Ashton Larkin
June 07, 2022
IT FDN 110 A Spring 2022: Foundations of Programming: Python
Assignment08: “Product List”
https://github.com/AshtonUniverse/IntroToProg-Python-Mod08

Introduction:

Module 08 examines the use of the software object that can combine functions or data (methods and attributes). It demonstrates writing classes, the constructor method that automatically creates and initializes object attributes, how to create class attributes and static methods and how to ensure object encapsulation using private attributes and properties. The objective of assignment 08 is to update a provided starter script adding code to the script sections of each step with a comment “# *TODO: Add Code Here*” and to maintain the integrity of the existing logic and formatting of the original programmer.

This document is a breakdown of the logic I used for assignment 08. I will list each step highlighting original code from the starter script and code I added or modified to complete the program. The “Product List” program (Assignment08-Starter.py) consists of four separations of concerns: Data, Processing, Presentation (Input-Output) and the Main Body of the Script. The starter script provided the following ***TODO:*** steps and pseudocode as scripted by the original programmer “RRoot”

```
Data:
    class Product:
        TODO: Add Code to the Product class

Processing:
    class FileProcessor:
        TODO: Add Code to process data from a file
        TODO: Add Code to process data to a file

Presentation
    class IO:
        TODO: Add docstring
        TODO: Add code to show menu to user
        TODO: Add code to get user's choice
        TODO: Add code to show the current data from the file to user
        TODO: Add code to get product data from user

Main Body of Script:
    TODO: Add Data Code to the Main body
        Load data from file into a list of product objects when script starts
        Show user a menu of options
        Get user's menu option choice
        Show user current data in the list of product objects
        Let user add data to the list of product objects
        Let user save current data to file
        Exit program
```

Added my name and date to the change log in the script header:

```
# ------------------------------------------------------------------------ #
# Title: Assignment 08
# Description: Working with classes
# ChangeLog (Who,When,What):
# RRoot,1.1.2030,Created started script
# RRoot,1.1.2030,Added pseudo-code to start assignment 8
# ALarkin,6.7.2022,Modified code to complete assignment 8
# ------------------------------------------------------------------------ #
```

Replaced:

```
# <Your Name>,<Today's Date>,Modified code to complete assignment 8
```

```
# Data -------------------------------------------------------------------- #
strFileName = 'Products.txt'
lstOfProductObjects = []  # A list that acts as a 'table' of rows

class Product:
    """Stores data about a product:
    properties:
       product_name: (string) with the product's name
       product_price: (float) with the product's standard price
    methods:
        to_string(): (str) all properties
    changelog: (When,Who,What)
        RRoot,1.1.2030,Created Class
    ALarkin,6.7.2022,Modified code to complete assignment 8
    """
    # TODO: Add Code to the Product class
    # -- Constructor --
    def __init__(self, product_name: str, product_price: float):
        # -- Attributes --
        self.__product_name = str(product_name)
        self.__product_price = float(product_price)

    # -- Properties --
    # product_name
    @property
    def product_name(self):
        return str(self.__product_name)

@product_name.setter
    def product_name(self, value: str):
        if str(value).isnumeric() == True:
            self.__product_name = value
        else:
            raise Exception("Product names cannot be number!")

    # product_price
    @property
    def product_price(self):
        return float(self.__product_price)
    @product_price.setter
    def product_price(self, value: float):
        if str(value).isnumeric() == False:
            self.__product_price = float(value)
        else:
            raise Exception("Price must be numbers!")

    # -- Methods --
    def to_string(self):
        """ alias of __str__()"""
        return self.__str__()

    def __str__(self):
        """ Convert product data to string"""
        return self.product_name + ',' + str(self.product_price)
# Data -------------------------------------------------------------------- #
```

In the "Data" section I added a Constructor to initialize the product_name and product_price attributes, a getter and setter property for each attribute that include encapsulation of private attributes for each and a value type validation with error exception handling, a __str__() method that converts the product data to a string and a custom method to_string as an alias that can be called by other functions in the script.

```
# Processing  ------------------------------------------------------------- #
class FileProcessor:
    """Processes data to and from a file and a list of product objects:
    methods:
        save_data_to_file(file_name, list_of_product_objects)
        read_data_from_file(file_name): -> (a list of product objects)
    changelog: (When,Who,What)
        RRoot,1.1.2030,Created Class
       ALarkin,6.7.2022,Modified code to complete assignment 8
    """
    # TODO: Add Code to process data from a file
@staticmethod
    def read_data_from_file(file_name: str):
        """Reads data from a file into a list of dictionary rows:
         :param file_name: (string) with name of file:
         :return: (list) of rows
        """
        list_of_rows = []
        try:
            import os.path
            isfile_bln = (os.path.isfile(file_name))
            if (isfile_bln == True):
                file = open(file_name, "r")
                for line in file:
                    data = line.split(",")
                    row = Product(data[0], float(data[1]))
                    list_of_rows.append(row)
                file.close()
        except FileNotFoundError as e:
            print("Error file not found:", e, sep='\n')
        except Exception as e:
            print()
            print("Error reading data from file:", e, sep='\n')
        return list_of_rows
```

The read_data_from_file method is loaded when the program starts and includes a try/except block that contains the variable isfile_bln that performs a validation on the file_name parameter passed to the function "os.path.isfile" and returns a Boolean value (True or False). This function also calls the os. path module. It is useful when processing files from different places in the system and for different purposes such as for merging, normalizing, and retrieving path names in python. If isfile_bln evaluates to "True," file_name is determined to be a valid file and I then load data from Products.txt into the file object and unpack it into the list table variable list_of_rows. I added two exceptions for the try/except block. The first "FileNotFoundError" is redundant code since I am already validating the file with os.path.isfile, however, I added this exception to demonstrate an alternative method for catching the error: FileNotFoundError: [Errno 2] No such file or directory. The last exception is a catch all for general error handling. For all exceptions in this script, I included print() statements for Pythons built in error information commented out to show the optional choice.

```
    # TODO: Add Code to process data to a file
    @staticmethod
    def save_data_to_file(file_name: str, list_of_rows: list):
        """Writes data from a list of dictionary rows to a file:
        :param file_name: (string) with name of file:
        :param list_of_rows: (list) you want filled with file data:
        :return: status_bln (boolean) return status
        """
        save_bln = False
        try:
            strOverwrite = str(input("Overwrite: " + file_name + "?" + " [y/n] ").strip().lower())
            if (strOverwrite == 'y'):
                objFile = open(file_name, "w")
                for row in list_of_rows:
                    objFile.write(row.to_string() + "\n")
                objFile.close()
                save_bln = True
                print()  # Add an extra line for looks
                print("*************")
                print("Data Saved")
                print("*************")
            else:
                print("Overwrite = No | File not overwritten")
        except Exception as e:
            print()  # adding a new line for looks
            print("Error saving data:", e, sep='\n')
        return save_bln
# Processing  ------------------------------------------------------------ #
```

The "save_data_to_file" method executes menu option three to save data. I added a try/except block that begins with the strOverwrite variable set to an input() statement asking the user for a "y/n" (yes or no) to overwrite the data file. If "no" it executes the else: block and prints "Overwrite = No | File not overwritten". If "yes," I call the open() function to open the data file and write the list_of_rows data to the file Products.txt. A print() statement prints "Data Saved" on completion. The exception block catches all general errors.

```
# Presentation (Input/Output)  -------------------------------------------- #
class IO:
    # TODO: Add docstring
    """Performs Input and Output tasks:
    methods:
    menu()
    choice()
    product_list()
    input_data()
    changelog: (When,Who,What)
        ALarkin,6.7.2022,
    """
```

