Gabriel Martinica

Programming Principles I

Dr. Juan Carlos Rojas

December 3, 2018

1. Objectives

* To create a program that analyzes customer data and outputs a recommendation based on the customer buying habits.
* To correctly use files and data structures.

1. Inputs and outputs to the system
   1. The inputs of the program will be three files: Customers.csv, Transactions.csv and Categories.csv. These files will contain the customer ID and their name, the list of transactions and a list of products assigned to a category. Moreover, there will be two user inputs that the program requires to register a purchase( The customer ID and the product being purchased).A screenshot of a cell phone
      
      Description automatically generated
   2. The outputs of the program will be a message with a product recommendation and name of the customer so that the seller can recommend him/her something else from the store. Another output will be a file that has a sample email that you can copy and paste to an email provider which contains the customer name and another product recommendation. The last output will be a string appended to the Transactions.csv file that updates the file with the new transaction that you register in the program.
2. Detailed description of all functions
   1. customer\_file
      1. It converts the customers.csv file into a dictionary that the program can use.
      2. Its parameter is the customer.csv file.
      3. Returns a dictionary of the file.
   2. Transaction\_file
      1. It converts the Transactions.csv file into a list.
      2. Its parameter is the Transactions.csv file.
      3. Returns a list of the transactions.
   3. Categories\_file
      1. It converts the Categories.csv file into a dictionary.
      2. Its parameter is the Categories.csv file.
      3. Returns a dictionary of the products and their categories.
   4. Register\_transac
      1. It creates a new transaction by creating a list that has two elements, the customer ID and the product. It checks if the product exists, in which case it runs successfully. In the case that the product does not exist in the database, it raises a KeyError which can be handled by the main function of the program by asking for the inputs again.
      2. Its parameters are the customer ID, product and the categories dictionary.
      3. It returns a list.
   5. Customer\_name
      1. This function just looks at the customer dictionary to find the name associated with the customer ID.
      2. Its parameters are the customer ID and the customer dictionary.
      3. It returns the name of the customer(string).
   6. Products\_customer
      1. It generates a list of products bought by the customer. It uses list comprehension to look for the products bought by a single customer ID. Because the transaction list is a list of lists, it looks for the first element in a transaction to check if it matches the customer ID we are looking for, and if it does, it grabs the second element of the list, which is the product and appends it to a new list.
      2. The parameters are the transaction list and the customer ID.
      3. It returns a list that has the products bought by the customer.
   7. Customer\_preferences
      1. This is the heart of the program. This function determines the customer’s preferences. The way it does this is by first creating a list of the products bought by the customer by calling the products\_customer function. Then it initializes counters for each category that start at 0. Then with the products bought and the total list of products with the category, it checks which category is the product the customer bought. For instance, if it is electronics, it adds 1 to the electronics counter. It does this for the rest of the products. In the end, it looks which of the counters has the maximum value and returns a string with the preference of the customer summarized in a category.
      2. The parameters are the customer ID, the categories dictionary and the transaction list.
      3. It returns a string with the customer’s preference.
   8. Products\_by\_Category
      1. This creates a list of lists of the categories of products. It creates three lists which equal the total number of categories in the database, and adds all the products related to those categories. It does it by looking at dictionary keys and finding the value, which is the category.
      2. The parameter is the categories dictionary
      3. Returns a list of lists with a list of categories that have inside a list of products.
   9. Get\_recommendation
      1. This gets the recommendation for the customer. It does this by having the customer’s preference and products bought. Then, it chooses a random product from the preferred customer’s category and checks if the product has been already bought by the customer, in which case it grabs another random product. If the product has not been bought by the customer, it returns a string with the name of the product.
      2. The parameters are the customer’s preference, the categories dictionary, the customer ID and the transaction list.
      3. It returns a string with the recommendation
   10. Update\_transac\_list
       1. This function just appends the register\_purchase return value, which is a list, to the list of transactions.
       2. The parameters are the transaction and the list of transactions and
       3. It does not return anything.
   11. Write\_email
       1. This writes a sample email that the seller can send to the customer really fast. It has a predetermined text that changes for each customer slightly because it has a dynamic customer name and a dynamic recommendation, as each customer is different.
       2. The parameters are the preference, the categories dictionary, the customer dictionary and the customer ID. It requires this many parameters because it calls the get recommendation function and the customer name function. Also, it should be noticed that this recommendation is different from the one that pops up in the screen.
       3. It returns a string with the sample email.
   12. Create\_email\_file
       1. This function just creates a text file with the sample email. It writes to the file.
       2. The parameter is the email and
       3. It outputs a text file.
   13. Append\_transac\_file.
       1. This function appends the new transaction to the already existing Transactions.csv file. It writes to it using csv.writer method.
       2. Its parameter is the purchase and
       3. It does not return anything.
3. Test plan

For the test plan, I create a small database of about 18 purchases, 4 customers and 20 products. I already know the customer’s preference in advance so customer 1 liked electronics, customer 2 liked food and so on. They preferred this because I create transactions based on what I wanted the customer to like. Therefore, I know what the program has to output. I will do several case studies for each customer. I will assign a different category preference for each customer and see what the program outputs.

1. Test results

For the test results, they did output exactly what was intended. This is the program running just when you execute it. I will start with the case study number 1, which is Gabriela Rosales and she likes food.

A screenshot of a cell phone

Description automatically generated

Once you give it an input, it gives you the recommended product along with the customer’s name. A screenshot of a cell phone

Description automatically generated

It also creates a text file with the email.

A screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generated

Moreover, if you fail to give a valid input because you made a mistake, it will ask you to try again until both the ID and the product are in the database. In this case, it will not accept an uppercase letter when it should be lowercase(Core != core). This is case study number 2 and the professor is the perfect candidate for this attempt. He prefers electronics over food and kitchen.

A screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generated

Also, it updated the transaction.csv with the new purchase.A screenshot of a cell phone

Description automatically generated

Observations:

* 1. What worked well?

What worked really well was the transformation of the csv files into dictionaries and lists. This was pretty easy to do.

* 1. What didn’t work, or was particularly hard?

It was hard to write the get\_recommendation function because I needed to think how to analyze the data in forms of dictionaries and lists so that it got the information that it needed. Another hard part was to think what data structure was the best for the database and the way to organize it so that everything fitted correctly.

* 1. What could be done to improve it in the future?

I was expecting to use a little of neural networks in this project but the knowledge was far too advanced to understand it and implement it in two weeks. The neural network would recommend better products according to what you buy instead of choosing a random element from the category, so it will recommend an Iphone charger if you buy an Iphone for example. Another thing that could improve is to be able to register many purchases at the same time, because most of the times people do not buy just one thing but rather many things.