Denise Albano
3/8/22
Foundations of Programming: Python
Assignment08
https://github.com/dla425/IntoToProg-Python-Mod08

Classes Python Script

Introduction

This paper will document the steps I took to modify a script with three classes in Python. In order to complete the assignment, I watched the course video in module 8 by Randall Root, read chapter87 in text book, and reviewed the additional web page assigned.

Creating the Python Script

For this assignment I used PyCharm to modify the assignment08 starter python script. To start, I created a new project in PyCharm that uses the folder _PythonClass\Assignment08. Within this project, I created a new python script file named Assignment08.py (see figure 1) by copying the information from the assignment08 starter python script.

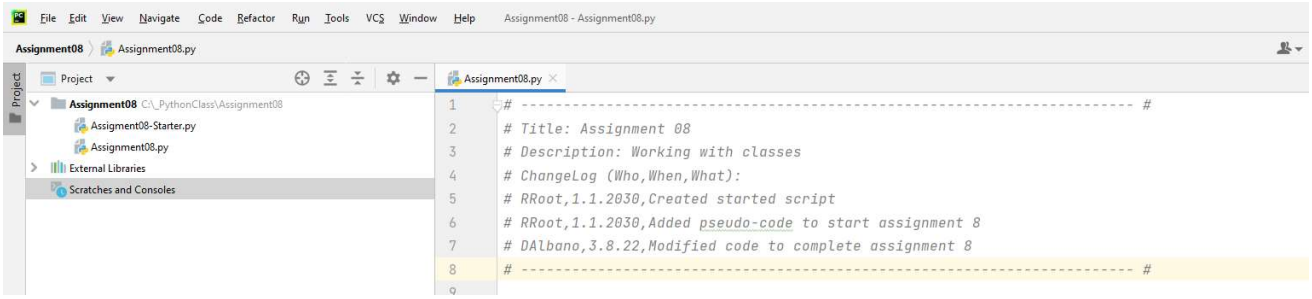


Figure 1: New project and python script in PyCharm

In my assignment08.py script, I updated the header to include my name and date in the change log. Since this assignment needed to read data from a text file, I created the 'products.txt' file and added a product and price (see Figure 2).

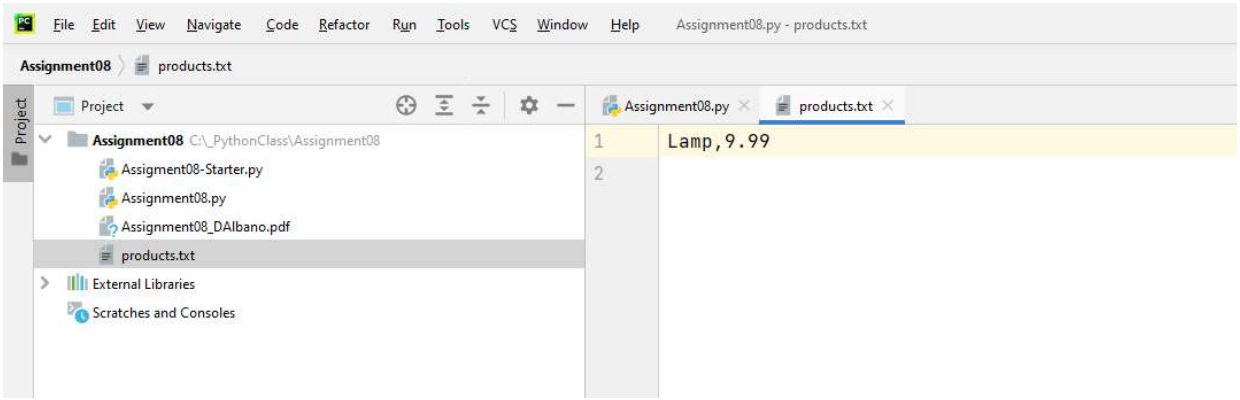


Figure 2: Screenshot of proudcts.txt file

Class Product:

To add code to the Product class, I reviewed lab 8-5 in the module which showed how to set up a class with a constructor, attributes, properties, and methods. First, I created a constructor to set the initial values of the fields, product_name and product_price, using the __init__() method. I then defined the attributes inside the constructor using the self parameter. By using the two underscore characters at the beginning of the attribute name, I designated the attributes as private.

Next, I created properties to manage the attributes. For each attribute, I set up a "getter" and "setter" property. For product_name, the function in the getter property formatted the data into a string and used the title() method to return the string with the first character in every word as upper case. The function in the setter property, also included error handling to ensure that the product name was valid. For product_price, the getter property formatted and returned a float variable and the setter property included error handling to ensure the price was a number and greater than 0.

Next, I created a method to override the built in __str__() method. To do this I created a function to return the product name and price with a comma separator. Lastly, I create a list, list_of_product_objects, containing the product names and prices.

Class FileProcessor:

Within the File Processor class, I created two static methods to read data from the file and save data to the file. I used static methods as this class since it processes the data to and from a file.

