Programming Assignment 3

Getting started

Complete the reading and practice assignments posted on the course schedule. Review class handouts and examples. This assignment requires knowledge of loops, lists and function definitions.

Programming Project: Orders **worth: 20 points**

Create an order summary matrix.

In this assignment you will be processing order data from a fictional coffee shop. The café's owner would like to optimize the business operations and has hired you to create a few summary analyses of the data. The first task concerns determining how many employees are needed in the shop at different times, for which the owner would like to see how many orders came within each hour of the work day.

The data, collected from online and in-person orders, includes a set of uniform records, each storing the following fields

- Date and time of the order. The date ranges over all dates of August, 2018. The time range of orders corresponds to the working hours of the business, which is 6 am until midnight.
- First and last name of the person making the order, if known. `'anon'` represents orders by anonymous customers.
- Number of items ordered for each of the following products: espresso, cappuccino, americano, pastry, muffin, scone.

The data is supplied in a file, *orderlog.py*, which is posted with the assignment. For simplicity, *orderlog.py* contains a definition of a python list, stored in variable called `orderlst`, as follows:

```
orderlst=[
['datetime', 'first name', 'last name', 'espresso', 'cappuccino', 'americano',
'pastry', 'muffin', 'scone'],
['2018-08-03 09:05:54', 'anon', 'anon', 0,1,1,3,0,4],
['2018-08-06 10:17:17', 'Jacquelin', 'Trevale', 0,2,1,3,1,0],
['2018-08-21 12:33:23', 'Zuleyma', 'Pemdevon', 2,2,2,3,2,1],

                        . . . the rest of the content is omitted . . .

]
```

As you can see from the above, `orderlst` is a two-dimensional list, in which the first row represents column titles, and each of the rest of the rows represents a single order, stored as a list. All such lists have similar structure, listing values of the fields described above, as shown.

To use the `orderlst` list in your program, download *orderlog.py* in your project folder and include the following code in the beginning of your program:

```
import orderlog
ORDERS = orderlog.orderlst # rename for brevity
```

This will make the `orderlst` content available to the program through the `ORDERS` global variable.

The program that you write must work as follows.

1. Ask the user how many days of data to summarize.
2. Create and display the order summary *matrix,* summarizing how many orders were placed during each one-hour interval starting from 6 am and ending at midnight, for each day starting from the first of the month until the day number, provided by the user in step 1. Recall, that each row in the ORDERS list represents one order.
3. Ask the user to specify the date, for which a *histogram* will be displayed. The histogram displays each order as one * symbol. Repeat step 3, until -1 is entered.

The following interaction, in which user input is indicated in boldface, illustrates one execution of the program.

```
How many days would you like to include? 10

                ORDER SUMMARY

    TIME \ DAY  |  1  2  3  4  5  6  7  8  9 10
-----------------------------------------------
    6:00 - 6:59 | 22 20 22 23 15 17 16 16 14 17
    7:00 - 7:59 | 19 23 17 10 20 15 18 25 12 18
    8:00 - 8:59 | 18 14 22 20 21 12 23 20 17 12
    9:00 - 9:59 | 24 14 12 15 17 17 26 12 16 20
  10:00 - 10:59 | 15 17 17 14 20  8 21 26 18 13
  11:00 - 11:59 |  8 15 14 25 12 16 15 13 21 17
  12:00 - 12:59 | 22 16 10 16 19 17 16 20 17 19
  13:00 - 13:59 | 12  9 16 19 22 18 22 16 24 15
  14:00 - 14:59 | 19 23 17 15 15 15 25 22 20 17
  15:00 - 15:59 | 17 17 15 16 24 23 16 30 18 16
  16:00 - 16:59 | 21 16 21 18 19 14 19 24 18 11
  17:00 - 17:59 | 15 21 18 19 16 17 19 17 14 18
  18:00 - 18:59 | 24 12 20 15 16 19 18 11 17  9
  19:00 - 19:59 | 18 12 25 24 22 23 17 18 19 18
  20:00 - 20:59 | 20 14 17 20 18 24 20 17 15 25
  21:00 - 21:59 | 19 20 17 24 15 19 18 25 18 21
  22:00 - 22:59 | 18 26 17 15 17 18 21 20 24 22
  23:00 - 23:59 | 22 21 11 19 17 17 15 22 23 13
```

