rating | Program successfully display correct departments with poor performance ratings | PASS | No action needed | PASS |
| 11 | Displaying poor performance works well | Displays departments with low survey rates of 50% or more of the customers that voted fair or poor | Sum (whole) of all votes not taking in poor votes | FAIL | Sum (whole) now adding all votes | PASS |
| 12 | Displaying excellent performance works well | Displays departments with excellent survey rates of 60% or more of the customers voted good | Does this accurately with no error | PASS | No action needed | PASS |

# Source Code:

|  |
| --- |
| surveyList = []  surveyInt = []  # Displays menu to user to select option  def menu():  print("0 - Quit")  print("1 - Display the menu")  print("2 - Display all the survey information")  print("3 - Given a department display the number of customers that answered the survey and the average result")  print("4 - Display the department(s) with the highest customer satisfaction")  print("5 - Display the department(s) with the lowest customer satisfaction")  print("6 - Display the departments where 50% or more of the customers voted fair or poor.")  print("7 - Display the departments where 60% or more of the customers voted good.")  print("8 - Work out and display the number of people that have used the customer satisfaction devices and the average value.")  thirdNum = int(input("Please select an option: "))  if thirdNum == 0: # Ends survey by exiting loop  print("Goodbye.")  return surveyList  elif thirdNum == 1:  menu()  elif thirdNum == 2:  displayData()  menu()  elif thirdNum == 3:  calculateTotalByDepartment()  calculateAverageByDepartment()  menu()  elif thirdNum == 4:  displayHighest()  menu()  elif thirdNum == 5:  displayLowest()  menu()  elif thirdNum == 6:  displayPoorPerformance()  menu()  elif thirdNum == 7:  displayExcellentPerformance()  menu()  elif thirdNum == 8:  calculateTotal()  calculateAverage()  menu()  else:  print("Option does not exist, please try again.") # Message notifying user of incorrect option entered  menu()  # Reads data from the text file and stores into list  def readData():  del surveyList[:]  surveyFile = open("survey.txt","r")  for line in surveyFile:  line = line.rstrip("\n")  line = line.split()  surveyList.append(line)  for num in surveyList:  surveyInt.append(int(num[1]))  surveyInt.append(int(num[2]))  surveyInt.append(int(num[3]))  surveyFile.close()  print("Data has been read from text file.")  return surveyList  return surveyInt  # Displays the data stored in the list  def displayData():  if surveyList == []:  print("Empty list, please read data.")  else:  print(surveyList)  # Sums up the number of people who took the survey  def calculateTotal():  print("Total number of people who took the survey:", (sum(surveyInt)))  # This function returns the average value of the survey  def calculateAverage():  i = 0  g = 1  f = 2  b = 3  whole = 0  for i in range (len(surveyList)):  good = round(int(surveyList[i][g])) \* 3  fair = round(int(surveyList[i][f])) \* 2  poor = round(int(surveyList[i][b])) \* 1  whole += good + fair + poor  toTal = (sum(surveyInt))  averAge = whole / toTal  print("Average value of the survey rounded is:", round(averAge))    # Sums up the number of people who took the survey per department  def calculateTotalByDepartment():  print("Please select department for total number of votes")  print("0 - Ladies")  print("1 - Gentlemen")  print("2 - Toys")  print("3 - Beauty")  print("4 - Technology")  print("5 - Home")  print("6 - Appliances")  print("7 - Food")  print("8 - Shoes")  print("9 - Children")  thirdNum = int(input("Please select an option: "))  if thirdNum == 0:  print("Number of people who took the survey:", (int(surveyList[0][1]) + int(surveyList[0][2]) + int(surveyList[0][3])))  elif thirdNum == 1:  print("Number of