Added a docstring with a description for the IO class, a list of all methods that I added to the class and a changelog.

```
    # TODO: Add code to show menu to user
    @staticmethod
    def menu():
        """
        Display a menu of options to the user

        :return: nothing
        """
        print('''
        ***************************
        Product List - Option Menu
        ***************************
        1) Show current data
        2) Add a product
        3) Save data to file
        4) Exit program
        ***************************
        ''')
        print()  # Add an extra line for looks
```

The menu() method prints a display menu of options to the user that includes four menu options for user selection.

```
    # TODO: Add code to get user's choice
    @staticmethod
    def choice():
        """
        Get users menu choice

        :return: (str) choice
        """
        choice = str(input("Choose an option? [1 to 4] - ")).strip()
        print()  # Add an extra line for looks
        return choice
```

The choice() method prompts the user via an input() function to choose from menu options 1 to 4 and returns a string variable "choice"

```
    # TODO: Add code to show the current data from the file to user
    @staticmethod
    def product_list(list_of_rows: list):
        """
        Print current products in the list of rows:
        :param list_of_rows: (list) of rows you want to display
        :return: nothing
        """
        print("******** Current Product List *************")
        for row in list_of_rows:
            print(row.product_name + " (" + str(row.product_price) + ")")
        print("*******************************************")
        print()  # Add an extra line for looks
```

The product_list method executes menu option one to show current data from the list table list_of_rows.

```
    # TODO: Add code to get product data from user
    @staticmethod
    def input_data():
        """
        Get user input data
        :return: Product object with input data
        """
        try:
            product = str(input("Enter product name? - ").strip())
            price = float(input("Enter product price? - ").strip())
            print()  # Add an extra line for looks
            p = Product(product_name=product, product_price=price)
        except Exception as e:
            print(e)
        return p
# Presentation (Input/Output)  -------------------------------------------- #
```

The input_data method executes menu option two to add data. It uses try/except block that contains two input() statements for the user to enter a product name and price. I used the strip() method to remove whitespace and characters. The "p" variable gets the values from the Product object attributes then returns that data at the end of the method. The exception block catches all general errors.

```
# Main Body of Script  ---------------------------------------------------- #
# TODO: Add Data Code to the Main body

# TODO: Load data from file into a list of product objects when script starts
lstOfProductObjects = FileProcessor.read_data_from_file(strFileName)
```

The first step in the main body sets the list variable "lstOfProductObjects" by calling the FileProcessor class method "read_data_from_file" to load data from the Products.txt. file if it exists.

```
while (True):
    # TODO: Show user a menu of options
    IO.menu()  # Shows menu
```

Executes the IO class menu() method to print a display of menu options.

```
    # TODO: Get user's menu option choice
    strChoice = IO.choice()
```

Executes the IO class method choice() to prompt the user to input a menu option choice.

```
    # TODO: Show user current data in the list of product objects
    if strChoice.strip() == '1':
        IO.product_list(lstOfProductObjects)  # Show current data in the list/table
        continue
```

Executes menu option one that calls the IO class product_list method to list current product data from the list table of rows.

```
  # TODO: Let user add data to the list of product objects
  if strChoice.strip() == '2':
    lstOfProductObjects.append(IO.input_data())
   continue
```

Executes menu option two that calls the IO class input_data() method to prompt the user to input a product name and price and then appends that data to the list table of rows.

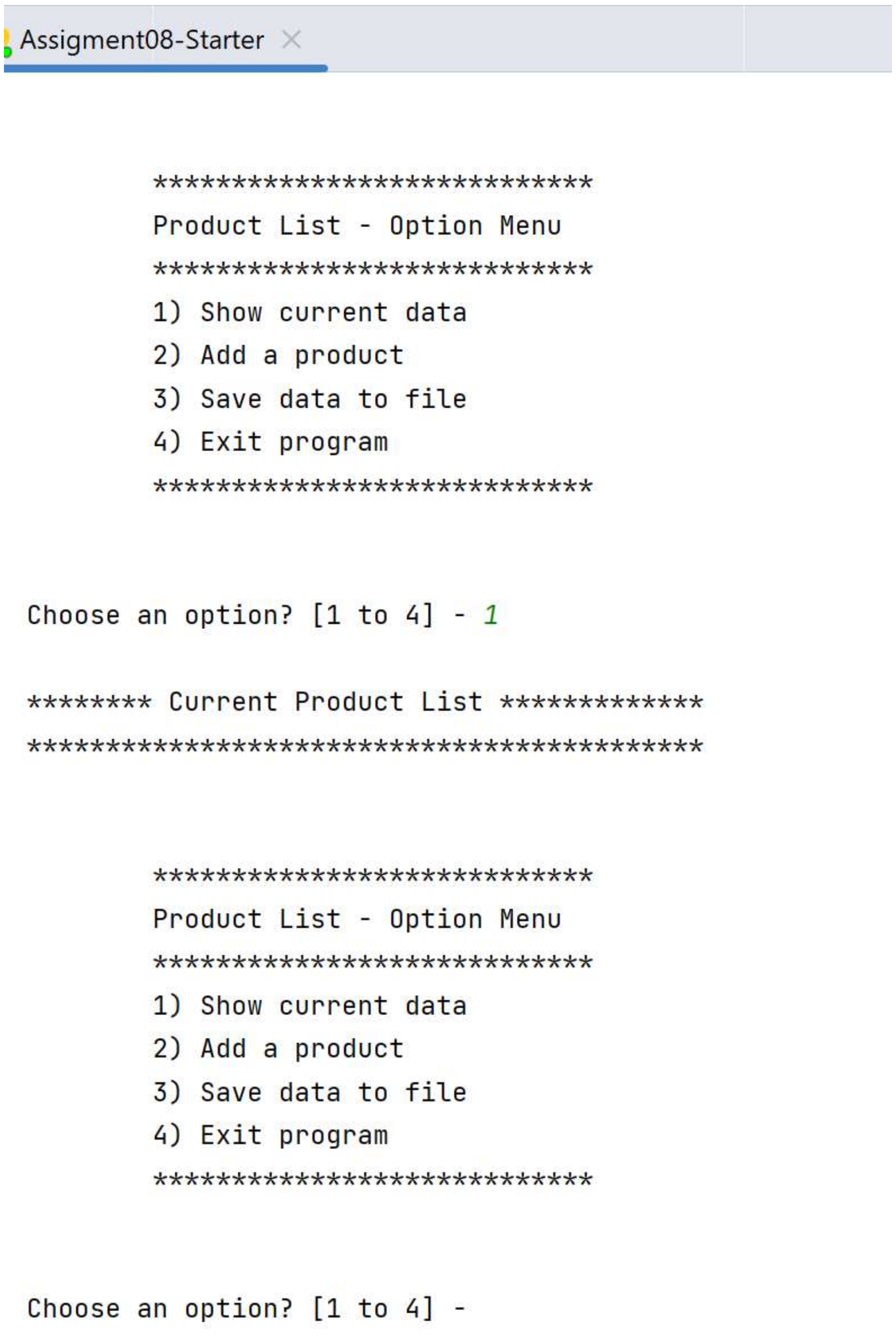
```
  # TODO: Let user save current data to file
  elif strChoice == '3':
    FileProcessor.save_data_to_file(strFileName, lstOfProductObjects)
    continue
```

