Sanghee Kang
March 8, 2022
IT FND 110 Foundations of Programming: Python
Assignment 08
GitHub URL: https://github.com/bsb218218/IntroToPro-Python-Mod08

Creating a Script with Three Classes

1. Introduction

In this document, I will explain how I modified a Python script file by creating three classes (Product, FileProcessor, and IO). Also, I will describe my observation of the performance of the code in both PyCharm and the command shell.

2. Creating a Python Script with Three Classes

2.1. Creating a sub-folder

As usually, I created a folder (Assignment08_SangheeKang) inside of the "_PythonClass" folder.

2.2. Creating a new Project in PyCharm

I created a new Project in Pycharm (File-New Project). For the location, I chose "C:_PythonClass\Assignment06_SangheeKang" and selected "New environment using Virtualenv". In the new project, I added the starter Python file provided by the instructor and renamed it as "Assignment08_SangheeKang". In the script header, I added my name and the date to indicate that I modified the file.

2.3. Adding codes

2.3.1. Class Product

As the outline is already created by the instructor, I added necessary codes to the file. For Product class, which deals with product data, I created a constructor, properties for two variables (product_name, product_price), and a method (the __str__() method to return data as a string). For the constructor, the __init__() was used to set a product name and product price. To handle an expected error, the try-except function was used. For the properties, both getter (for formatting) and setter (for validation) were created for product_name and product_price. As product_name should be a string, for the getter (@property), product_name was defined as a string, and for the setter, the if-else functions were used to validate if product_name is a string or not (i.e., not numeric). In terms of product_price, for the getter, it was defined as a float number and for the setter, it was tested whether it is numeric or not. Lastly, the __str__() method was used to convert product data into a string.

```python
class Product:
    """Stores data about a product:

    properties:
        product_name: (string) with the products's  name
        product_price: (float) with the products's standard price
    methods:
    changelog: (When,Who,What)
        RRoot,1.1.2030,Created Class
        SKang,3.8.22,Modified code to complete assignment 8
    """
    # TODO: Add Code to the Product class
    # Constructor
    def __init__(self, product_name, product_price):
        """set product name and price of a new product """
        # Attributes
        try:
            self.product_name = str(product_name)
            self.product_name = float(product_price)
        except Exception as e:
            raise Exception("Error with setting values: \n" + str(e))

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

    @product_name.setter
    def product_name(self, value):
        if str(value).isnumeric() == False:
            self.__product_name = value
        else:
            raise Exception("Names cannot be numeric")
```

```python
    # Product price
    @property
    def product_price(self):
        return float(self.__product_price)

    @product_price.setter
    def product_price(self, value):
        if str(value).isnumeric():
            self.__product_price = float(value)
        else:
            raise Exception("Prices should be numeric")

    # Methods
    def __str__(self):
        """Convert product data to string"""
        return "[" + self.product_name + "," + str(self.product_price) + "]"
```

Figure 1. The codes for class Product

2.3.2. Class FileProcessor

The class FileProcessor was created to process data to and from a file and a list of product rows. In the class, I added methods such as save_data_to_file(file_name, list_of_product_rows), read_data_from_file(file_name), and add_data_to_list(name, price, list_of_product_rows) as shown in Figure 2. To call the functions from the class directly, I used @staticmethod. Also, as instructed, I added try-except functions to deal with errors. For saving data to the file, functions such as the open(),file.write(), file.close(), and the for loop were used. To read data from the file, functions like the for loop, split(), file.close() were used. Lastly, for adding new product information, I used the append() method.

```python
# TODO: Add Code to process data to a file
@staticmethod
def save_data_to_file(file_name, list_of_product_rows):
    """ Writes data from a list of product rows to a File

    :param file_name: (string) with name of file:
    :param list_of_product_rows: (list) you want filled with file data:
    :return: (list) of product rows
    """
    # TODO: Add Code Here!
    try:
        file = open(file_name, "w")
        for row in list_of_product_rows:
            file.write(str(row["Product"] + "," + str(row["Price"]))+ "\n")
        file.close()
    except Exception as e:
        print("There was an error!")
        print(e, e.__doc__, type(e), sep='\n')
    return list_of_product_rows
```

```python
# TODO: Add Code to process data from a file
@staticmethod
def read_data_from_file(file_name):
    """ Reads data from a file into a list of product rows

    :param file_name: (string) with name of file:
    :return: (list) of product rows
    """
    list_of_product_rows = []
    try:
        file = open(file_name, "r")
        for line in file:
            data = line.split(",")
            row = Product(data[0], data[1])
            list_of_product_rows.append(row)
        file.close()
    except Exception as e:
        print("There was an error")
        print(e, e.__doc__, type(e), sep='\n')
    return list_of_product_rows
```

```python
@staticmethod
def add_data_to_list(name, price, list_of_product_rows):
    """ Adds data to a list of product rows

    :param name: (string) with name of product:
    :param price: (string) with price of product:
    :param list_of_product_rows: (list) you want filled with file data:
    :return: (list) of product rows
    """
    row = {"Product": str(name).strip(), "Price": str(price).strip()}
    # TODO: Add Code Here!
    list_of_product_rows.append(row)
    return list_of_product_rows
```

