Beniamin-Alexandru Popa

Module QHO426

Main file

The run function is used to navigate through the program. The function is called at the end of the code if the script is run as the main program. It calls the welcome function, which is defined in the tui module and is used to display a welcome message to the user.

The code then opens the hotel_reviews.csv file and uses a csv.reader object to read the data from the file. The csv.reader object is a generator that provides access to the data in the CSV file row by row. The code uses a generator expression to count the number of rows in the file, which is stored in the counter variable. This is to ensure that the progress function is updated accordingly when loading the files.

The code then rewinds the file to the beginning and uses the csv.reader object to read the data again. This time, it adds each row of data to the reviews list. The progress function, which is also defined in the tui module, is called to display the progress of the data loading process.

After the data has been loaded into the reviews list, the code enters an infinite loop that displays the main menu and waits for the user to make a choice. The main_menu function, which is defined in the process module, is called to display the main menu and return the user's choice. (Appendix 1A)

Based on the user's choice, the code calls the sub_menu function, which is also defined in the process module, to display a sub-menu and get the user's choice. The sub_menu function takes an argument that determines what type of sub-menu to display, either search-related or visualization-related. (Appendix 1B)

The code then performs different actions based on the user's choices. For example, if the user chooses to search for reviews, the code calls functions like nameSearch, dateSearch, nationalitySearch, or dateCompleteSearch to perform the search. If the user chooses to display visual representations of the data, the code calls functions like pieChart, nationalityGraph, or animationGraph to display the desired visualization. (Appendix 1C,)

If the user chooses to exit the program, the code ends the infinite loop and prints a message to indicate that the session is ending. If the user enters an invalid choice, the code calls the error function, which is defined in the process module, to display an error message. (Appendix 1D)

The export function of the program was not attempted.

Process file

The numberReviews(reviews=[]) function takes in a list of reviews as input, and returns the number of reviews in the list. It uses the len() function to calculate the length of the reviews list, and prints the result as a string. (Appendix 2A)

The nameSearch(reviews=[]) function takes in a list of reviews as input, and returns all the reviews that match a specific hotel name entered by the user. The function first calls a function hotel_name() to get the name of the hotel the user wants to search for. It then loops through each element in the reviews list, and checks if the name of the hotel in the current element matches the name entered by the user. If a match is found, the current element (representing a review) is printed. (Appendix 2B)

The dateSearch(reviews=[]) function takes in a list of reviews as input, and returns all the reviews that match a specific date entered by the user. The function first calls a function review_dates() to get the date the user wants to search for. It then loops through each element in the date list, and for each element, loops through the reviews list, checking if the date in the current review matches the date in the date list. If a match is found, the current review is printed. (Appendix 2C)

The nationalitySearch(reviews=[]) function takes in a list of reviews as input, and returns all the reviews that match a specific nationality entered by the user. The function first prompts the user to enter the country of origin (nationality) they want to search for. It then loops through each element in the reviews list, and checks if the nationality in the current review matches the one entered by the user. If a match is found, the current review is printed. (Appendix 2D)

The dateCompleteSearch(reviews=[]) function takes in a list of reviews as input, and returns the overall rating of a hotel on specific dates, counting both positive and negative reviews. The function first sorts the reviews by date (earliest to latest), using a bubble sort algorithm. It then loops through the sorted list, and creates a list of unique dates. For each unique date, it counts the number of positive and negative reviews, and calculates the overall rating as the ratio of positive reviews to the total number of reviews. Finally, it prints the overall rating for each date, rounded to one decimal place. (Appendix 2E)

TUI file

The welcome function is used to display the message "———————————————— Hotel Reviews ————————————————". (Appendix 3A)

The error function takes a string argument msg and displays an error message with the provided string as the error message. The message displayed is "Error! <msg>.". (Appendix 3B)

The progress function takes two arguments: operation and value. It is used to display the progress of an operation. Based on the value of value, the progress is displayed as initiated, in progress, or completed. (Appendix 3C)

The main_menu function is used to display the main menu of the program and receive input from the user. It prints the options [1] Process Data, [2] Visualise Data, [3] Export Data, and [4] Exit. It then prompts the user to enter their choice and returns the value of their choice as an integer. (Appendix 3D)