Executes menu option three that calls the FileProcessor class method "save_data_to_file" and passes in the strFileName and listOfProductObjects parameters to save the product data to the Products.txt delimited text file.

```
  # TODO: Exit program
  elif strChoice == '4':
    break

# Exit the program
input("\nPress the enter key to exit.")
```

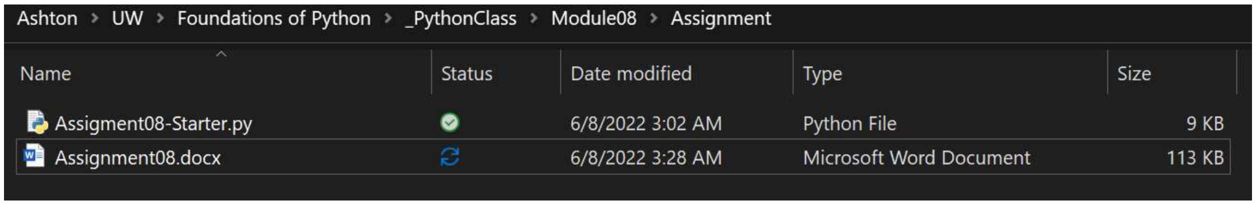
Executes menu option four that breaks the loop and prompts the user with an input() function asking the user to press the "enter" key to exit and close the program.

Run the script from PyCharm.



Assigment08-Starter

```
            ****************************
            Product List - Option Menu
            ****************************
            1) Show current data
            2) Add a product
            3) Save data to file
            4) Exit program
            ****************************


 Choose an option? [1 to 4] - 1

 ******** Current Product List *************
 *******************************************


            ****************************
            Product List - Option Menu
            ****************************
            1) Show current data
            2) Add a product
            3) Save data to file
            4) Exit program
            ****************************


 Choose an option? [1 to 4] -
```



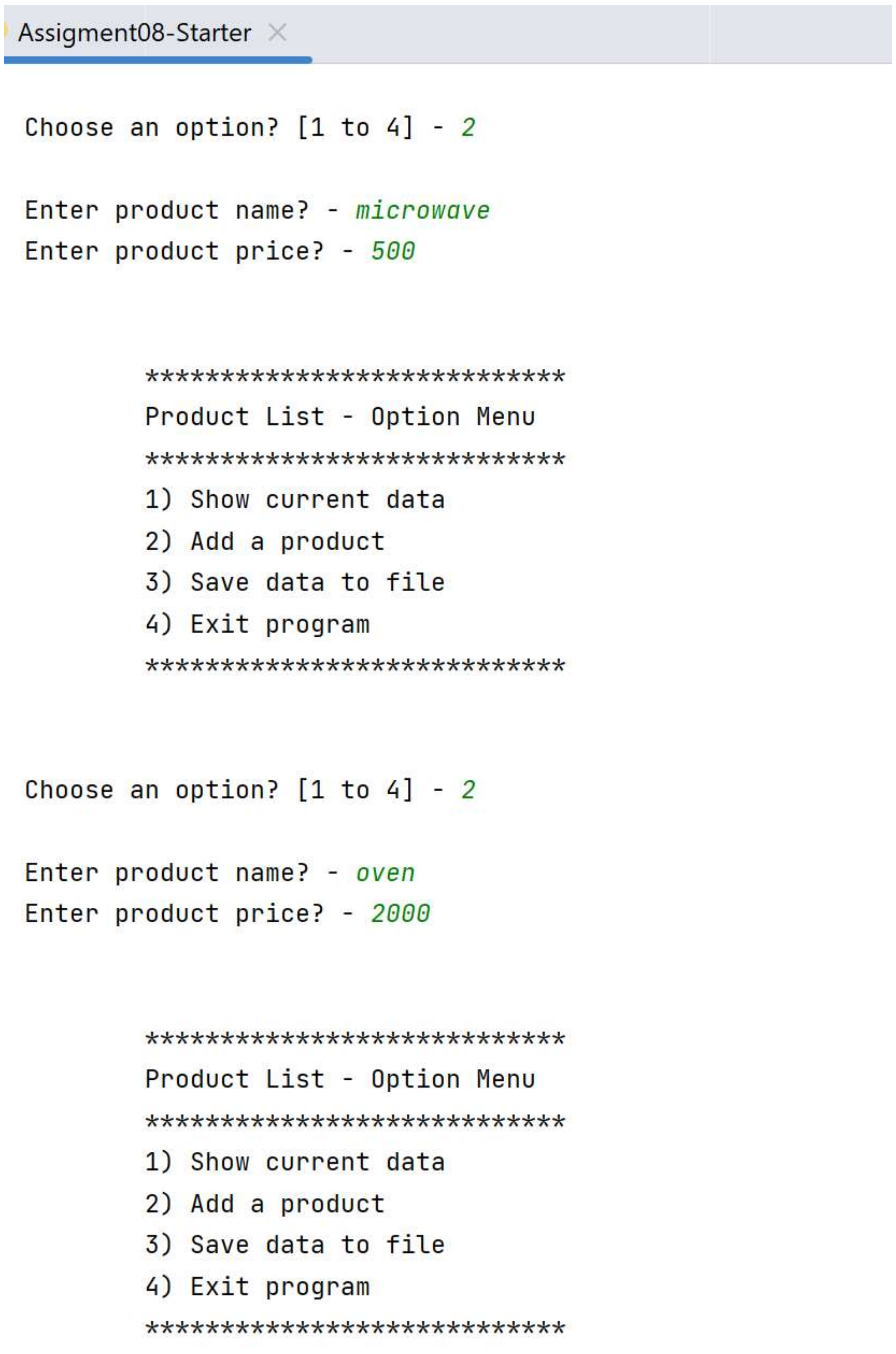
Ashton › UW › Foundations of Python › _PythonClass › Module08 › Assignment

Name	Status	Date modified	Type	Size
Assigment08-Starter.py	✓	6/8/2022 3:02 AM	Python File	9 KB
Assignment08.docx	⟳	6/8/2022 3:28 AM	Microsoft Word Document	113 KB