Figure 2. The codes for class FileProcessor

2.3.3. Class IO

Lastly, for class IO, which was for presentation (i.e., input and output), I first added docstring to show what tasks the codes perform, methods that are used in the class, and the changelog (Figure 3). In this class, four functions were included: output_menu_options() (to show menu options to a user), input_menu_choice() (to get the menu choice from a user), output_current_items_in_list() (to show the current items to a user), and input_new_product_and_price() (to get product name and product price to be added to the list). As we have already created these functions in the previous assignment, I used the same functions but modified them so that they use the same variable names and also for the overall consistency.

```python
# Presentation (Input/Output)  -------------------------------------------- #
class IO:
    # TODO: Add docstring
    """ Performs Input and Output tasks

    methods:
        output_menu_options():
        input_menu_choice():
        output_current_items_in_list(list_of_product_rows):
        input_new_product_and_price():

    changelog: (When,Who,What)
        RRoot,1.1.2030,Created Class
        SKang,3.8.2022,Modified code to complete assignment 8"""
```

```python
    # each of the following will be functions
    # TODO: Add code to show menu to user
    @staticmethod
    def output_menu_options():
        """  Display a menu of choices to the user

        :return: nothing
        """
        print('''
            Menu of Options
            1) Show the current data
            2) Add a new item
            3) Save data to file
            4) Exit program
            ''')
        print()  # Add an extra line for looks
```

```python
# TODO: Add code to get user's choice
@staticmethod
def input_menu_choice():
    """ Gets the menu choice from a user

    :return: string
    """
    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
@staticmethod
def output_current_items_in_list(list_of_product_rows: list):
    """ Shows the current items in the list of product rows

    :param list_of_rows: (list) of rows you want to display
    :return: nothing
    """
    print("******* The current items are: *******")
    for row in list_of_product_rows:
        print(row["Task"] + " (" + row["Price"] + ")")
    print("*****************************************")
    print()  # Add an extra line for looks
```

```python
# TODO: Add code to get product data from user; product name and price
@staticmethod
def input_new_product_and_price():
    """  Gets product and price to be added to the list

    :return: (string, string) with product and price
    """
    # pass  # TODO: Add Code Here!
    try:
        print("Type in 'Product' and 'Price' for your product list")
        product_name = str(input(" Enter an Item: ")).strip()
        product_price = float(input(" Enter Price: ").strip())
        # print()
        productItem = Product(product_name=product_name, product_price=product_price)
    except Exception as e:
        print(e)
    return product_name, product_price
```

Figure 3. The codes for class IO

2.3.4. Main Body of Script

Based on the outline provided in the starter file, I only added codes to make sure the codes perform its work. But I spent quite some time to understand how it is organized, especially focusing on which functions should be added to which part. I added the try-except to address unexpected errors and used the while loop for showing a user a menu of options. There are four options (1) Show the current data; 2) Add a new item; 3) Save data to file; 4) Exit program). For each choice, functions from the three classes were used as shown in Figure 4.

```python
# Main Body of Script  ------------------------------------------------------ #
# Load data from file into a list of product objects when script starts
try:
    list_of_product_rows = FileProcessor.read_data_from_file(strFileName)  # read file data

# Show user a menu of options
    while (True):
        IO.output_menu_options()  # Shows menu
        choice_str = IO.input_menu_choice()  # Get menu option

# Get user's menu option choice
    # Choice 1: Show user current data in the list of product objects
    # Choice 2: Let user add data to the list of product objects
    # Choice 3: let user save current data to file and exit program
        if choice_str.strip() == '1':
            IO.output_current_items_in_list(list_of_product_rows)  # Show current data in the list/table
            continue

        elif choice_str.strip() == '2':  # Add a new Task
            product_name, product_price = IO.input_new_product_and_price()
            table_lst = FileProcessor.add_data_to_list(name=product_name, price=product_price, list_of_product_rows=table_lst)
            continue  # to show the menu

        elif choice_str == '3':  # Save Data to File
            table_lst = FileProcessor.save_data_to_file(file_name=strFileName, list_of_product_rows=table_lst)
            print("Data Saved!")
            continue  # to show the menu

        elif choice_str == '4':  # Exit Program
            print("Goodbye!")
            break  # by exiting loop
except Exception as e:
    print("There was an error!")
    print(e, e.__doc__, type(e), sep='\n')

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

Figure 4. The codes for main body of the script

2.4. Running the modified Python script and verifying that it worked

To run the modified Python script in PyCharm, I right-clicked and selected “Run Assignment08_SangheeKang.” I added a text file (products.txt), which does not have any item in it in the folder. I typed in four options one by one to check whether all of them perform its task successfully. The following figures (Figures 5,6 and 7) show the output of each choice. For choice 2, I added two tasks (lamp: 11.99; desk: 119.99) using the program. When I am done with running all functions in the code, I checked output and the text file saved in the same folder. The text file included lamp: 11.99 and desk: 119.99 (Figure 8). Therefore, my python code completed the task successfully.