The sub_menu function takes an argument variant and displays a different sub-menu based on the value of variant. For variant equal to 1, it prints options [1] Reviews for Hotel, [2] Reviews for Dates, [3] Reviews for Nationality, and [4] Reviews Summary. For variant equal to 2, it prints options [1] Positive/Negative Pie Chart, [2] Reviews Per Nationality Chart, and [3] Animated Summary. For variant equal to 3, it prints options [1] All Reviews, and [2] Reviews for Specific Hotel. It then prompts the user to enter their choice and returns the value of their choice as an integer. If the value of variant is 0 or not equal to 1, 2, or 3, it displays an error message. (Appendix 3E, 3F)

The total_reviews function takes an argument num_reviews and displays the message "There are <num_reviews> reviews in the data set.". (Appendix 3G)

The hotel_name function prompts the user to enter the name of a hotel and returns the entered value. (Appendix 3H)

The review_dates function is used to receive input of dates from the user. It prints the message "Do you want to enter another date?" and prompts the user to enter either 1 (for "Yes") or 2 (for "No"). If the

user enters 1, they are prompted to enter a date in the format "mm/dd/yyyy". If the entered date is valid, it is appended to the list of dates. If the user enters 2, the list of dates is returned. (Appendix 3I)

The display_review function takes two arguments: review and cols. It is used to display a review, and cols specifies which columns of the review to display. If cols is not provided or is an empty string, the entire review is displayed. Otherwise, the specified columns are displayed.(Appendix 3J)

The display_reviews function takes two arguments: reviews and cols. It is used to display a list of reviews and cols specifies which columns of the reviews to display. If cols is not provided or is an empty string, the entire review is displayed. Otherwise, the specified columns are displayed. (Appendix 3K, 3L)

The dateCheck function takes a string argument date and returns True if the date is valid (in the format "mm/dd/yyyy") and False if it is not. (Appendix 3M)

The choiceCheck function takes a string argument value and returns True if the value can be converted to an integer and is between 1 and 4 included and False if the value is outside the range. (Appendix 3N)

Visual file

These functions create visual representations of data in the form of pie charts and bar graphs, based on customer reviews of a hotel. The pieChart function generates a pie chart that shows the distribution of positive and negative reviews for a particular hotel. The nationalityGraph function generates a bar graph that shows the distribution of reviews based on the nationality of the reviewer. The animationGraph function is not attempted.

The pieChart function takes in an optional argument reviews, which is a list of lists, each inner list representing a review with information such as the hotel name, nationality of the reviewer, etc. If the argument is not provided, the function assumes that reviews is an empty list.

The first step in the function is to get the name of the hotel using the hotel_name function.

The function then initializes two variables positiveReviews and negativeReviews to store the count of positive and negative reviews, respectively. The function also initializes two variables labels and sizes, which are used for labeling the pie chart.

Next, the function uses a for loop to loop through all the reviews in the reviews list and increment the count of positive and negative reviews accordingly. The criteria for determining whether a review is positive or negative is based on the value of the 4th item in the inner list of each review. If the value is 'No Negative', the review is positive. If the value is 'No Positive', the review is negative. If the value is anything else, both the positive and negative reviews count are incremented.

Finally, the function creates a pie chart using the matplotlib library, with the sizes list representing the size of each slice of the pie and the labels list representing the labels for each slice. The pie chart is then displayed using the plt.show() function. (Appendix 4A)

The nationalityGraph function takes in an optional argument, reviews, which is a list of lists, each inner list representing a review with information such as the hotel name, nationality of the reviewer, etc. If the argument is not provided, the function assumes that reviews is an empty list.

The function starts by initializing two lists, countries and reviewsCounter, which will be used to store the unique countries of the reviewers and the number of reviews from each country, respectively.

The function then uses two nested for loops to first find all the unique countries and count the number of reviews from each country. The first for loop loops through all the reviews and adds the country of each reviewer to the countries list if it doesn't already exist. At the same time, it also initializes the count of reviews from that country in the reviewsCounter list. The second for loop loops through the countries list and increments the count of reviews from each country in the reviewsCounter list if the country of the current review matches the current country being processed.