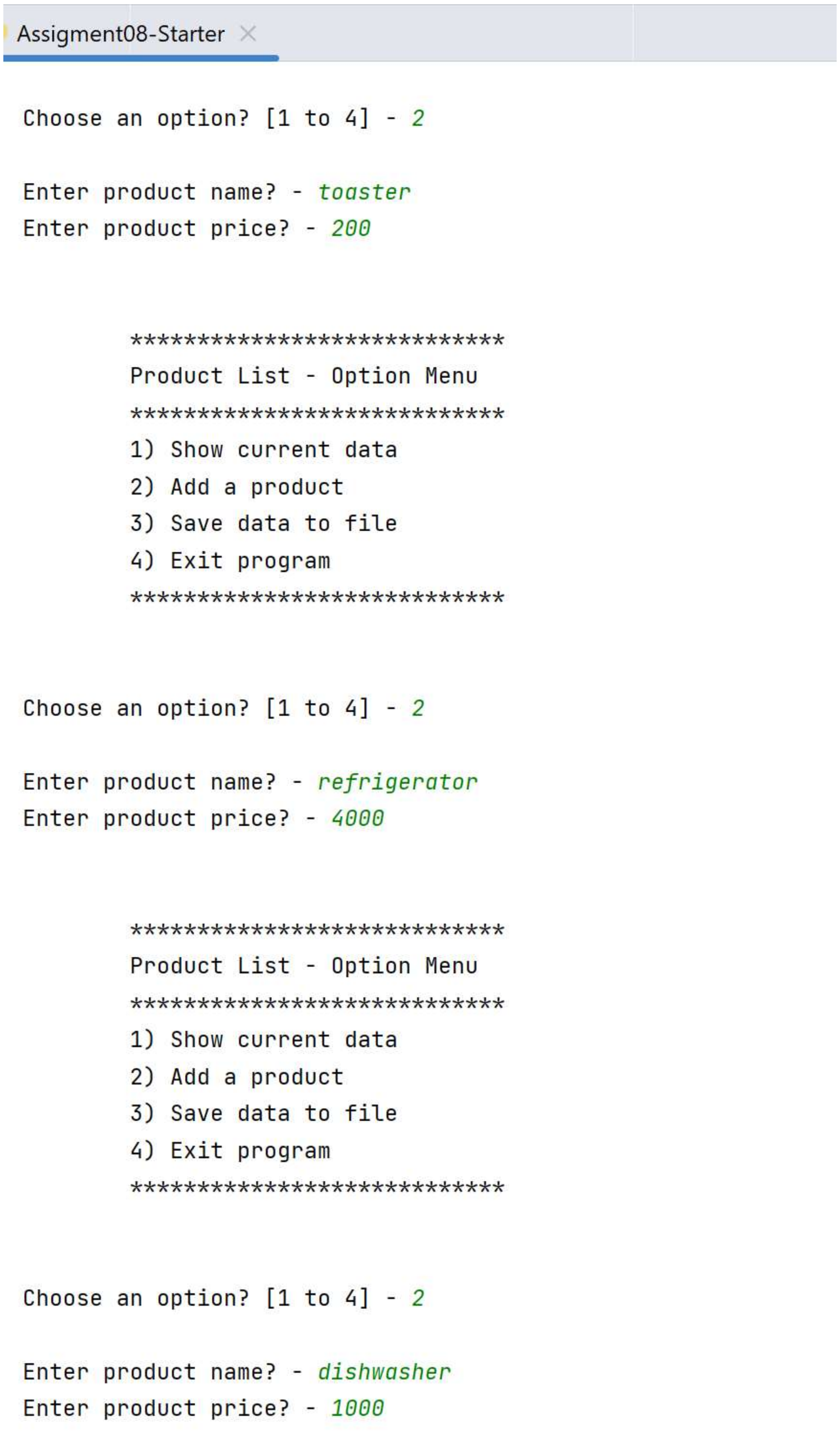
Products.txt file does not exist on initial program execution.

Execute option two and add the following test data:

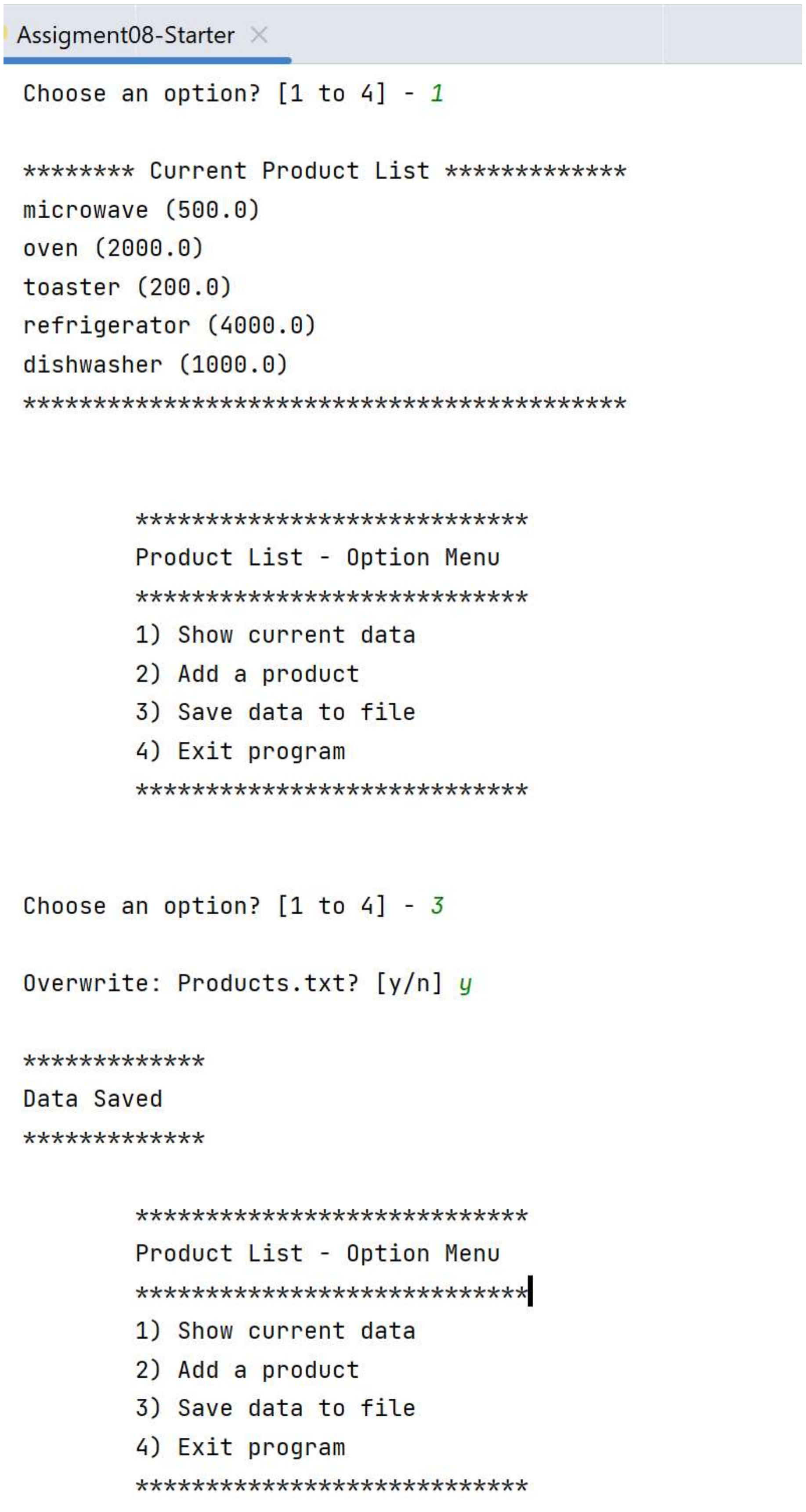
# Products
# test data: kitchen appliances
# products = ['microwave', 'oven', 'toaster', 'refrigerator', 'dishwasher']
# prices = [500, 2000, 200, 4000, 1000]



```
Choose an option? [1 to 4] - 2

Enter product name? - microwave
Enter product price? - 500


        ****************************
        Product List - Option Menu
        ****************************
        1) Show current data
        2) Add a product
        3) Save data to file
        4) Exit program
        ****************************


Choose an option? [1 to 4] - 2

Enter product name? - oven
Enter product price? - 2000


        ****************************
        Product List - Option Menu
        ****************************
        1) Show current data
        2) Add a product
        3) Save data to file
        4) Exit program
        ****************************
```

