Overview

This project continues the work you did on project 2 but incorporates a more "real-life" mode of handling customer orders, using files to provide input data and including data for multiple orders/customers and for orders having more than one or two item lines.

Objectives

To adapt/enhance/build on an existing code set (using your project 2 program) to meet new requirements as specified in this project description. By completing this project, you should gain experience with and familiarity with:

- reading from text files
- reading from csv files
- using loops for repeated tasks, including condition loops, iteration loops and counter loops.

In addition, you will continue to use the skills from Project 2 such as

- string manipulation
- list manipulation
- string formatting

Project Description

You will again be preparing code to process transactions that come in to The OneOh2 Company, but this time all of the transactions will come from a file. In addition, the initial customer and product information will also come from files instead of pre-defined lists in the starter code.

Information to be Provided

Initially, you will prompt the user for the names of two files: one will contain Customer information, and the other will contain Product information, as described below. Sample files are attached (below) to this project so you can download and review them. Some test cases may use other files, which you will not be able to see, so it is important that your program ask the user for the file names, and be able to open and read any file.

The Customer Information file will be a csv file, with no header row, in which each line has the following fields, which are separated by commas

Information to be Provided

Initially, you will prompt the user for the names of two files: one will contain Customer information, and the other will contain Product information, as described below. Sample files are attached (below) to this project so you can download and review them. Some test cases may use other files, which you will not be able to see, so it is important that your program ask the user for the file names, and be able to open and read any file.

The Customer Information file will be a csv file, with no header row, in which each line has the following fields, which are separated by commas.

- Customer_Number
- Customer_Name
- Customer_Balance
- Customer_Password

For example, the sample file named **`customers.csv`** has several lines, and the first two are:

```
24155,"Carey, Drew Allison",838.41,Bo7&J
24426,"Butler, Geoffrey Barbara",722.93,Ep5&
```

The Product Information file will be a csv file, with no header row, in which each line has the following fields, which are separated by commas:

- Item_Number
- Item_Description
- Item_Price

For example, the sample file named **`inventory.csv`** has several lines, and the first two are:

```
K733,Pens,86.2
LO917,Pencils,74.01
```

Note: It may help you as you read in the csv files to put the information into parallel lists like you were given in Project 2 so that the structure of the data is the same as what you started with in Project 2. This will help you to reuse your code.

Transaction Info

Transaction Info

Your program will prompt for another file name which contains the orders that need to be processed. These are in the form of a text file, where the lines represent transactions described in a similar way they were described in Project 2. There will only be CO (customer order) type transactions for this project, but there are a some differences in the format of the Customer Orders this time, which are described below. There may be many transactions contained in the file, and you will need to continue processing the file until the special marker 'EOF' occurs at the end of the file. The example file you can download is **`orders.txt`**.

Item order format

A customer item order will consist of data in multiple lines. The order will have one "header" record which contains the customer number, the number of item lines in the order, an indication of a request for a Detailed or Summary invoice, and the date of the order. The number of items ordered will be in a 3 character wide field which indicates how many different item lines are in the order, and you will have to read and process each of those additional lines for the specific order. The format of the order header is given below: Note the new field X, which will indicate whether a summary invoice should be printed or a detailed invoice should be printed.

The item order header (first row of data provided) format is

```
          1         2         3         4         5         6         7
12345678901234567890123456789012345678901234567890123456789012345678901234567890
CO#####NNNXMMDDYYYY
```

where

- CO indicates the transaction type
- ##### is the customer number
- NNN is a 3 digit number representing the number of order lines (there will always be 3 digits; if 2 items are ordered NNN will be 002) An order may have up to 999 lines.
- X is a single letter, either a 'D' for 'Detailed' or 'S' for 'Summary' (see below for additional insight)
- MMDDYYYY is the order date in monthdayyear format

The item detail data (the row(s) that follow the header row) is in the following format (same as project 2):

```
quantity^item^MMDDYYYY
```

where

where

- quantity is the quantity being ordered
- item is the product/item number being ordered
- MMDDYYYY is the requested date for the order
- ^ is the separator character between data fields

The following example:

```
CO03251002S10182020
10^A2331^10232020
100^BDLS31234134^10252020
```

*represents a complete **customer order** transaction for customer **03251** with **2** order lines, an indication that a **S**ummary invoice should be produced, order date of **10/18/2020** and one order line for quantity **10** of product **A2331** with a requested date of 10/23/2020, and a second order line for quantity **100** of product **BDLS31234134** and a requested date of 10/25/2020.*

Processing / Output

After reading in the Customer and Product information, and getting the information there into the form you need, you will prompt for the transaction file name. For each customer transaction you read from that file, you will be printing either a detailed or summary invoice.

