Programming Assignment 5

Getting started

Review class handouts and examples, complete reading and practice assignments posted on the course schedule. This assignment is designed to perform data analysis tasks using Pandas.

Programming Project: Recommendation worth: 25 points

Examine item co-purchasing.

Data and program overview

In this assignment you will be working with data on purchases from retail stores. The task will be to create recommendations for someone who has just bought a product, based on which items are often bought together with the product. The following data will be provided using csv files:

- A table with product information (*prod.csv*); we will call this the *product data.*
- A table with records of customer purchases of products (*purchases.csv*); let's call this the *purchase data.*

The columns in these files are self-explanatory.

I provide two data sets for this assignment, in one zip files. Download and unzip the file into your project folder. Unzipping should result in two folders added to your project folder: *pdata* and *pdata-tiny.* To see the folders in Eclipse, you must execute the Refresh command (see File menu) on the project. You must review the data files to familiarize yourself with their content and structure. For simplicity, the sets of product ids in the product and purchase data will always be equal.

The program that you write must work as follows.

1. Ask the user to specify the subfolder in the current working directory, where the files are stored.
2. Based on the purchase data (*purchases.csv*), compute the *co-purchasing matrix.* An example co-purchasing-matrix corresponding to the data in the pdata-tiny set is shown in Figure 1 below. A co-purchasing matrix in each entry (i, j) lists the number of co-purchases (call it *co-purchase value* or *score*) between item i and item j. For example, the values in row (column) `shampoo` indicate that one customer bought shampoo together with `bowl`; nobody bought shampoo and `broom`, `curtain`, or `fork`; two different customers bought both `shampoo` and `hairdryer`, and so forth.

```
          bowl broom curtain fork hairdryer plate shampoo spoon tablecloth teacup teapot
bowl         0     0       1    0         1     1       1     1          1      0      0
broom        0     0       0    0         0     0       0     0          0      0      0
curtain      1     0       0    0         0     1       0     0          1      0      0
fork         0     0       0    0         0     1       0     0          0      1      1
hairdryer    1     0       0    0         0     0       2     1          0      1      1
plate        1     0       1    1         0     0       0     0          1      1      1
shampoo      1     0       0    0         2     0       0     1          0      1      1
spoon        1     0       0    0         1     0       1     0          0      1      0
tablecloth   1     0       1    0         0     1       0     0          0      0      0
teacup       0     0       0    1         1     1       1     1          0      0      2
teapot       0     0       0    1         1     1       1     0          0      2      0
```

Figure 1. Co-purchasing matrix computed from *purchases.csv* in *pdata-tiny* set.

A way to compute this matrix from purchase data is included in the description of function `fillProductCoPurchase()` further on.

3. Repeat until user requests to terminate the process: ask the user to enter a product that was purchased and use the co-purchasing matrix to determine which product(s) to recommend. The recommendation must list all products that have the highest co-purchase value for the user-specified product. For example, for the co-purchase matrix above, the recommendation for `shampoo` must be `hairdryer,` while recommendation for `bowl` must be `curtain, hairdryer, plate, shampoo, spoon,` and `tablecloth.`

   Note that in the interactions below, the output in the gray rectangle is simply an expanded presentation of the recommended products, first presented as a list of product ids. The expanded presentation includes each product's description, category and price (if available), listed by category.

   In case when the user-entered product was not co-purchased with any product (co-purchasing row consists entirely of 0s, as is the case with the **`broom`** in the interaction below), your program should recommend the most popular product(s), i.e. the product(s) that that have been purchased by more customers than any other product.

   The display of recommended products must use information from the product data (*prod.csv*) and follow the format illustrated in the sample interaction.

The sample interactions below demonstrate the running of the program on the *pdata-tiny* set. Boldface indicates user input. Pay attention to the formatting in the shaded regions.

Sample interaction

```
Please enter name of folder with product and purchase data files: (prod.csv and
purchases.csv): pdata-tiny

Preparing the co-purchasing matrix…

Which product was bought? Enter product id or press enter to quit. broom
[Maximum co-purchasing score 0]
Recommend with BROOM: ['shampoo', 'teacup']
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Suggest one of our most popular products:

IN DINNERWARE -- French Garden Fleurence Teacup, $25.00
IN HAIR CARE  -- Amino Acid Shampoo, $43.00
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

This is just an expanded presentation of the recommended products from the list above.

```
Which product was bought? Enter product id or press enter to quit. bowl
[Maximum co-purchasing score 1]
Recommend with BOWL: ['curtain', 'hairdryer', 'plate', 'shampoo', 'spoon',
'tablecloth']
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
People who bought it were most likely to buy:
```

```
IN DINNERWARE        -- White Elements Plates
IN HAIR CARE         -- Compact Folding Hair Dryer, $123.00
                     -- Amino Acid Shampoo, $43.00
IN SILVERWARE        -- Steel Spoon
IN TABLE LINENS      -- La Classica Luxury Tablecloth
IN WINDOW TREATMENTS -- Sheer Voile Rod Pocket Curtain Panel, $120.00
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Which product was bought? Enter product id or press enter to quit. shampoo
[Maximum co-purchasing score 2 ]
Recommend with SHAMPOO: ['hairdryer']
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
People who bought it were most likely to buy:

