Name: Marita Quiroz
Date: June 21, 2023
Course: IT FDN 110 A Sp 23: Foundations Of Programming: Python
Assignment: 08
Github URL: https://github.com/mquiroz1/IntroToProg-Python-Mod08

Assignment 08 – INTRO TO PROGRAMMING (PYTHON)

Introduction

The purpose of this assignment is to go through the assignment starter file, understand the pseudo-code, and add the appropriate code to allow the application to work. The program uses methods and classes to take user input (products and prices) and saves it to a file.

Program

The header section (Figure 1) provides information about the title, developer name, date of origination, and change log. The change log has been updated to include my name, date, and basic information regarding what was modified or updated.

_Assigment08.py products.txt

```
# ------------------------------------------------------------------------------ #
# Title: Assignment 08 - Final
# 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
# Marita Quiroz,06.20.2023,Modified code to complete assignment 8
# ------------------------------------------------------------------------------ #

```

Figure 1: Program header

Figure 2 shows the first section proceeding the header, the data section. Here, the file name and list variables are initially presented.

```python

# Data ------------------------------------------------------------------- #
strFileName = 'products.txt'
lstOfProductObjects = []

```

Figure 2: Data section with variable declaration

The Product class is presented in this Data section. The Product class begins with a constructor (Figure 3). The constructor is the initialization method, which will be called whenever the object is initially created. When the Product class takes the product name and product price in as input, the variables with values are initialized as object attributes.

```python
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:
    changelog: (When,Who,What)
        RRoot,1.1.2030,Created Class
        Marita Quiroz,06.20.2023,Modified code to complete assignment 8
    """

    # TODO: Add Code for Product class (Constructor, Properties, & Methods)
    # -- Constructor --
    def __init__(self, product_name, product_price):
        # Use a property to set the attribute
        self.product_name = product_name
        self.product_price = product_price

```

Figure 3: Constructor

Figure 4 shows the property attributes used within the Product class. Property allows the data to be indirectly accessed, while imposing restrictions on the client's ability to change the values of the attributes. The property attributes are created for the product names and the product prices.

```
    # -- Properties --
    3 usages (1 dynamic)
    @property  # You don't use for the getter's directive!
    def product_name(self):  # (getter or accessor)
        return str(self.__product_name)

    2 usages (1 dynamic)
    @product_name.setter  # The @NAME.setter must match the getter's name!
    def product_name(self, value):  # (setter or mutator)
        if str(value).isnumeric() == False:
            self.__product_name = value
        else:
            raise Exception("Names cannot be numbers")

    3 usages (1 dynamic)
    @property  # You don't use for the getter's directive!
    def product_price(self):  # (getter or accessor)
        return (self.__product_price)  # Format attribute as float

    2 usages (1 dynamic)
    @product_price.setter  # The @NAME.setter must match the getter's name!
    def product_price(self, value):  # (setter or mutator)
        self.__product_price = value
        if str(value).isnumeric() == True:
            self.__product_price = value
        else:
            raise Exception("Price must be a number")
```

Figure 4: Property attributes

Within each property attribute setter, the exception handling takes place. The product name input value will go through a Boolean check to see if value is a number. If the value is not a number, the attribute will be set. If the value is a number, an exception will be raised with the message, “Names cannot be numbers”. The product price input value will go through a similar Boolean check, but if the value checks out as a number, it will set the attribute. If the value is not a number, an exception will be raised with the message, “Price must be a number”.

The last part of the Data section defines a to_string method (Figure 5). The method will take the product name and price values and return an object, formatted to be in csv format. This format has a comma separator with the product name and price. This will be used when saving the data to file.

```
    1 usage (1 dynamic)
    def to_string(self):
        """  Returns object data in a comma separated string of values
        :return: (string) CSV data
        """
        object_data_csv = self.product_name + ',' + self.product_price
        return object_data_csv

# Data ------------------------------------------------------------------- #
```

Figure 5: to_string method

The next section of the program is the Processing section, where the FileProcessor class is defined (Figure 6).

```
# Processing  ------------------------------------------------------------ #
2 usages
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
        Marita Quiroz,06.20.2023,Modified code to complete assignment 8
    """
```

Figure 6: FileProcessor class

This section is where the defined methods will manipulate the file, either saving data to the file or reading data from the file. Figure 7 shows the method which saves the data to the file. The file is opened in write mode, the data is saved line by line from the list, and the file is closed. If the file is unable to be opened, or the data is unable to be saved, an exception is raised with a general error message.

```
1 usage
@staticmethod
def save_data_to_file(file_name, list_of_product_objects):
    """ Write data to a file from a list of object rows
            :param file_name: (string) with name of file
            :param list_of_product_objects: (list) of product objects data saved to file
            :return: (bool) with status of success status
    """
    success_status = False
    try:
        file = open(file_name, "w")
        for row in list_of_product_objects:
            file.write(row.to_string() + "\n")
        file.close()
        success_status = True
    except Exception as e:
        print("There was a general error!")
        print(e, e.__doc__, type(e), sep='\n')
    return success_status
```