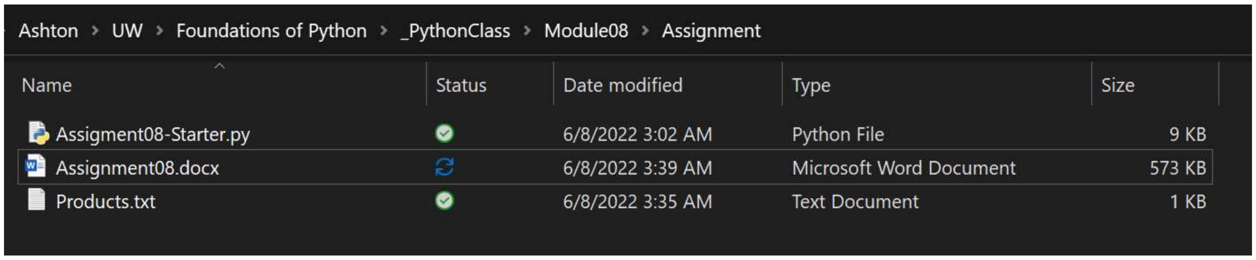

Assigment08-Starter

```
Choose an option? [1 to 4] - 2

Enter product name? - toaster
Enter product price? - 200


        ****************************
        Product List - Option Menu
        ****************************
        1) Show current data
        2) Add a product
        3) Save data to file
        4) Exit program
        ****************************


Choose an option? [1 to 4] - 2

Enter product name? - refrigerator
Enter product price? - 4000


        ****************************
        Product List - Option Menu
        ****************************
        1) Show current data
        2) Add a product
        3) Save data to file
        4) Exit program
        ****************************


Choose an option? [1 to 4] - 2

Enter product name? - dishwasher
Enter product price? - 1000
```



```
Choose an option? [1 to 4] - 1

******** Current Product List *************
microwave (500.0)
oven (2000.0)
toaster (200.0)
refrigerator (4000.0)
dishwasher (1000.0)
*******************************************



        ****************************
        Product List - Option Menu
        ****************************
        1) Show current data
        2) Add a product
        3) Save data to file
        4) Exit program
        ****************************


Choose an option? [1 to 4] - 3

Overwrite: Products.txt? [y/n] y

*************
Data Saved
*************

        ****************************
        Product List - Option Menu
        ****************************
        1) Show current data
        2) Add a product
        3) Save data to file
        4) Exit program
        ****************************
```

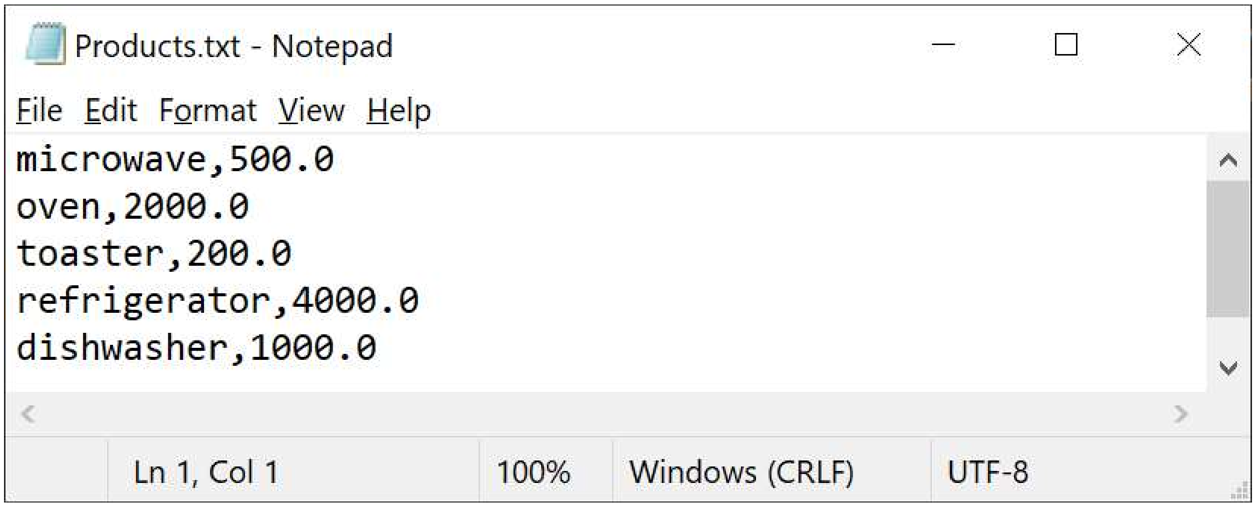
```
Choose an option? [1 to 4] - 1

******** Current Product List *************
microwave (500.0)
oven (2000.0)
toaster (200.0)
refrigerator (4000.0)
dishwasher (1000.0)
*******************************************


        ****************************
        Product List - Option Menu
        ****************************
        1) Show current data
        2) Add a product
        3) Save data to file
        4) Exit program
        ****************************


Choose an option? [1 to 4] - 4


Press the enter key to exit.

Process finished with exit code 0
```



Ashton › UW › Foundations of Python › _PythonClass › Module08 › Assignment

Name	Status	Date modified	Type	Size
Assigment08-Starter.py	✓	6/8/2022 3:02 AM	Python File	9 KB
Assignment08.docx	⟳	6/8/2022 3:39 AM	Microsoft Word Document	573 KB
Products.txt	✓	6/8/2022 3:35 AM	Text Document	1 KB

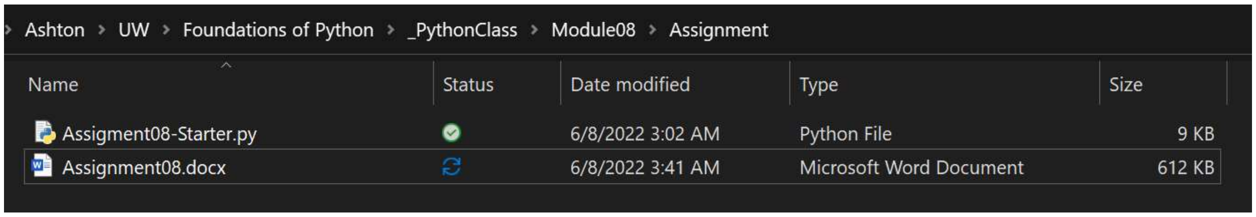
Products.txt file exists after program execution.



```
microwave,500.0
oven,2000.0
toaster,200.0
refrigerator,4000.0
dishwasher,1000.0
```

Products.txt data validation.