The function then sorts the countries and reviewsCounter lists in descending order based on the number of reviews from each country. If there are more than 15 countries, the function aggregates the count of reviews from all countries except the top 15 into a new category named "Others."

Finally, the function creates a bar graph using the matplotlib library, with the sortedCountries list representing the x-axis and the sortedReviews list representing the y-axis. The bar graph is then displayed using the plt.show() function. (Appendix 4B)

The animationGraph function takes in an optional argument reviews, which is a list of reviews. The function was not attempted.

Appendix 1

Appendix1A

```python
from process import *
from visual import *
import csv


def run():
    # Task 12: Call the function welcome of the module 'tui'.
    # This will display our welcome message when the program is executed.

    welcome()

    # Task 13: Load the data.
    # - Use the appropriate function in the module 'tui' to display a message to indicate that the data loading
    # operation has started.
    # - Load the data. Each line in the file should represent a review in the list 'reviews_data'.
    # You should appropriately handle the case where the file cannot be found or loaded.
    # - Use the appropriate functions in the module 'tui' to display a message to indicate how many reviews have
    # been loaded and that the data loading operation has completed.

    reviews = []

    with open('data\hotel_reviews.csv') as file:
        read = csv.reader(file)
        counter = sum(1 for row in read)
        print(counter)
        file.seek(0)
        read = csv.reader(file)
        check = 0
        progress("Loading Review List", check)
        for row in read:
            reviews.append(row)
            if int(check) != int(len(reviews)/counter*100):
                check = int(len(reviews)/counter*100)
                progress("Loading Review List", check)

    while True:
        # Task 14: Using the appropriate function in the module 'tui', display the main menu
        # Assign the value returned from calling the function to a suitable local variable

        choice1 = main_menu()
```

Appendix 1B

```python

        if choice1 == 1:
            choice2 = sub_menu(1)
            if choice2 == 1:
                nameSearch(reviews)
            elif choice2 == 2:
                dateSearch(reviews)
            elif choice2 == 3:
                nationalitySearch(reviews)
            elif choice2 == 4:
                dateCompleteSearch(reviews)

```

Appendix 1C

```
117
118         elif choice1 == 2:
119             choice2 = sub_menu(2)
120             if choice2 == 1:
121                 pieChart(reviews)
122             elif choice2 == 2:
123                 nationalityGraph(reviews)
124             elif choice2 == 3:
125                 animationGraph()
126
```

Appendix 1D

```
138
139         elif choice1 == 3:
140             pass
141
142         # Task 26: Check if the user selected the option for exiting the program.
143         # If so, then break out of the loop
144
145         elif choice1 == 4:
146             print("Ending session!")
147             break
148
149         # Task 27: If the user selected an invalid option then use the appropriate function of the
150         # module tui to display an error message
151         elif 4 < choice1 or choice1 < 1:
152             error('Incorrect Choice!')
153
154
155 if __name__ == "__main__":
156     run()
157
```

Appendix 2

Appendix 2A

```
28
29  def numberReviews(reviews=[]):
30      print(f'The total number of reviews in the list is {len(reviews)}')
31
```

Appendix 2B

```
32
33  def nameSearch(reviews=[]):
34      progress('Search by Hotel Name', 0)
35      hotelName = hotel_name()
36      for i in reviews:
37          if i[1] == hotelName:
38              print(i)
39      progress('Search by Hotel Name', 100)
40
```

Appendix 2C

```
41
42  def dateSearch(reviews=[]):
43      progress('Search by Date', 0)
44      date = review_dates()
45      for i in date:
46          for j in reviews:
47              if i == j[0]:
48                  print(j)
49      progress('Search by Date', 100)
50
```

Appendix 2D

```
51
52  def nationalitySearch(reviews=[]):
53      progress('Search by Nationality', 0)
54      nationality = input("Enter the country of origin: ")
55      for i in reviews:
56          if nationality == i[2]:
57              print(i)
58      progress('Search by Nationality', 100)
59
```