IN HAIR CARE -- Compact Folding Hair Dryer, $123.00
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Which product was bought? Enter product id or press enter to quit.
Bye!
```

Important Notes and Requirements

In addition to the requirements stated so far, your code must satisfy the following to gain full credit:

Output and formatting requirements:

- In response to a product entered by user, the program should output:
  - The maximum co-purchasing score, i.e. the maximum score in the entered product's row/column – see Figure 1.
  - The sorted (python) list of recommended product ids – those products that have the co-purchasing score equal to the maximum.
  - As shown by the shaded regions, a list of product names, prices (if available) and categories formatted as follows:
    a. Items are presented by categories derived from the product data.
    b. Categories are sorted alphabetically.
    c. Each category name is listed only once, with the first product in that category, using all capital letters.
    d. The formatting of the recommendation printout should use the length of the longest category in formatting the output in a way that aligns product names.

- Note that the product with maximum co-purchase score equal to 0 (broom) results in output line:
  `Suggest one of our most popular products:`
  while for the other products, you should have a different line:
  `People who bought it were most likely to buy:`
- Your program should not use any global variables and should have no code outside of function definitions, except for a single call to `main`.
- All file related operations should use device-independent handling of paths (use `os.getcwd()` and `os.path.join()` functions to create paths, instead of hardcoding them).

- You must define and use functions specified below in the **Required Functions** section. You may and should define other methods as appropriate.
- You should use the `pandas` data structures effectively to efficiently achieve the goals of your functions and programs.

Required Functions

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

a. (4 points) Function `fillPeopleProducts()` with one parameter of type DataFrame, storing the purchasing data. The function should create and return a new data frame, which will summarize which products were bought by which customer. The new data frame must be indexed by the customer ids (presented in USER_ID column of the purchasing data) with column titles corresponding to product ids (PRODUCT_ID column of the purchasing data). To illustrate the content of the new data frame, here's what the data should look like based on *purchases.csv* in the *pdata-tiny* set. Value 1 indicates that the customer bough the product, 0 – did not buy.

```
       bowl  broom  curtain  fork  hairdryer  plate  shampoo  spoon  tablecloth  teacup  teapot
Amy       0      0        0     0          1      0        1      0           0       1       1
Doug      1      0        1     0          0      1        0      0           1       0       0
Greg      0      1        0     0          0      0        0      0           0       0       0
Joyce     0      0        0     0          0      0        1      0           0       0       0
Lisa      0      0        0     1          0      1        0      0           0       1       1
Sara      0      0        0     0          0      0        0      1           0       1       0
Tom       1      0        0     0          1      0        1      1           0       0       0
```

b. (5 points) Function `fillProductCoPurchase ()` with one parameter of type DataFrame, storing the purchasing data. The function should create a data frame representing the co-purchasing matrix (see Figure 1 for an illustration). To do it, it must first call function `fillPeopleProducts()` to obtain the data frame with the summary of purchases by customer, let's call this data frame `peopleProducts`. Recall, that for each row, representing a customer and column representing a product, `peopleProducts[i,j]` stores 1 if customer i bought product j, and 0 otherwise.

   To create the co-purchasing data frame, first create an empty data frame with both index and columns set to the PRODUCT_ID values found in the purchases data, filled with default values of 0. Then, for each pair of products, product_i and product_j, compute the value for (product_i, product_j) entry in the co-purchasing matrix as follows:

   Let vectors $p_i= (n^i_1, n^i_2, \ldots, n^i_c)$ and $p_j=(n^j_1, n^j_2, \ldots, n^j_c)$ represent respectively the columns from `peopleProducts,` corresponding to product_i and product_j, with c being the number of customers. When product_i != product_j, (product_i, product_j) co-purchasing value equals to the sum:

   $$(n^i_1 * n^j_1 + n^i_2 * n^j_2 + \ldots + n^i_c * n^j_c).$$

   Note that (product_i, product_j) is the same as (product_j, product_i).
   Set (product_i, product_i) equal to 0 for all product_i.

   The `fillProductCoPurchase()` function must return a tuple consisting of the co-purchasing matrix data frame and the `peopleProducts` data frame, returned by `fillPeopleProducts()`.

c. (3 points) Function `findMostBought`(), which will be passed the `peopleProducts` data frame as a parameter, and must return a list of items that have been purchased by more customers than any other item.

d. (3 points) Function `reformatProdData(),` which will be passed the product data frame as a parameter. The product data contains a combination of the product name and category in the DESCRIPTION column. This function must separate the name from the category, leaving only the name in the DESCRIPTION column and creating a new CATEGORY column, with the category name. For example, from the original product DESCRIPTION value `Colorwave Rice Bowl (Dinnerware), Colorwave Rice Bowl` should be stored under DESCRIPTION and `Dinnerware` under CATEGORY.
This function should not return anything.

e. (4 points) Function `printRecProducts()`, which will be passed the product data frame and the list of recommended product ids. The function must produce a printout of all recommendations passed in via the second parameter, in alphabetical order by the category. Make sure to examine the sample interactions and implement the formatting requirements. The function should return no value.

f. (4 points) Function main(), which will be called to start the program and works according to your design.

Hints

- To see the data frames completely when they are printed, you can include the following function calls in your program to set display parameters for pandas

```
pd.set_option('display.max_columns', 1000)
pd.set_option('display.max_rows', 1000)
pd.set_option('display.width', 1000)
```

- When you open and save the csv files in Excel, Excel may change the encoding used for file characters. To return the encoding to the UTF-8 standard that Python likes, open the file in Notepad, and when saving it, specify the encoding as UTF-8.
- Although I have provided a sample small data sets, for testing purposes, I encourage you to create your own one, perhaps based on the pdata-tiny, for which you should know the result in advance.

Grading

The grading schema for this project is roughly as follows:

Points will be awarded, as specified, for the correct implementation of each of the required functions above (which may call other functions that you define), which use data structures, methods and functions of the `pandas/numpy` package appropriately and effectively.

Two points will be awarded for style, as defined by the guidelines in Handout 1.

The code must be sufficiently general to handle different input files, i.e. not tied to the specific content of the files that you are given, though assume the presence of the columns mentioned in the assignment narrative.