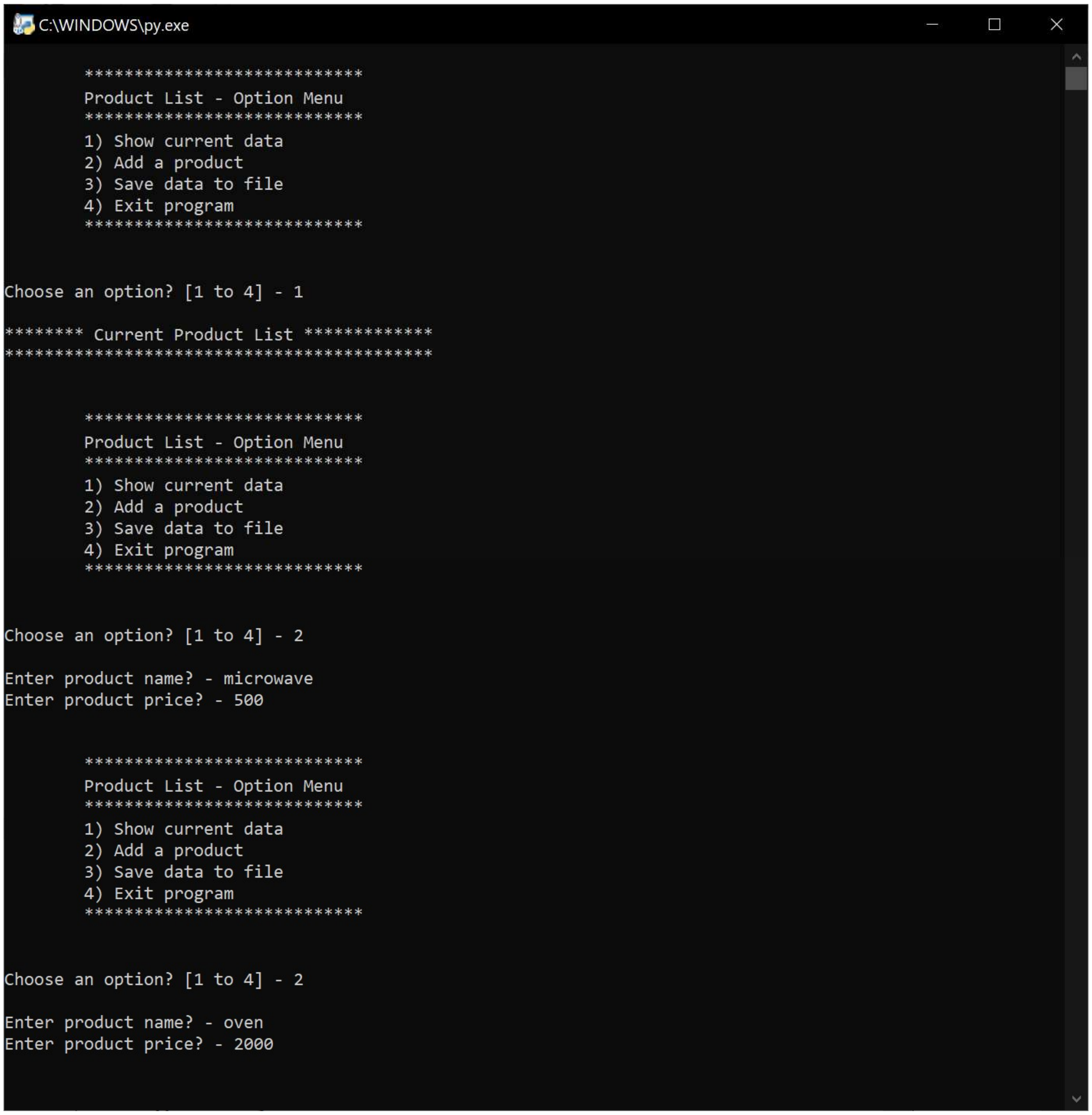
Run the Python Script from the Windows OS Command Shell and verify the data in the ToDoList.txt file.



Ashton > UW > Foundations of Python > _PythonClass > Module08 > Assignment

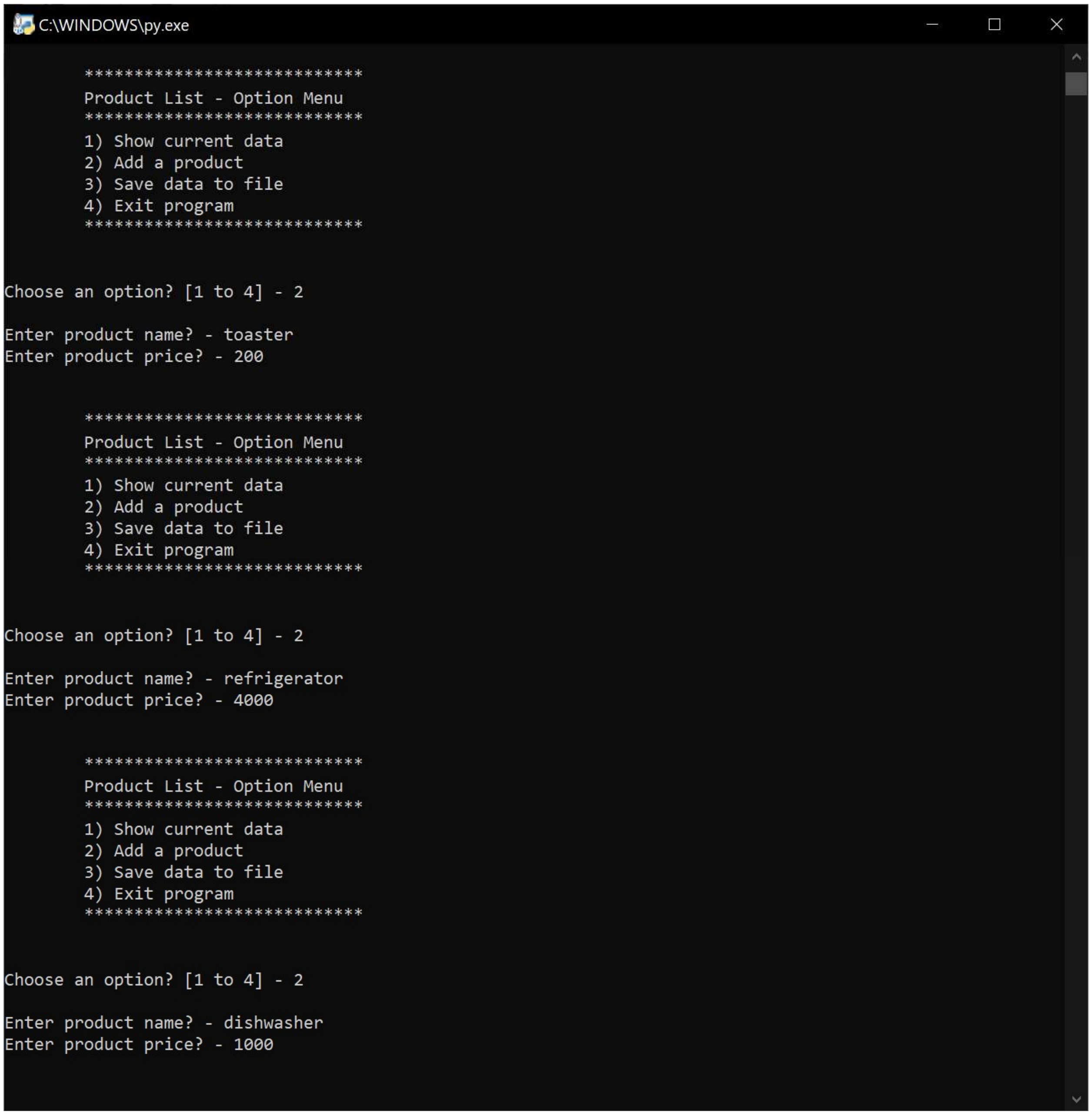
Name	Status	Date modified	Type	Size
Assigment08-Starter.py	✓	6/8/2022 3:02 AM	Python File	9 KB
Assignment08.docx	⟳	6/8/2022 3:41 AM	Microsoft Word Document	612 KB

Products.txt file does not exist on initial program execution.