Appendix 2E

```
def dateCompleteSearch(reviews=[]):
    sortedReviews = reviews
    progress('Loading reviews in order', 0)
    for i in range(len(sortedReviews)):
        for j in range(1, len(sortedReviews) - i - 1):
            month1, day1, year1 = sortedReviews[j][0].split("/")
            month2, day2, year2 = sortedReviews[j + 1][0].split("/")
            month1 = int(month1)
            day1 = int(day1)
            year1 = int(year1)
            month2 = int(month2)
            day2 = int(day2)
            year2 = int(year2)
            if year1 > year2:
                sortedReviews[j], sortedReviews[j + 1] = sortedReviews[j + 1], sortedReviews[j]
            elif year1 == year2 and month1 > month2:
                sortedReviews[j], sortedReviews[j + 1] = sortedReviews[j + 1], sortedReviews[j]
            elif year1 == year2 and month1 == month2 and day1 > day2:
                sortedReviews[j], sortedReviews[j + 1] = sortedReviews[j + 1], sortedReviews[j]
    date_list = []
    for i in range(1, len(sortedReviews)):
        if sortedReviews[i][0] not in date_list:
            date_list.append(sortedReviews[i][0])
    for i in range(0, len(date_list)):
        positive_reviews = 0
        negative_reviews = 0
        for j in range(1, len(sortedReviews)):
            if date_list[i] == sortedReviews[j][0] and sortedReviews[j][3] == 'No Negative':
                positive_reviews += 1
            elif date_list[i] == sortedReviews[j][0] and sortedReviews[j][4] == 'No Positive':
                negative_reviews += 1
            elif date_list[i] == sortedReviews[j][0]:
                negative_reviews += 1
                positive_reviews += 1
        print(f'{date_list[i]}: Negative reviews - {negative_reviews}; Positive reviews - {positive_reviews}; Overall rating - ' + str(round((positive_reviews / (positive_reviews + negative_reviews)) * 10, 1)))
    progress('Loading reviews in order', 100)
```

Appendix 3

Appendix 3A

```
def welcome():
    """
    Task 1: Display a welcome message.

    The welcome message should display the title 'Hotel Reviews'.
    The welcome message should contain dashes before and after the title.
    The number of dashes should be equivalent to the number of characters in the title.
    There should be a space between the dashes and the title i.e. a space before and after the title

    :return: Does not return anything.
    """

    print("————————— Hotel Reviews —————————")
```

Appendix 3B

```
def error(msg):
    """
    Task 2: Display an error message.

    The function should display a message in the following format:
    'Error! {error_msg}.'
    Where {error_msg} is the value of the parameter 'msg' passed to this function

    :param msg: a string containing an error message
    :return: does not return anything
    """

    error_msg = msg
    print(f'Error! {error_msg}.')
```

Appendix 3C

```
41
42  def progress(operation, value):
43      """
44      Task 3: Display a message to indicate the progress of an operation.
45
46      The function should display a message in the following format:
47      'Operation: {operation} [{status}].'
48
49      Where {operation} is the value of the parameter passed to this function
50      and
51      {status} is 'initiated' if the value of the parameter 'value' is 0
52      {status} is 'in progress ({value}% completed)' if the value of the parameter 'value' is between,
53      but not including, 0 and 100
54      {status} is 'completed' if the value of the parameter 'value' is 100
55
56      :param operation: a string indicating the operation being started
57      :param value: an integer indicating the amount of progress made
58      :return: does not return anything
59      """
60
61      if value == 0:
62          status = 'initiated'
63          print(f'Operation: {operation} [{status}].')
64      elif 0 < value < 100:
65          status = f'in progress ({value}% completed)'
66          print(f'Operation: {operation} [{status}].')
67      elif value == 100:
68          status = 'completed'
69          print(f'Operation: {operation} [{status}].')
70
```

Appendix 3D

```
71
72  def main_menu():
73      """
74      Task 4: Display the main menu and read the user's response.
75
76      The following options should be displayed:
77
78      '[1] Process Data', '[2] Visualise Data', '[3] Export Data' and '[4] Exit'
79
80      In each of the above cases, the user's response should be read in and returned as an integer
81      corresponding to the selected option.
82      E.g. 1 for 'Process Data', 2 for 'Visualise Data' and so on.
83
84      If the user enters a invalid option then a suitable error should be displayed and the user
85      prompted to try again.
86
87      :return: an integer for a valid selection
88      """
89
90      while True:
91          print('------ Menu ------')
92          print('[1] Process Data')
93          print('[2] Visualise Data')
94          print('[3] Export Data')
95          print('[4] Exit')
96          choice3 = input('Enter your choice: ')
97          if choiceCheck(choice3):
98              if 1 <= int(choice3) <= 4:
99                  return int(choice3)
100         else:
101             error('Incorrect Choice!')
102
```

