COMP 1405Z
Fall 2022 – Tutorial #2

Objectives

- Practice writing code with for loops, while loops, and nested loops
- Practice writing code that uses files for data storage
- Practice designing algorithms to work with structured file data
- Practice writing functions and using functions to organize code
- Practice following a precise specification

For the first three problems of this tutorial, you will develop a module for performing some sales analysis by processing files with recorded sales information. You should assume that each order saved to a file will consist of 6 lines of information in the following pattern: Customer Name, Number of Desktops Purchased, Number of Laptops Purchased, Number of Tablets Purchased, Number of Toasters Purchased, Total Cost of Purchase. A single file will consist of many orders, resulting in a file with the structure:

Purchase #1 Customer Name
Purchase #1 Number of Desktops
Purchase #1 Number of Laptops
Purchase #1 Number of Tablets
Purchase #1 Number of Toasters
Purchase #1 Total Cost
Purchase #2 Customer Name
Purchase #2 Number of Desktops
Purchase #2 Number of Laptops
Purchase #2 Number of Tablets
Purchase #2 Number of Toasters
Purchase #2 Total Cost
…etc…

The functions you implement must work with any file that follows this format, regardless of the number of orders contained in the file. You do not need to create the files. You can use the test files and the sales-stats-tester.py file provided in the sales-stats directory included within the tutorial zip to see if your code is producing the correct values. You should read through the first three problems of the tutorial first and plan accordingly. If you approach this problem correctly, you will not have to write a lot of code. **Your module should only have variable definitions and function code** (there should be no print statements, etc., outside of the functions). If you want to test your module beyond the provided sales-stats-tester.py file, you can create a secondary .py file, import your module, and run additional tests.

Problem 1 - Total Sales Statistics

Create a file called **salesanalyzer.py** within the sales-stats directory. This file will contain each of the functions outlined in the first three problems of the tutorial. The first function you write must be called get_number_purchases(filename). This function must process the file with the name specified in the filename argument and return an integer representing the total number of purchases that have occurred (that is, the total number of orders, not the total number of products).

Add another function to the salesanalyzer.py file called get_total_purchases(filename). This function must process the file with the name specified in the filename argument and return the total cost of all purchases that have occurred (that is, the sum of the last lines of each purchase record).

Add another function to the salesanalyzer.py file called get_average_purchases(filename). This function must process the file with the name specified in the filename argument and return the average cost of all purchases that have occurred.

Problem 2 - Customer Sales Statistics

Add another function to the salesanalyzer.py file called get_number_customer_purchases(filename, customer). This function must process the file with the name specified in the filename argument and return an integer representing the number of purchases that have been made by a person with the name specified in the customer argument.

Add another function to the salesanalyzer.py file called get_total_customer_purchases(filename, customer). This function must process the file with the name specified in the filename argument and return the total cost of all purchases that have been made by a person with the name specified in the customer argument.

Add another function to the salesanalyzer.py file called get_average_customer_purchases(filename, customer). This function must process the file with the name specified in the filename argument and return the average cost of all purchases that have been made by a person with the name specified in the customer argument.

Problem 3 - Product Sales Statistics

Add a final function to the salesanalyzer.py file called get_most_popular_product(filename). This function must process the file with the name specified in the filename argument and return a string value representing the product that sold the highest number of units (“Desktop”, “Laptop”, “Tablet”, or “Toaster”). In the case of a tie, your function can return any of the highest selling products.

Note: If it makes it easier to program, you can assume that no customer will ever have the name "*". It may be worth thinking about how this could allow you to generalize some of your functions and eliminate duplicated code.

Problem 4 – Searching Files

The course project will involve creating a web crawler and search engine. We will start with a more basic search problem that will search through files. In an abstract sense, though, these files can be thought of as representing pages from the internet. To start this problem, look in the search-engine folder included in the tutorial zip file. Within the folder, there is a pages.txt file, along with some N-X.txt files and a search-results.txt file. The pages.txt file contains a list of the other files included (i.e., the files to search), while the N-X.txt files represent web pages with various words on them (each line contains a word) The search-results.txt file can be used for testing. You can use the pages.txt file to read the names of the other files that your code will have to search through.

Create a **search.py** file within the search-engine directory. When this program starts, the program must ask a user to enter a search word. The program must then tell the user the most relevant page (in this case, file) based on the frequency of the given search word in the page. Your program must print out the most relevant page based on the following two metrics:

1. The maximum frequency of the given word (i.e., the 'page' where the given word occurs most often).
2. The maximum word ratio of the given word (i.e., frequency of the given word divided by total number of words in the 'page').

Once you have implemented your solution, you can use the search-results.txt file to see if it is working correctly. This file contains some sample search words, along with the page(s) with the highest maximum frequency and maximum ratio for the given word.

Submission

Add both the sales-stats and search-engine directories, including your solution files, to a single .zip file named "**tutorial2.zip**" and submit it to Brightspace. Ensure that your file is a .zip and has the proper name. Download your submission afterward and check the contents to ensure everything works as expected.