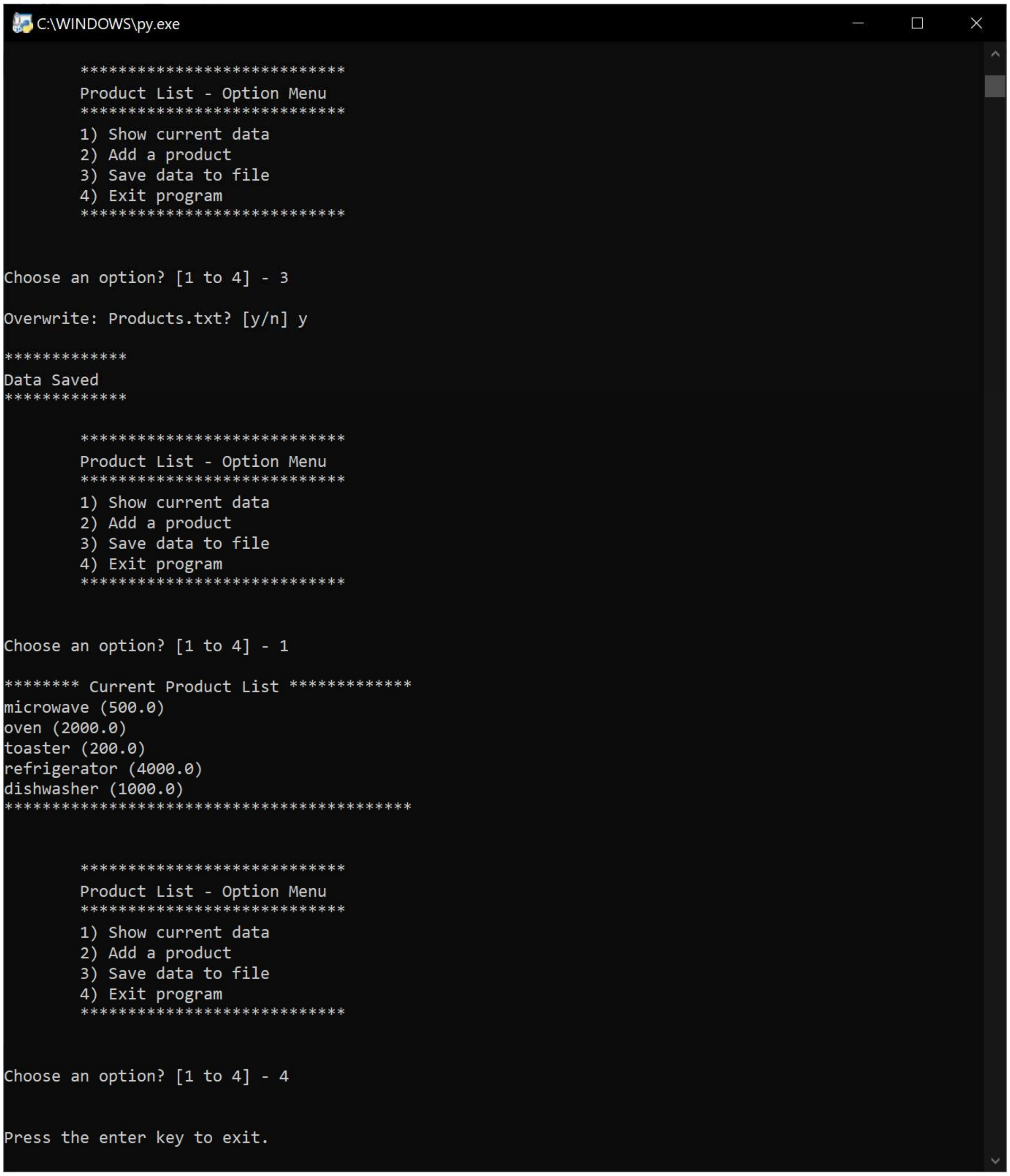


C:\WINDOWS\py.exe

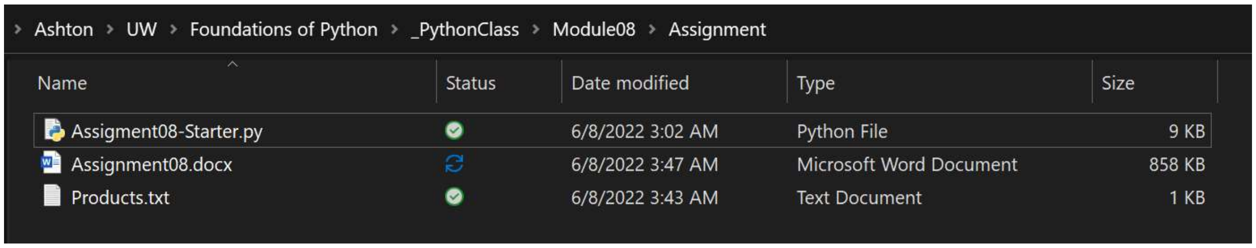
```
        ****************************
        Product List - Option Menu
        ****************************
        1) Show current data
        2) Add a product
        3) Save data to file
        4) Exit program
        ****************************


Choose an option? [1 to 4] - 1

******** Current Product List ************
*******************************************


        ****************************
        Product List - Option Menu
        ****************************
        1) Show current data
        2) Add a product
        3) Save data to file
        4) Exit program
        ****************************


Choose an option? [1 to 4] - 2

Enter product name? - microwave
Enter product price? - 500


        ****************************
        Product List - Option Menu
        ****************************
        1) Show current data
        2) Add a product
        3) Save data to file
        4) Exit program
        ****************************


Choose an option? [1 to 4] - 2

Enter product name? - oven
Enter product price? - 2000
```



```
        ****************************
        Product List - Option Menu
        ****************************
        1) Show current data
        2) Add a product
        3) Save data to file
        4) Exit program
        ****************************


Choose an option? [1 to 4] - 2

Enter product name? - toaster
Enter product price? - 200


        ****************************
        Product List - Option Menu
        ****************************
        1) Show current data
        2) Add a product
        3) Save data to file
        4) Exit program
        ****************************


Choose an option? [1 to 4] - 2

Enter product name? - refrigerator
Enter product price? - 4000


        ****************************
        Product List - Option Menu
        ****************************
        1) Show current data
        2) Add a product
        3) Save data to file
        4) Exit program
        ****************************


Choose an option? [1 to 4] - 2

Enter product name? - dishwasher
Enter product price? - 1000
```

