Assignment 1 – Small Business Deliveries

Due Date: Wednesday, Feb 10th 2021 at 23:55 (Edmonton time)
Percentage overall grade: 6%
Penalties: No late assignments allowed
Maximum marks: 100

Goal: refresher of Python and hands-on experience with file input/output, built-in data structures, and string manipulation.

Assignment Specifications

The owners of a number of small businesses have teamed up to offer a shared delivery service for their customers. Their customers can order items online from any of their stores during the week, and then on Saturday, the store owners will work together to assemble the items and have them delivered as a single package to each customer's home. This approach should cut down on delivery costs for the store owners, while providing a great service for their customers.

You have been hired to create a Python program that will take all of the orders placed for all of the stores each week, and summarize the information into an easy to read table that tells the store owners how many drivers will be needed to deliver all of the packages for that week. They will need a driver for each zone of the city where at least one delivery needs to be made, with a maximum of 10 deliveries per driver. So, for example, if there are 13 deliveries that need to made to the West zone of the city, 2 drivers will be needed for that zone.

The store owners also want to keep track of the cost of this new delivery service. Assuming they will pay their drivers $12 for every package that they deliver, your program should calculate the total delivery cost for the week. Your program should also calculate the percentage that this delivery cost is of the total amount purchased by all of the customers for the week. Both of these values should be displayed in the summary table.

In addition to calculating the above information, your program should also be able to create an invoice summarizing all of the items that will be delivered to a given address. Note that many different people living at the same address may have made orders on different days during the week, but all items that are to be delivered to the same address should be put together in a single delivery package.

Input

You will be provided with **THREE** text files. These text files will be updated every week (on Saturday), so your program must be able to read in the most up-to-date information every time it is run.

The first text file is called *products.txt*, and it contains information about all of the products that all of the stores offer for purchase online. Specifically, each line contains a unique product ID, followed by a description of the product, followed by the product's price in cents (so it is a whole number). All of

these items are separated by a semicolon (";"), as can be seen in the sample *products.txt* provided with the assignment description.

The second text file is called *orders.txt*, and it contains information about all of the items purchased during the current week. Specifically, each line contains information about one product that has been purchased:

- the date of the purchase (yyyy-mm-dd). e.g. 2021-01-18 represents January 18th, 2021.
- the customer's name.
- the customer's complete address, with the postal code at the very end. The format of a postal code is 3 characters, followed by a single space, followed by 3 characters. e.g. T6G 2E8. You can assume that all postal codes provided in *orders.txt* are correct and valid.
- the product ID of the item purchased.
- the number of those items that were purchased.

All pieces of information are separated by a percent sign ("%"), as can be seen in the sample *orders.txt* provided with the assignment description.

The third text file is called *zones.txt*, and it contains information about which areas of the city (as indicated by the first 3 characters of the postal codes) are included in each delivery zone. Each line will contain the name of the zone, followed by a "#", followed by the 3-character starting sequence of all postal codes that belong to that zone. If there are multiple 3-character postal code starting sequences for that zone, they will all be listed on the same line, separated by a comma (","). For example, there are five postal code starting sequences in the South-East zone of Edmonton, and would be listed as:

```
South-East#T6A,T6B,T6C,T6E,T6P
```

Please keep in mind that as this delivery service becomes more popular, the team of small business owners may decide to split the current zones up into smaller delivery zones or expand their delivery to areas outside the city. So your program must read in the latest information from *zones.txt* each week (i.e. every time the program is run).

Output

Menu:

Your program should output a menu which allows the user to keep selecting options until s/he chooses to quit the program. It should look exactly like this:

```
*********************************************
Welcome to the Small Business Delivery Program
*********************************************
What would you like to do?
1. Display DELIVERY SUMMARY TABLE for this week
2. Display and save DELIVERY ORDER for specific address
3. Quit
>
```

The only valid options for the user to enter are the numbers 1, 2, or 3. If the user enters an invalid choice, your program should display an error message and then re-prompt the user to enter a valid choice. It should continue to re-prompt until a valid choice is entered. For example:

```
*********************************************
Welcome to the Small Business Delivery Program
*********************************************
What would you like to do?
1. Display DELIVERY SUMMARY TABLE for this week
2. Display and save DELIVERY ORDER for specific address
3. Quit
> 4
Sorry, invalid entry. Please enter a choice from 1 to 3.
> abc
Sorry, invalid entry. Please enter a choice from 1 to 3.
>
```

Option 1:
When the user chooses option 1, the delivery summary table should be displayed, and should look like this sample output:

```
*********************************************
Welcome to the Small Business Delivery Program
*********************************************
What would you like to do?
1. Display DELIVERY SUMMARY TABLE for this week
2. Display and save DELIVERY ORDER for specific address
3. Quit
> 1

+---------------+------------+-----------+
| Delivery Zone | Deliveries |  Drivers  |
+---------------+------------+-----------+
| Millwoods     |     2      |     1     |
| North-West    |     10     |     1     |
| South-East    |     5      |     1     |
| West          |     13     |     2     |
+---------------+------------+-----------+
| Total drivers needed                5 |
| Total delivery cost          $  360.00 |
| Delivery cost/purchases          14.0% |
+----------------------------------------+
```

Only zones that have at least one delivery should be included in the table. The delivery zones should be displayed in alphabetical order.

The table should be formatted exactly as in the sample output. Specifically, pay attention to the column widths, text alignments, and borders. For example, the **Delivery Zone** column is left aligned, the **Deliveries** column is center aligned, and the **Drivers** column is center aligned. In the bottom section of

the table, you can safely assume that the total delivery cost will never be greater than $99,999.99, and you will always want to show the delivery cost/purchases as a percentage rounded to one decimal (right aligned).

After the table has been displayed, the main menu is displayed again for the user to make another choice.