```
          Menu of Options
          1) Show the 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 items are: *******
*****************************************


          Menu of Options
          1) Show the current data
          2) Add a new item
          3) Save data to file
          4) Exit program
```

Figure 5. Output of choice 1

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

Type in 'Product' and 'Price' for your product list
 Enter an Item: lamp
 Enter Price: 11.99

            Menu of Options
            1) Show the 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

Type in 'Product' and 'Price' for your product list
 Enter an Item: desk
 Enter Price: 119.99

            Menu of Options
            1) Show the current data
            2) Add a new item
            3) Save data to file
            4) Exit program
```

Figure 6. Output of choice 2

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

Data Saved!

            Menu of Options
            1) Show the 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

Goodbye!

Process finished with exit code 0
```

Figure 7. Output of choices 3 and 4

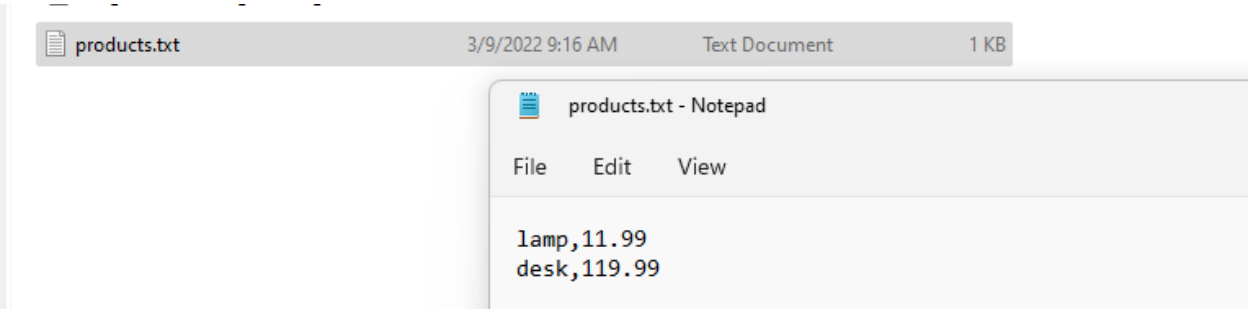

Figure 8. Content of the products text file

Then, I ran the Python code in a command shell. I followed the same steps as I did in the PyCharm and the same output was produced including the content of the text file. (see Figures 9 and 10).

```
Microsoft Windows [Version 10.0.22000.493]
(c) Microsoft Corporation. All rights reserved.

C:\Users\sangh>CD "C:\_PythonClass\Assignment08_SangheeKang"

C:\_PythonClass\Assignment08_SangheeKang>Python.exe Assignment08_SangheeKang.py

            Menu of Options
            1) Show the 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 items are: *******
*******************************************
```

```
           Menu of Options
           1) Show the 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

Type in 'Product' and 'Price' for your product list
 Enter an Item: lamp
 Enter Price: 11.99

           Menu of Options
           1) Show the 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

Type in 'Product' and 'Price' for your product list
 Enter an Item: desk
 Enter Price: 119.99

           Menu of Options
           1) Show the 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
           1) Show the 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

Goodbye!

C:\_PythonClass\Assignment08_SangheeKang>
```

Figure 9. Final output of the modified Python script in a command shell (top and bottom)

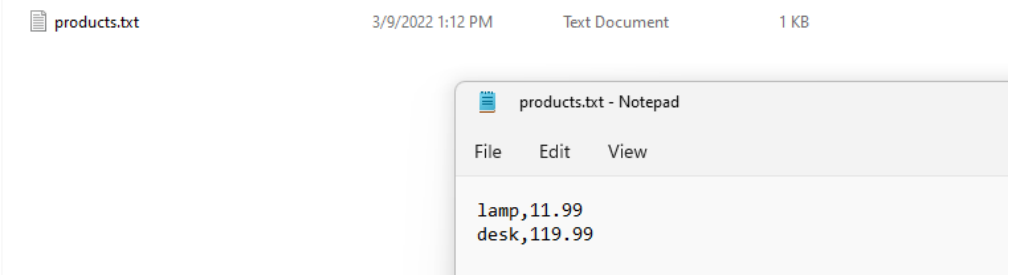

Figure 10. A screenshot of the content of products.txt (final output) and its location after running the code in the command shell

3. Summary

For this assignment, I modified a code that is created by someone else by adding three classes. It was not an easy task as I needed to add more and more functions and organize the codes using classes.