Appendix 3E

```
103
104 def sub_menu(variant=0):
105     """
106     Task 5: Display a sub menu of options and read the user's response.
```

Appendix 3F

```
    if variant == 0:
        error("Incorrect Choice!")
        return 0
    elif variant == 1:
        print('[1] Reviews for Hotel\n[2] Reviews for Dates\n[3] Reviews for Nationality\n[4] Reviews Summary')
        choice4 = input('Enter your choice: ')
        if choiceCheck(choice4):
            if 1 <= int(choice4) <= 4:
                return int(choice4)
        else:
            error("Incorrect Choice!")
            return 0
    elif variant == 2:
        print('[1] Positive/Negative Pie Chart\n[2] Reviews Per Nationality Chart\n[3] Animated Summary')
        choice4 = input('Enter your choice: ')
        if 1 <= int(choice4) <= 3:
            return int(choice4)
        else:
            error("Incorrect Choice!")
            return 0
    elif variant == 3:
        print('[1] All Reviews\n[2] Reviews for Specific Hotel')
        choice4 = input('Enter your choice: ')
        if choiceCheck(choice4):
            if int(choice4) == 1 or int(choice4) == 2:
                return int(choice4)
        else:
            error("Incorrect Choice!")
            return 0
    else:
        error("Incorrect Choice!")
        return 0
```

Appendix 3G

```
def total_reviews(num_reviews):
    f"""
    Task 6: Display the total number of reviews in the data set.

    The function should display a message in the following format:

    "There are {num_reviews} reviews in the data set."

    Where {num_reviews} is the value of the parameter passed to this function

    :param num_reviews: the total number of reviews in the data set
    :return: Does not return anything
    """

    print(f"There are {num_reviews} reviews in the data set.")
```

Appendix 3H

```
def hotel_name():
    """
    Task 7: Read in and return the name of a hotel.

    The function should ask the user to enter a hotel name e.g. Hotel Arena
    The function should then read in and return the user's response as a string.

    :return: the name of a hotel
    """

    name = input('Enter the hotel name: ')
    return name
```

Appendix 3I

```
def review_dates():
    """
    Task 8: Read in and return a list of review dates.

    The function should ask the user to enter some review dates
    This should be entered in the format mm/dd/yyyy (same as that in the file)
    where dd is two-digit day,
    mm is two digit month and
    yyyy is a four digit year
    e.g. 01/22/2020
    The function should return a list containing the specified review dates.

    :return: a list of review dates
    """

    program_run = True
    date_list = []
    while program_run:
        print(' Do you want to enter another date?\n1. Yes\n2. No')
        choice5 = input()
        if choiceCheck(choice5):
            if int(choice5) == 1:
                date = input('Enter a date (mm/dd/yyyy): ')
                if dateCheck(date):
                    date_list.append(date)
                else:
                    error("Invalid date")
            elif int(choice5) == 2:
                return date_list
            else:
                error("Incorrect Choice!")
```