Option 2:
When the user chooses option 2, s/he is prompted to enter a complete address. If it does not match an address from *orders.txt* exactly, an error message is displayed and the main menu is displayed again for the user to make another choice.

```
*********************************************
Welcome to the Small Business Delivery Program
*********************************************
What would you like to do?
1. Display DELIVERY SUMMARY TABLE for this week
2. Display and save DELIVERY ORDER for specific address
3. Quit
> 2

Address: 123 doesn't exist
Invalid address.

*********************************************
Welcome to the Small Business Delivery Program
*********************************************
What would you like to do?
1. Display DELIVERY SUMMARY TABLE for this week
2. Display and save DELIVERY ORDER for specific address
3. Quit
>
```

If a valid address is entered, a delivery order listing all items in the package delivered to that address is displayed on the screen **AND** saved to a text file called *invoice.txt*.

```
*********************************************
Welcome to the Small Business Delivery Program
*********************************************
What would you like to do?
1. Display DELIVERY SUMMARY TABLE for this week
2. Display and save DELIVERY ORDER for specific address
3. Quit
> 2
```

```
Address: 13420-114 Ave T5M 2Y5

Delivery for:           13420-114 Ave T5M 2Y5
============================================
Date    Item                          Price
------  ----------------------------  ---------
JAN 17  002 x Sunflower seeds         $   10.38
JAN 17  001 x Lettuce seeds           $    3.29
JAN 17  001 x Cherry tomato seeds     $    4.29
JAN 17  001 x Yellowfin zucchini *    $    5.49
JAN 19  001 x Pumpkin seeds           $    3.29
JAN 20  003 x Garden tools (5-pc)     $   77.97
                                      ---------
                                      $  104.71
```

The format of the delivery order should be exactly as shown above. In particular, note that the address is redisplayed (right-aligned) at the top of the delivery order. If the address is longer than 30 characters, only the first 29 characters are displayed followed by an asterisk ("*").

The items should be sorted in order of delivery date. (If multiple items were ordered on the same date, the relative order of those items does not matter.)

- The first column contains the date. The date should be displayed as the first three letters of the month (all capitalized) followed by a single space and the two-digit day. All together, this should always occupy a field width of 6 (as shown).
- The second column contains the number of items ordered, along with the description. The number of items will not exceed 999, and should be right aligned in a field width of 3 with leading zeros. If the item description is longer than 20 characters, only the first 19 characters should be displayed, followed by an asterisk ("*").
- The third column contains the total price charged for that line item. You can assume that the price will not exceed $99,999.99.

At the very bottom of the delivery order, the total amount paid for the package delivered should be listed, aligned under the third column. You can assume that the total will not exceed $99,999.99.

Note that the user can select option 2 before selecting option 1. After the delivery order has been displayed, the main menu is displayed again for the user to make another choice.

Option 3:
When the user chooses option 3, a thank you message will be display and the program will end:

```
*******************************************
Welcome to the Small Business Delivery Program
*******************************************
What would you like to do?
1. Display DELIVERY SUMMARY TABLE for this week
2. Display and save DELIVERY ORDER for specific address
```

```
3. Quit
> 3
Thank you for using the Small Business Delivery Program! Goodbye.
```

Testing

Use the sample input text files to test your code. However, keep in mind that the markers will test your code with DIFFERENT input data in the input text files. So you should also test your code by adding at least one new product to products.txt, one new zone to zones.txt, and additional purchases in orders.txt.

Assessment

In addition to making sure that your code runs properly, we will also check that you follow good programming practices. For example, divide the problem into smaller sub-problems, and write functions to solve those sub-problems so that each function has a single purpose; use the most appropriate data structures for your algorithm; use concise but descriptive variable names; define constants instead of hardcoding literal values throughout your code; include meaningful comments and docstrings to document your code; and be sure to acknowledge any collaborators/references in a header comment at the top of your Python file.

Restrictions for this assignment: you cannot use break/continue, and you cannot import any modules. Doing so will result in deductions.

Rubric

Code quality and adherence to the specifications: 20%
File handling (input/output) and use of appropriate data structures: 15%
Program control and input validation: 10%
Option 1: 35%
Option 2: 20%

Hints for getting started

- Figure out your algorithms BEFORE trying to write your code.
- Break tables down into single lines of strings.
- There's a lot of different data here that forms various relationships – dictionaries will be your friend when solving this problem.
- When doing calculations with money amounts, keep the amounts in cents (i.e. whole numbers) until the very end – i.e. only convert to dollars when you are about to display the amount on the screen. If you convert to float values too early, you may get unexpected results when you try to add or subtract them. This is because computers can't represent the decimal portion of float numbers exactly, just a close approximation.

Submission Instructions

Please follow these instructions to correctly submit your solution:

- All of your code should be contained in a single Python file: **assignment1.py**.
- Make sure that you include your name (as author) in a header comment at the top of assignment1.py, along with an acknowledgement of any collaborators/references.
- Please submit your assignment1.py file via eClass before the due date/time.
- Do not include any other files in your submission.
- Note that late submissions ***will not be accepted***. You can make as many submissions as you would like before the deadline – only your last submission will be marked. So submit early, and submit often.

REMINDER: Plagiarism will be checked for

Just a reminder that, as with all submitted assessments in this course, we use automated tools to search for plagiarism. In case there is any doubt, you **CANNOT** post this assignment (in whole or in part) on a website like Chegg, Coursehero, StackOverflow or something similar and ask for someone else to solve this problem (in whole or in part) for you. Similarly, you cannot search for and copy answers that you find already posted on the Internet. You cannot copy someone else's solution, regardless of whether you found that solution online, or if it was provided to you by a person you know. **YOU MUST SUBMIT YOUR OWN WORK.**