people who took the survey:", (int(surveyList[1][1]) + int(surveyList[1][2]) + int(surveyList[1][3])))  elif thirdNum == 2:  print("Number of people who took the survey:", (int(surveyList[2][1]) + int(surveyList[2][2]) + int(surveyList[2][3])))  elif thirdNum == 3:  print("Number of people who took the survey:", (int(surveyList[3][1]) + int(surveyList[3][2]) + int(surveyList[3][3])))  elif thirdNum == 4:  print("Number of people who took the survey:", (int(surveyList[4][1]) + int(surveyList[4][2]) + int(surveyList[4][3])))  elif thirdNum == 5:  print("Number of people who took the survey:", (int(surveyList[5][1]) + int(surveyList[5][2]) + int(surveyList[5][3])))  elif thirdNum == 6:  print("Number of people who took the survey:", (int(surveyList[6][1]) + int(surveyList[6][2]) + int(surveyList[6][3])))  elif thirdNum == 7:  print("Number of people who took the survey:", (int(surveyList[7][1]) + int(surveyList[7][2]) + int(surveyList[7][3])))  elif thirdNum == 8:  print("Number of people who took the survey:", (int(surveyList[8][1]) + int(surveyList[8][2]) + int(surveyList[8][3])))  elif thirdNum == 9:  print("Number of people who took the survey:", (int(surveyList[9][1]) + int(surveyList[9][2]) + int(surveyList[9][3])))  else:  print("Department does not exist, please try again.")  calculateTotalByDepartment()  # Returns the average value per department  def calculateAverageByDepartment():  print("Please select department for average value")  print("0 - Ladies")  print("1 - Gentlemen")  print("2 - Toys")  print("3 - Beauty")  print("4 - Technology")  print("5 - Home")  print("6 - Appliances")  print("7 - Food")  print("8 - Shoes")  print("9 - Children")  thirdNum = int(input("Please select an option: "))  if thirdNum == 0:  whole = 0  good = round(int(surveyList[0][1])) \* 3  fair = round(int(surveyList[0][2])) \* 2  poor = round(int(surveyList[0][3])) \* 1  whole += good + fair + poor  add = (int(surveyList[0][1]) + int(surveyList[0][2]) + int(surveyList[0][3]))  averAge = whole / add  print("Average value to the nearest whole number:", (round(averAge)))  elif thirdNum == 1:  whole = 0  good = round(int(surveyList[1][1])) \* 3  fair = round(int(surveyList[1][2])) \* 2  poor = round(int(surveyList[1][3])) \* 1  whole += good + fair + poor  add = (int(surveyList[1][1]) + int(surveyList[1][2]) + int(surveyList[1][3]))  averAge = whole / add  print("Average value to the nearest whole number:", (round(averAge)))  elif thirdNum == 2:  whole = 0  good = round(int(surveyList[2][1])) \* 3  fair = round(int(surveyList[2][2])) \* 2  poor = round(int(surveyList[2][3])) \* 1  whole += good + fair + poor  add = (int(surveyList[2][1]) + int(surveyList[2][2]) + int(surveyList[2][3]))  averAge = whole / add  print("Average value to the nearest whole number:", (round(averAge)))  elif thirdNum == 3:  whole = 0  good = round(int(surveyList[3][1])) \* 3  fair = round(int(surveyList[3][2])) \* 2  poor = round(int(surveyList[3][3])) \* 1  whole += good + fair + poor  add = (int(surveyList[3][1]) + int(surveyList[3][2]) + int(surveyList[3][3]))  averAge = whole / add  print("Average value to the nearest whole number:", (round(averAge)))  elif thirdNum == 4:  whole = 0  good = round(int(surveyList[4][1])) \* 3  fair = round(int(surveyList[4][2])) \* 2  poor = round(int(surveyList[4][3])) \* 1  whole += good + fair + poor  add = (int(surveyList[4][1]) + int(surveyList[4][2]) + int(surveyList[4][3]))  averAge = whole / add  print("Average value to the nearest whole number:", (round(averAge)))  elif thirdNum == 5:  whole = 0  good = round(int(surveyList[5][1])) \* 3  fair = round(int(surveyList[5][2])) \* 2  poor = round(int(surveyList[5][3])) \* 1  whole += good + fair + poor  add = (int(surveyList[5][1]) + int(surveyList[5][2]) + int(surveyList[5][3]))  averAge = whole / add  print("Average value to the nearest