Figure 7: save_data_to_file method

Figure 8 shows the read_data_to_file method. The file is opened in read mode, and each line is added into a row within a list variable. The file is closed once this has been completed. A similar exception will be raised if anything goes wrong within the reading of data from the file.

```
# TODO: Add Code to process data to a file
1 usage
@staticmethod
def read_data_from_file(file_name):  # (a list of product objects)
    """ Reads data from a file into a list of object rows
            :param file_name: (string) with name of file
            :return: (list) of object rows
    """
    list_of_rows = []
    try:
        file = open(file_name, "r")
        for line in file:
            row = line.split(",")
            list_of_rows.append(row)
        file.close()
    except Exception as e:
        print("There was a general error!")
        print(e, e.__doc__, type(e), sep='\n')
    return list_of_rows
```

Figure 8: read_data_from_file method

The IO class is defined within the Presentation section of the program (Figure 9). This class includes methods which interact directly with the user and acts as a liaison between the inner workings of the program and the user.

```
# Presentation (Input/Output)  ------------------------------------------- #
4 usages
class IO:
    """  A class for performing Input and Output
    methods:
        print_menu_items():
        print_current_list_items(list_of_rows):
        input_product_data():
    changelog: (When,Who,What)
        RRoot,1.1.2030,Created Class:
        Marita Quiroz, 06.20.2023, Modified code to complete Assignment 8
    """
    # Add code to show menu to user (Done for you as an example)
```

Figure 9: IO class

The first method displays the menu of choices to the user (Figure 10).

```
    1 usage
    @staticmethod
    def print_menu_items():
        """  Display a menu of choices to the user
            :return: nothing
        """
        print('''
        Menu of Options
        1) Show current data
        2) Add a new item.
        3) Save Data to File
        4) Exit Program
        ''')
        print()  # Add an extra line for looks in the terminal window

```

Figure 10: Menu method

The next two methods are shown in Figure 11.The first method takes the user's menu choice as input and saves the value in the choice variable. The next method prints the current list of items, which will go through the list and print each row.

```python
    @staticmethod
    def input_menu_options():
        choice = str(input("Which option would you like to perform? [1 to 4] - ")).strip()
        print()  # Add an extra line for looks
        return choice

    # TODO: Add code to show the current data from the file to user
    1 usage
    @staticmethod
    def print_current_list_items(list_of_rows):
        """ Print the current items in the list of Product rows
                :param list_of_rows: (list) of rows you want to display
        """
        print("******* The current products are: *******")
        for row in list_of_rows:
            print(str(row.product_name)
                  + ","
                  + row.product_price)
        print("*****************************************")
        print()  # Add an extra line for looks
```

Figure 11: Menu input and print current items methods

The last method takes the user input for product name and price, and save them in the appropriate variables (Figure 12).

```python

    # TODO: Add code to get product data from user
    1 usage
    @staticmethod
    def input_product_data():
        """ Gets data for a product object
                :return: (product) object with input data
        """
        try:
            product_name = str(input("What is the product name? - ").strip())
            product_price = str(input("What is the product price? - ").strip())
            print()  # Add an extra line for looks
            prod = Product(product_name, product_price)
        except Exception as e:
            print(e)
        return prod

# Presentation (Input/Output)  ------------------------------------------ #
```

Figure 12: input_product_data method

The main body of the program is next (Figure 13). This is where all of the previous sections and methods come together to enable to program to work and interact with the user. A while loop is used, to be able to continue the user interaction with the menu options until the user chooses to exit the program.

As the user makes choices, the corresponding method is called from the appropriate class. For example, if the user chooses option 1, the print_current_list_items method is called from the IO class. Continue is then called so the menu will be presented to the user, to enable them to make their next choice. I have added an extra piece at the end of the while loop for if the user does not enter a choice between 1 and 4. If the user chooses something else, a message will be displayed to remind the user to make a choice within the range, then continue and display the menu to provide a chance to make a valid choice.

```python
while(True):
    # Show user a menu of options
    IO.print_menu_items()
    menuchoice = int(IO.input_menu_options())
    # Get user's menu option choice
    if menuchoice == 1:
        # Show user current data in the list of product objects
        IO.print_current_list_items(lstOfProductObjects)
        continue # to show menu
    elif menuchoice == 2:
        # Let user add data to the list of product objects
        lstOfProductObjects.append(IO.input_product_data()) # Append new data to end of list
        continue # to show menu
    elif menuchoice == 3:
        # let user save current data to file
        FileProcessor.save_data_to_file("products.txt", lstOfProductObjects)
        print("Data saved")
        continue # to show menu
    elif menuchoice == 4:
        print("Exiting program")
        break # exit loop
    else:
        print("Please enter a valid choice [1-4]")
        continue # to show menu
exit()
```