order summary matrix

```
Enter day number from 1 to 10 to see a histogram, or -1 to exit: 1

            NUMBER OF ORDERS PER 60 min FOR DAY 1

    6:00 - 6:59 | **********************
    7:00 - 7:59 | ******************
    8:00 - 8:59 | ******************
    9:00 - 9:59 | ************************
  10:00 - 10:59 | ***************
  11:00 - 11:59 | ********
  12:00 - 12:59 | **********************
  13:00 - 13:59 | ************
```

```
  14:00 - 14:59 | ******************
  15:00 - 15:59 | ****************
  16:00 - 16:59 | ********************
  17:00 - 17:59 | ***************
  18:00 - 18:59 | ************************
  19:00 - 19:59 | ******************
  20:00 - 20:59 | ******************
  21:00 - 21:59 | ******************
  22:00 - 22:59 | ******************
  23:00 - 23:59 | ********************

Enter day number from 1 to 10 to see a histogram, or -1 to exit: 12
Enter day number from 1 to 10 to see a histogram, or -1 to exit: 10

          NUMBER OF ORDERS PER 60 min FOR DAY 10

    6:00 - 6:59 | ****************
    7:00 - 7:59 | ******************
    8:00 - 8:59 | ************
    9:00 - 9:59 | ********************
  10:00 - 10:59 | *************
  11:00 - 11:59 | ****************
  12:00 - 12:59 | ******************
  13:00 - 13:59 | ***************
  14:00 - 14:59 | ****************
  15:00 - 15:59 | ***************
  16:00 - 16:59 | ***********
  17:00 - 17:59 | ******************
  18:00 - 18:59 | *********
  19:00 - 19:59 | ******************
  20:00 - 20:59 | ************************
  21:00 - 21:59 | ********************
  22:00 - 22:59 | ********************
  23:00 - 23:59 | *************

Enter day number from 1 to 10 to see a histogram, or -1 to exit: -1
Bye!
```

Important Notes and Requirements:

1. Your program should not use any global variables except for
   a. `ORDERS`,
   b. variables defining the opening and closing times (6am and 24am, I suggest to express both in minutes), and
   c. a variable defining the length of the time interval for which the data is summarized (60 minutes).

2. You **must** define and use the following functions, plus define and user others as you see fit:

a. Function `composeOrderMatrix(),` with a single parameter, defining the number of days, with a default value of 31. The method should create and **return** a two-dimensional list, representing the order summary matrix shown in the interaction as the shaded part of the order summary display. In the matrix, each column c represents one day's data for day number (c+1). Each row r represents the number of orders in the time interval number r+1 from the beginning of the day.
First, create the two-dimensional list populated with as many rows of 0s, as the number of time intervals that would fit in the work day (must be calculated). The length of each row must equal to the number of days.

b. Function `printOrderSummaryMatrix()`, with a single parameter, which is a two-dimensional list of integers. The function should display the content of the matrix as shown, with the exact formatting, which includes
   i. the header row showing day numbers,
   ii. the leftmost column, showing time intervals,
and displays all items aligned as shown.

c. Function `printHistogram()`, which accepts two parameters: a two-dimensional list storing matrix values and a column number (0-based). The function must display a histogram, as shown twice in the interaction above for column values 0 and 9. The histogram visualizes the numbers from the specified column of the matrix using * symbols. Thus, when passed the order summary matrix and a column number, each * represents a single order placed in the consecutive time periods, as designated on the left of the histogram.

   So, for example, the first histogram in the interaction corresponds to orders in day 1, so the first row displays 22 stars, the second – 19, and so forth.

d. Define method `main(),` to start the program flow, read user input and call other methods as needed.

3. All time/day labels in the charts must be computed from the data and not hardcoded.

4. There should be no code outside of function definitions, except for the definitions of the four global variables and a call to method main.

Hints:

- Remove the first row from the `ORDERS` list to get rid of the column headers.
- Conduct all time arithmetic in minute-based representation, converting from minutes to hours and minutes when you need to display the time intervals.
- Start your development using a small data set so that you could test the correctness of your summary data. You can create a testing copy of the *orderlog.py* with a much smaller, manageable number of orders. Furthermore, you can modify the data to test specific parts of your program.

Grading

The grading schema for this project is roughly as follows:

Your program should compile without syntax errors to receive any credit. If a part of your program is working, you will receive partial credit, but only if the program compiles without syntax errors.

- 2 point will be awarded for correctly handling the input, including repeated entry for invalid input values.
- 8 points for correctly constructing the matrix
- 4 points for correctly displaying the matrix in the specified form
- 4 points for the correct printing of the histogram
- 2 points will be awarded for good programming style, as defined in handout 1. Make sure to include introductory comments to functions, documenting what the function does, the parameters and the return values.

Created by Tamara Babaian on September 18, 2018