For each transaction, your program should do the following error checking on the provided data:

- testing for valid customer number (does the customer number exist on the company's customer list?)
- testing for a valid product number (does the product number exist on the company's product list?) *This is the only error checking you need to do -- you may assume all other data provided will be valid. In particular, if you read an NNN field of the number of items ordered, you can be sure there will be exactly that many lines in the file.*

If the customer number is invalid, you should print the following message (assuming 98765 was the invalid customer number):

```
Customer number 98765 is invalid.
```

Even though the customer number is invalid, there will be item lines following the (invalid) CO header. You will need to read past (and ignore) the items that follow the invalid customer until you get to the next CO record. You should also not print any invoice output (either detailed or summary) for an invalid customer.

Even though the customer number is invalid, there will be item lines following the (invalid) CO header. You will need to read past (and ignore) the items that follow the invalid customer until you get to the next CO record. You should also not print any invoice output (either detailed or summary) for an invalid customer.

Customer Item Order

For each Customer Order transaction you will have both a first row containing some data but also some lines following, one for each item ordered.

For each order line, you need to do the following:

- look up the item number in the company product data.
  - If the product number is not in the company's product list, you should use a price of 0, but keep the quantity as specified in the order data.
  - if the product number is in the company's product list, you should use the price and description from the product data for that product and the quantity given on the order item input.
- calculate the total line cost (quantity ordered x item price). (Don't forget to do appropriate rounding)
- if the order requests a Detailed invoice, then you should print the order line information (whether valid item or not) including the calculated line total. If the item is invalid, you should use "*** Item not found ***" as the item description.

A sample output for both a valid and invalid output is presented below (for Detailed invoice only):

```
Ln#    Item #          Item Description          Req Date        Qty      Price          Total
 1     T5333           *** Item not found ***    10/28/2020      447       0.00    $       0.00
 2     R207            Glue                      09/25/2020      402      79.46    $   31942.92
```

Your overall output for a given customer/invoice will depend on whether a Detailed or Summary invoice has been requested:

- invoice heading information: whether a detailed or summary invoice is requested, you will print the same header lines as you did on project 2:
  - Order Date, and
  - Customer
- order item lines (printed only for Detailed invoices):
  - order line headings (column labels)
  - each order line below the column headings)
- invoice summary information (to be printed for all invoices, whether they are Detailed or Summary)

- invoice summary information (to be printed for all invoices, whether they are Detailed or Summary)
  - Total Ordered (a total dollar amount that is the sum of all order line costs)
  - Balance: the customer's account balance from their customer information
  - Total Due: total due from the customer given their account balance before the order, and the total order amount.

Important Note: A customer may have multiple orders in the input file. Because of this, you will need to update the customer's account balance in the customer information at the end of processing each order so that the following order uses an updated balance. For instance, if you have the following information:

```
- customer balance at start of program  of $1000.
- order 1 for the customer with a total of $500
- order 2 for the customer with a total of $250
```

your invoice for order 1 would report the following:

```
Total Ordered  $ 500
Balance        $1000
Total Due      $1500
```

At that point, you want to update the customer's balance to 1500 so that when you process order 2, you produce the following summary:

```
Total Ordered  $ 250
Balance        $1500
Total Due      $1750
```