Figure 13: Main body

There is a bonus section I have left in, but commented out, to show what I had done to test the various methods within the program (Figure 14). I used this to test each of the methods before implementing them within the main body of the script, to eliminate complication for troubleshooting.

```python
# Testing  ---------------------------------------------------- #

"""
# Test Product class
objP1 = Product("hammer", "10")
objP2 = Product("nail", "1")
lstOfProductObjects = [objP1, objP2]
for row in lstOfProductObjects:
    print(row.to_string(), type(row))

# Test File Processor class
FileProcessor.save_data_to_file("products.txt", lstOfProductObjects)
lstFileData = FileProcessor.read_data_from_file("products.txt")
for row in lstFileData:
    p = Product(row[0], row[1])
    print(p.to_string().strip(), type(p))


# Test IO class
IO.print_menu_items()
IO.print_current_list_items(lstOfProductObjects)
print(IO.input_product_data())
print(IO.input_menu_options())
"""

# Testing  ---------------------------------------------------- #
```

Figure 14: Testing section

Challenges

One of the challenges I faced while adding to the existing program was formatting the output to the screen. As I was testing the output display of the list of products, the type was being displayed (Figure 15 and 16), from a piece of code I had taken from a previous lab/ lecture.

```
# Main Body of Script  ----------------------------

objP1 = Product("hammer", "10")
objP2 = Product("nail", "1")
lstOfProductObjects = [objP1, objP2]
for row in lstOfProductObjects:
    print(row.to_string(), type(row))
```

Figure 15: Testing output of list

```
/Users/maritaquiroz/Documents/_PythonClass/Assig
Hammer,10 <class '__main__.Product'>
Nail,1 <class '__main__.Product'>

Process finished with exit code 0
```

Figure 16: Output of list while testing

Program in PyCharm

The following series of Figures (17-21) show the output of running the program in PyCharm.

```
/Users/maritaquiroz/Documents/_PythonClass/Assignment08/ve

        Menu of Options
        1) Show current data
        2) Add a new item.
        3) Save Data to File
        4) Exit Program
```

Figure 17: Initial menu display

```
Which option would you like to perform? [1 to 4] - 2

What is the product name? - hammer
What is the product price? - 10


        Menu of Options
        1) Show current data
        2) Add a new item.
        3) Save Data to File
        4) Exit Program


Which option would you like to perform? [1 to 4] - 2

What is the product name? - drill
What is the product price? - 200


        Menu of Options
        1) Show current data
        2) Add a new item.
        3) Save Data to File
        4) Exit Program
```

Figure 18: Adding new items

```
Which option would you like to perform? [1 to 4] - 1

******* The current products are: *******
hammer,10
drill,200
*****************************************


        Menu of Options
        1) Show current data
        2) Add a new item.
        3) Save Data to File
        4) Exit Program
```

Figure 19: Displaying items

```
Which option would you like to perform? [1 to 4] - 3

Data saved

        Menu of Options
        1) Show current data
        2) Add a new item.
        3) Save Data to File
        4) Exit Program


Which option would you like to perform? [1 to 4] - 4

Exiting program

Process finished with exit code 0
```

Figure 20: Saving items to file and exiting program

```
_Assigment08.py        products.txt
1  hammer,10
2  drill,200
3
```

Figure 21: File contents

Program in the Console

The following series of Figures (22-26) show the output of running the program in the console.

```
(venv) Maritas-MacBook-Pro:Assignment08 maritaquiroz$ python3 _Assigment08.py

        Menu of Options
        1) Show current data
        2) Add a new item.
        3) Save Data to File
        4) Exit Program


Which option would you like to perform? [1 to 4] - 2

What is the product name? - screwdriver
What is the product price? - 10


        Menu of Options
        1) Show current data
        2) Add a new item.
        3) Save Data to File
        4) Exit Program


Which option would you like to perform? [1 to 4] - 2

What is the product name? - nails
What is the product price? - 20
```

Figure 22: Adding items

```
        Menu of Options
        1) Show current data
        2) Add a new item.
        3) Save Data to File
        4) Exit Program


Which option would you like to perform? [1 to 4] - 1

******* The current products are: *******
screwdriver,10
nails,20
*****************************************


        Menu of Options
        1) Show current data
        2) Add a new item.
        3) Save Data to File
        4) Exit Program


Which option would you like to perform? [1 to 4] - 3

Data saved

        Menu of Options
```

Figure 23: Displaying current items and saving to file

```
        Menu of Options
        1) Show current data
        2) Add a new item.
        3) Save Data to File
        4) Exit Program


Which option would you like to perform? [1 to 4] - 5

Please enter a valid choice [1-4]

        Menu of Options
        1) Show current data
        2) Add a new item.
        3) Save Data to File
        4) Exit Program


Which option would you like to perform? [1 to 4] - 4

Exiting program
(venv) Maritas-MacBook-Pro:Assignment08 maritaquiroz$
```

Figure 24: Entering a choice out of range and exiting program

Summary

This program was used to demonstrate my ability to read pseudo-code, understand the intent of the program, and complete the program from its unfinished state. The program put many concepts together from the class in its entirety and was good practice for real world applications with modifying or adding to someone's work. For this assignment, I consulted the

lecture, labs, the book, and the python documentation on the internet. I used the https://docs.python.org/ website to consult documentation. I also referenced previous labs from other modules.