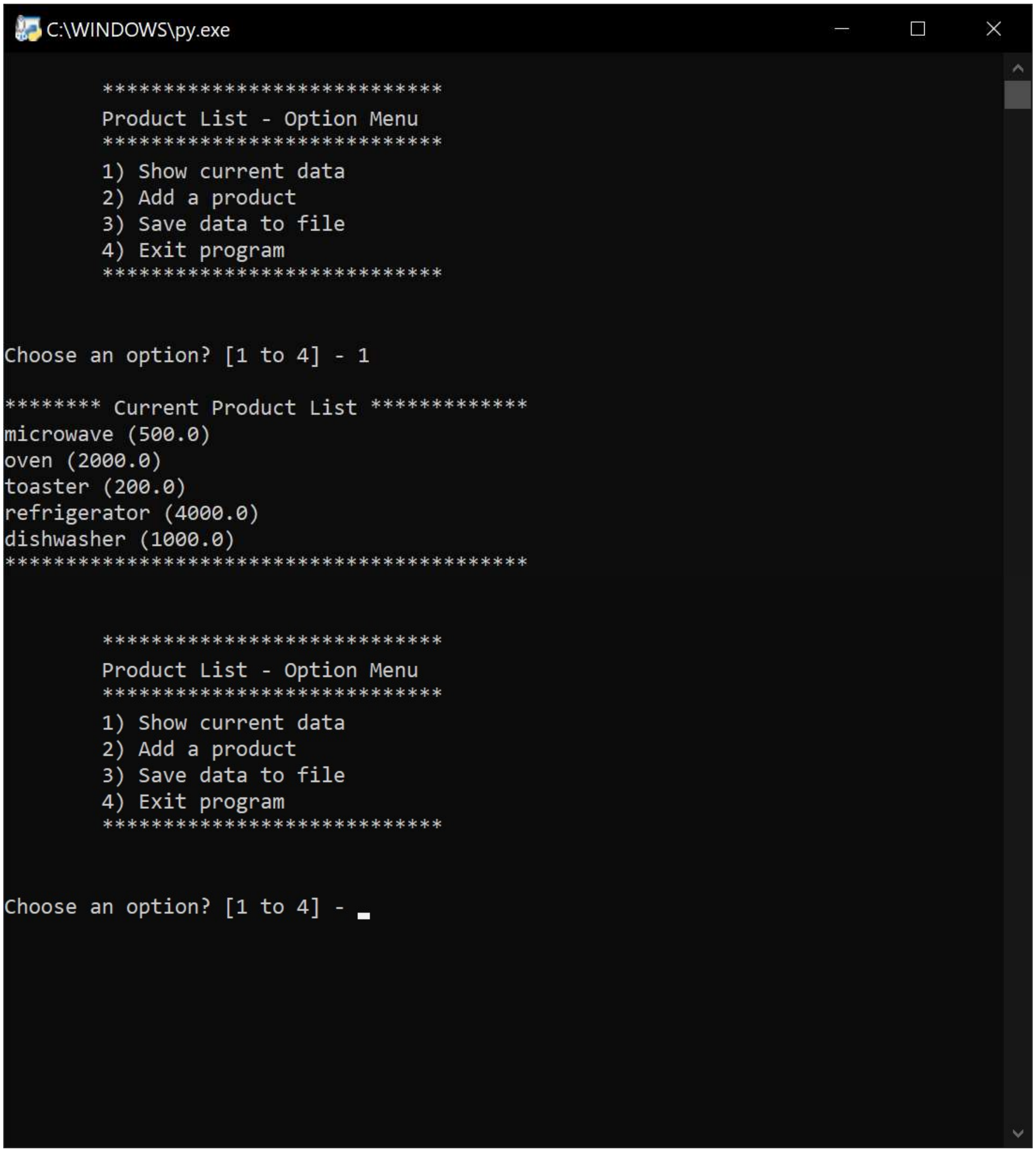


```
        ****************************
        Product List - Option Menu
        ****************************
        1) Show current data
        2) Add a product
        3) Save data to file
        4) Exit program
        ****************************


Choose an option? [1 to 4] - 3

Overwrite: Products.txt? [y/n] y

*************
Data Saved
*************

        ****************************
        Product List - Option Menu
        ****************************
        1) Show current data
        2) Add a product
        3) Save data to file
        4) Exit program
        ****************************


Choose an option? [1 to 4] - 1

******** Current Product List *************
microwave (500.0)
oven (2000.0)
toaster (200.0)
refrigerator (4000.0)
dishwasher (1000.0)
*******************************************


        ****************************
        Product List - Option Menu
        ****************************
        1) Show current data
        2) Add a product
        3) Save data to file
        4) Exit program
        ****************************


Choose an option? [1 to 4] - 4


Press the enter key to exit.
```


Ashton › UW › Foundations of Python › _PythonClass › Module08 › Assignment

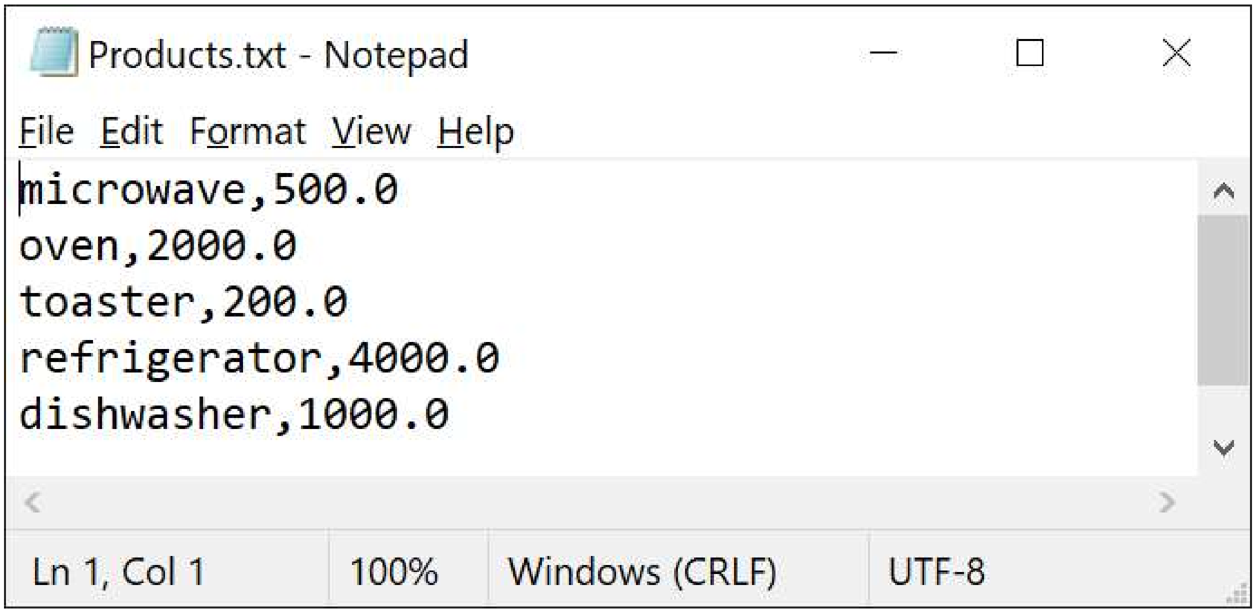
Name	Status	Date modified	Type	Size
Assigment08-Starter.py	✓	6/8/2022 3:02 AM	Python File	9 KB
Assignment08.docx		6/8/2022 3:47 AM	Microsoft Word Document	858 KB
Products.txt	✓	6/8/2022 3:43 AM	Text Document	1 KB

Products.txt file exists after program execution.


C:\WINDOWS\py.exe

```
        ****************************
        Product List - Option Menu
        ****************************
        1) Show current data
        2) Add a product
        3) Save data to file
        4) Exit program
        ****************************


Choose an option? [1 to 4] - 1

******** Current Product List *************
microwave (500.0)
oven (2000.0)
toaster (200.0)
refrigerator (4000.0)
dishwasher (1000.0)
*******************************************


        ****************************
        Product List - Option Menu
        ****************************
        1) Show current data
        2) Add a product
        3) Save data to file
        4) Exit program
        ****************************


Choose an option? [1 to 4] -
```

Reload the program and run option one to verify that the current data loads when the script starts.



```
microwave,500.0
oven,2000.0
toaster,200.0
refrigerator,4000.0
dishwasher,1000.0
```

Products.txt data validation.

Summary:

Module 08 introduced me to the methodology of Object-oriented programming (OOP) and its basic building block, the software object. In this module, I learned about creating classes (blueprints of objects) to define objects, writing methods, and creating attributes for objects, instantiating objects from classes and encapsulation for restricting access to an object's attributes. Assignment 08 introduced me to a different way of programming using software objects to combine functions and data. To complete the assignment 08 starter program, I added code to each step in the program at the "TODO" comments. I created multiple methods across the product, file processor, and IO classes in addition to the main body of the script that execute inputs/outputs that prompt the user to enter product and price data, appends that data to a list table of rows, and reads the data from and writes the data to a delimited text file.