Your Detailed invoice should be in the format illustrated with the following example outputs in the case of a 2 item order. *Note: the column numbering is to assist you with lining up output and should not be part of your actual output. Also, be sure your browser zoom is set to allow the entire window to be viewed or some text in the display may wrap. In the event you cannot see the full output on either the problem description or when running the program, you can copy and paste from zyBooks into a notepad type application to see the full width formatting.*

```
          1         2         3         4         5         6         7         8         9
12345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901
```

```
         1         2         3         4         5         6         7         8         9
123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890


Order Date:     10/21/2020
  Customer:          77812        Costanza, George Louis

Ln#    Item #            Item Description            Req Date         Qty      Price          Total
 1     LO917             Pencils                    11/21/2020        125      74.01    $   9251.25
 2     IL993             Spiral notebooks           11/25/2020        235      72.70    $  17084.50

                                                                 Total Ordered:          26335.75
                                                                       Balance:            515.55

                                                                    Total Due:           26851.30


123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890

         1         2         3         4         5         6         7         8         9
```

Your Summary invoice is the same, but you leave out the individual items ordered and their headers, as follows. Again, the column numbering is only for assistance in lining up the output and is not part of the actual output.

```
         1         2         3         4         5         6         7         8         9
123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890


Order Date:     10/21/2020
  Customer:          77812        Costanza, George Louis

                                                                 Total Ordered:          26335.75
                                                                       Balance:            515.55
```

```
         1         2         3         4         5         6         7         8         9
123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890


Order Date:     10/21/2020
  Customer:          77812        Costanza, George Louis

                                                          Total Ordered:          26335.75
                                                                Balance:            515.55

                                                             Total Due:           26851.30


123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890

         1         2         3         4         5         6         7         8         9
```

Since you will be potentially printing multiple invoices with each run, you should print a line of dashes after each invoice, and after the message for an invalid customer. You can use the following line:

```
print("---------------")
```

IMPORTANT! There are two components to this lab: pseudo code of your algorithm, (10 points -- D2L hand-in assignment) and the final program (90 points). The pseudo code is due **Tuesday**, Nov 24 at 9:00 PM via D2L Assignment file upload. Your final program is due Friday Dec 4 at 9:00 PM.

Before you start to code, and even before you start to develop pseudo-code, make sure you thoroughly understand the problem. Your pseudocode should clearly outline the activities and the structure your program needs. Your investment of time and thought before you touch the Python keyboard, will give you significant value when you actually do start programming the problem.

Test Cases

There will be some visible test cases and some hidden test cases. A sample of each kind of input file is available for you to download and examine, however, some of the hidden test cases will be based on input files that you will not be able to see. Make sure your program will work for any input as outlined above.

Downloadable files

`customers.csv` , `inventory.csv` `orders.txt` **Download**

main.py Load default template...

```
1  # One of the challenges on project 2 was getting output formatted correctly. Although many students accomplished this with a
2  #   correct variation of string formatting, horizontal formatting is not a primary objective of this project. Accordingly,
3  #   we are providing print formatting strings for the project output in case you would like to use them. These are provided
4  #   in commented statements below.
5  # If you choose to use them, you need to remove the commenting, insert them into your program at the appropriate places, and
6  #   adjust them to match up to the remainder of your program.
7  # You may remove these comments if/when you no longer need them to complete your program.
8  #
9  # Order Date line:
10 #       print("Order Date:{:>15}".format(<you need to fill this in>))
11 # Customer line:
12 #       print("  Customer:{:>15}{:>30}".format(<you need to fill this in>))
13 # Headings line:
14 #       print("{:^3}   {:<18}{:<28}{:^10}{:>11}{:>11}{:>14}".format("Ln#","Item #", "Item Description", "Req Date", "Qty","Pr
15 # Items line:
16 #       print("{:^3d}   {:<18}{:<28}{:^10}{:>11d}{:>11.2f}   ${:>10.2f}".format(<you need to fill this in>))
17 # Summary line:
18 #       print("{:>80}{:>18.2f}".format(<you need to fill this in>))
```

Develop mode **Submit mode**

Run your program as often as you'd like, before submitting for grading. Below, type any needed input values in the first box, then click **Run program** and observe the program's output in the second box.

Predefine program input (optional)

If you'd like to predefine your inputs, provide them here.

Run my program Stop **Clear terminal**

```
>
```