Appendix 3J

```
def display_review(review, cols=None):
    """
    Task 9: Display a review. Only the data for the specified column indexes will be displayed.
    If no column indexes have been specified, then all the data for the review will be displayed.

    The parameter review is a list of values
    e.g. ["08/03/2017", "Hotel Arena", "Russia", "I am so angry...", "Only the park...", 2.9,   7, "[' Leisure trip ', ' Couple ', ' Duplex Double Room ', ' Stayed 6 nights ']", "0 days"]
    [Note: in the above example, the ... is an ellipsis and is only included here as the text is long]

    The parameter cols is a list of column indexes e.g. [0,3,5]

    The function should display a list of values.
    The displayed list should only consist of those values whose column index is in cols.

    E.g. if cols is [1,3] then for the sample review above the following should be displayed:
    ["Hotel Arena", "I am so angry..."]

    E.g. if cols is [0,1,5] then for the sample review above the following should be displayed:
    ["08/03/2017", "Hotel Arena", 2.9]

    E.g. if cols is an empty list or None then all the values will be displayed
    ["08/03/2017", "Hotel Arena", "Russia", "I am so angry...", "Only the park...", 2.9,    7, "[' Leisure trip ', ' Couple ', ' Duplex Double Room ', ' Stayed 6 nights ']", "0 days"]

    :param review: A list of data values for a review
    :param cols: A list of integer values that represent column indexes
    :return: Does not return anything
    """

    processedReviews = []
    if cols == 'None' or cols == '':
        print(review)
    else:
        for j in cols:
            if j.isdigit():
                processedReviews.append(review[int(j)])
        print(processedReviews)
```

Appendix 3K

```
def display_reviews(reviews, cols):
    """
    Task 10: Display each review in the specified list of reviews.
```

Appendix 3L

```
299
300     processed_review = []
301     for i in range(0, len(reviews)):
302         values = []
303         if cols == 'None' or cols == '':
304             processed_review.append(reviews[i])
305         else:
306             for j in cols:
307                 if j.isdigit():
308                     values.append(reviews[i][j])
309                     processed_review.append(values)
310     for i in range(0, len(processed_review)):
311         print(processed_review[i])
312
```

Appendix 3M

```
321
322 def dateCheck(date):
323     try:
324         month, day, year = date.split("/")
325         month = int(month)
326         day = int(day)
327         year = int(year)
328         if month < 1 or month > 12:
329             return False
330         if day < 1 or day > 31:
331             return False
332         if year < 0:
333             return False
334         return True
335     except:
336         return False
337
```

Appendix 3N

```
313
314 def choiceCheck(value):
315     try:
316         if 1 <= int(value) <= 4:
317             return True
318     except:
319         return False
320
```

Appendix 4

Appendix 4A

```
def pieChart(reviews=[]):
    hotelName = hotel_name()
    labels = ['Positive Reviews', 'Negative Reviews']
    positiveReviews = 0
    negativeReviews = 0
    for i in range(0, len(reviews)):
        if hotelName == reviews[i][1]:
            if reviews[i][3] == 'No Negative':
                positiveReviews += 1
            elif reviews[i][3] == 'No Positive':
                negativeReviews += 1
            else:
                negativeReviews += 1
                positiveReviews += 1
    sizes = [positiveReviews, negativeReviews]
    fig1, ax = plt.subplots()
    ax.pie(sizes, labels=labels, startangle=90)
    ax.axis('equal')
    ax.set_title(hotelName)
    plt.show()
```

Appendix 4B

```
def nationalityGraph(reviews=[]):
    countries = []
    reviewsCounter = []
    for i in range(1, len(reviews)):
        if reviews[i][2] not in countries:
            countries.append(reviews[i][2])
            reviewsCounter.append(0)
    for i in range(0, len(countries)):
        for j in range(1, len(reviews)):
            if countries[i] == reviews[j][2]:
                reviewsCounter[i] += 1
    for i in range(0, len(countries)):
        for j in range(0, len(countries) - i - 1):
            if reviewsCounter[j] < reviewsCounter[j + 1]:
                reviewsCounter[j], reviewsCounter[j + 1] = reviewsCounter[j + 1], reviewsCounter[j]
                countries[j], countries[j + 1] = countries[j + 1], countries[j]
    sortedCountries = countries
    sortedReviews = reviewsCounter
    if len(countries) > 15:
        othersReviews = sum(reviewsCounter[15:])
        sortedCountries = countries[:15] + ['Others']
        sortedReviews = reviewsCounter[:15] + [othersReviews]
    fig, ax = plt.subplots()
    ax.bar(sortedCountries, sortedReviews)
    ax.set_xlabel('Nation')
    ax.set_ylabel('Number of reviews')
    ax.set_title('Reviews per nation')
    plt.xticks(rotation=90)
    plt.show()
```

Appendix 4C

```
73
74  def animationGraph(reviews=[]):
75      # Not attempted
76      pass
77
```