whole number:", (round(averAge)))  elif thirdNum == 6:  whole = 0  good = round(int(surveyList[6][1])) \* 3  fair = round(int(surveyList[6][2])) \* 2  poor = round(int(surveyList[6][3])) \* 1  whole += good + fair + poor  add = (int(surveyList[6][1]) + int(surveyList[6][2]) + int(surveyList[6][3]))  averAge = whole / add  print("Average value to the nearest whole number:", (round(averAge)))  elif thirdNum == 7:  whole = 0  good = round(int(surveyList[7][1])) \* 3  fair = round(int(surveyList[7][2])) \* 2  poor = round(int(surveyList[7][3])) \* 1  whole += good + fair + poor  add = (int(surveyList[7][1]) + int(surveyList[7][2]) + int(surveyList[7][3]))  averAge = whole / add  print("Average value to the nearest whole number:", (round(averAge)))  elif thirdNum == 8:  whole = 0  good = round(int(surveyList[8][1])) \* 3  fair = round(int(surveyList[8][2])) \* 2  poor = round(int(surveyList[8][3])) \* 1  whole += good + fair + poor  add = (int(surveyList[8][1]) + int(surveyList[8][2]) + int(surveyList[8][3]))  print("Average value to the nearest whole number:", (round(avy)))  elif thirdNum == 9:  whole = 0  good = round(int(surveyList[9][1])) \* 3  fair = round(int(surveyList[9][2])) \* 2  poor = round(int(surveyList[9][3])) \* 1  whole += good + fair + poor  add = (int(surveyList[9][1]) + int(surveyList[9][2]) + int(surveyList[9][3]))  averAge = whole / add  print("Average value to the nearest whole number:", (round(averAge)))  else:  print("Department does not exist, please try again.")  calculateAverageByDepartment()  # Displays department(s) with highest customer satisfaction  def displayHighest():  i = 0  b = 1  n = 2  g = 3  for i in range (len(surveyList)):  whole = int(surveyList[i][b]) + int(surveyList[i][2]) + int(surveyList[i][3])  good = round(int(surveyList[i][b]) / whole \* 100)  fair = round(int(surveyList[i][n]) / whole \* 100)  poor = round(int(surveyList[i][g]) / whole \* 100)  poorAndFair = poor + fair  if good > poorAndFair:  print(surveyList[i][0], "has a high customer satisfaction rating:", str(good) + "%", "voted good,", str(fair) + "%", "voted fair and", str(poor) + "%", "voted poor.")  # Displays department(s) with lowest customer satisfaction  def displayLowest():  i = 0  b = 1  n = 2  g = 3  for i in range (len(surveyList)):  whole = int(surveyList[i][b]) + int(surveyList[i][2]) + int(surveyList[i][3])  good = round(int(surveyList[i][b]) / whole \* 100)  fair = round(int(surveyList[i][n]) / whole \* 100)  poor = round(int(surveyList[i][g]) / whole \* 100)  if good < fair or good < poor:  print(surveyList[i][0], "has a low customer satisfaction rating:", str(good) + "%", "voted good,", str(fair) + "%", "voted fair and", str(poor) + "%", "voted poor.")  # Displays departments with low survey rates of 50% or more of the customers that voted fair or poor  def displayPoorPerformance():  i = 0  b = 1  n = 2  g = 3  for i in range (len(surveyList)):  whole = int(surveyList[i][b]) + int(surveyList[i][2]) + int(surveyList[i][3])  good = round(int(surveyList[i][b]) / whole \* 100)  fair = round(int(surveyList[i][n]) / whole \* 100)  poorAndfair = round(int(surveyList[i][g]) / whole \* 100 + fair)  if poorAndfair >= 50:  print(surveyList[i][0], "department voted low,", str(poorAndfair) + "%", "having voted poor or fair.")  # Displays departments with excellent survey rates of 60% or more of the customers voted good.  def displayExcellentPerformance():  i = 0  b = 1  n = 2  g = 3  for i in range (len(surveyList)):  whole = int(surveyList[i][b]) + int(surveyList[i][2]) + int(surveyList[i][3])  good = round(int(surveyList[i][b]) / whole \* 100)  if good >= 60:  print(surveyList[i][0], "has a high survey rate", str(good) + "%", "good")  else:  print("No department displayed an excellent survey rate of 60% or more of the customers voting good.")  break  readData()  menu() |

Listing 1