The function to read data from a file used the open() function to open the file for reading (using 'r' mode). To read data from the file, I used a for loop to read each line of the file using the split() method, and append the indexed data to a list of products. Once all the lines have been read, I used the close() method to close the file and then returned the list of products.

The function to save data to the file used the open() function to open the file for writing (using the 'w' mode). I then used a for loop to write each product in list_of_product_objects to the file using the __str__() method and adding a newline after each product. I then used the close() method to close the file.

Class IO:

In the input/output class, I created four static methods to show the menu of options to the user, get the user's choice, show the current data to the user, and allow the user to add new items.

The function to show the menu of options used a while loop to create the menu of choices for the user and a triple quoted string so that the menu text would be displayed over multiple lines.

The function to get the user's choice, created a string variable, choice, and used the input() function to ask the user to enter their choice from the menu between 1 and 4. A return statement was then used to send the results back.

The function to show the current data, used a for loop to print out the product name and price line by line.

The function to add new items, created a string variable for the product name and a float variable for the product price using input() functions to ask the user for the product name and price. This function then called the Product class and used a return statement to send the results back.

Main Body of Script:

To start the main body of the script, I needed to load data from the products.txt file into a list of product objects by calling the read_data_from_file function from the FileProcessor class. Using the try-except error handling method, if the products.txt file did not exist then an error message was displayed along with documentation and type of the error.

The rest of the main body of script is a while loop that shows the menu to the user and calls the get_user_choice function from the IO class. Once the user enters a menu choice, an if … elif statement is used to call each function from the IO or FileProcessor classes that are associated with the menu option. For option 1, the script calls the show_current_data function from IO class. For option 2, the script calls the add_new_item from IO class and appends the list of product objects with the new product and its price. For option 3, the save_data_to_file function in the FileProcessor class to write the new data to the file. For option 4, a print() function displays goodbye to the user and exits the program.

The final code for this assignment is shown below:

```
# ------------------------------------------------------------------------ #
# 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
# DAlbano,3.8.22,Modified code to complete assignment 8
# ------------------------------------------------------------------------ #

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

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

    properties:
        product_name: (string) with the products'  name
        product_price: (float) with the products' standard price
    methods:
    changelog: (When,Who,What)
        RRoot,1.1.2030,Created Class
        DAlbano,3.8.22,Modified code to complete assignment 8
    """

    # -- 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).title()

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

    # 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():
            self.__product_price = value
        else:
            raise Exception("Prices must be numbers")

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

    def __str__(self):
        return self.product_name + ',' + str(self.product_price)

# Data ---------------------------------------------------------------------- #

# 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
        DAlano,3.8.22,Modified code to complete assignment 8
    """
    @staticmethod
    def read_data_from_file(file_name:str):
        list_of_product_rows = []
        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()
        return list_of_product_rows

    @staticmethod
    def save_data_to_file(file_name:str, list_of_product_objects:list):
        file = open(file_name, "w")
        for product in list_of_product_objects:
            file.write(product.__str__() + "\n")
        file.close()

# Processing  --------------------------------------------------------------- #

# Presentation (Input/Output)  ---------------------------------------------- #
class IO:
    """Performs Input and Output tasks:

    methods:
    show_menu():

    get_user_choice():
    # strChoice = ""
```

```
    show_current_data(list_of_rows):

    add new items()

    changelog: (When,Who,What)
        RRoot,1.1.2030,Created Class
        DAlano,3.8.22,Modified code to complete assignment 8
    """

    @staticmethod
    def show_menu():
        print('''
        Menu of Options
        1) Show current data
        2) Add new item
        3) Save data to file
        4) Exit program
        ''')
        print()

    @staticmethod
    def get_user_choice():
        strChoice = str(input("Which option would you like to perform [1 to 4]:
").strip())
        print()
        return strChoice

    @staticmethod
    def show_current_data(list_of_rows:list):
        print("The current products are:")
        for row in list_of_rows:
            print(row.product_name + ' (' + str(row.product_price) + ')')
            print()

    @staticmethod
    def add_new_item():
        name = str(input("What is the product name?  ").strip())
        price = float(input("What is the product price?  ").strip())
        print()
        p = Product(product_name=name, product_price=price)
        return p

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

# Main Body of Script  ---------------------------------------------------- #
# 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 and exit program

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

try:
    listOfProductObjects = FileProcessor.read_data_from_file(strFileName)

    while True:
        IO.show_menu()
        strChoice = IO.get_user_choice()
```

```
        if strChoice == '1':
            IO.show_current_data(listOfProductObjects)
            continue

        elif strChoice == '2':
            lstOfProductObjects.append(IO.add_new_item())
            continue

        elif strChoice == '3':
            FileProcessor.save_data_to_file(strFileName, lstOfProductObjects)
            continue

        elif strChoice == '4':
            print("Goodbye")
        break

except Exception as e:
    print("Text file must exist before running this script")
    print(e, e.__doc__, type(e), sep='\n')
```

Running the program:

I ran the program in PyCharm and entered in each menu choice to confirm that the script was working. Screenshots of the output in PyCharm is shown in Figure 3 as well as the final contents of the products.txt file in Figure 4.

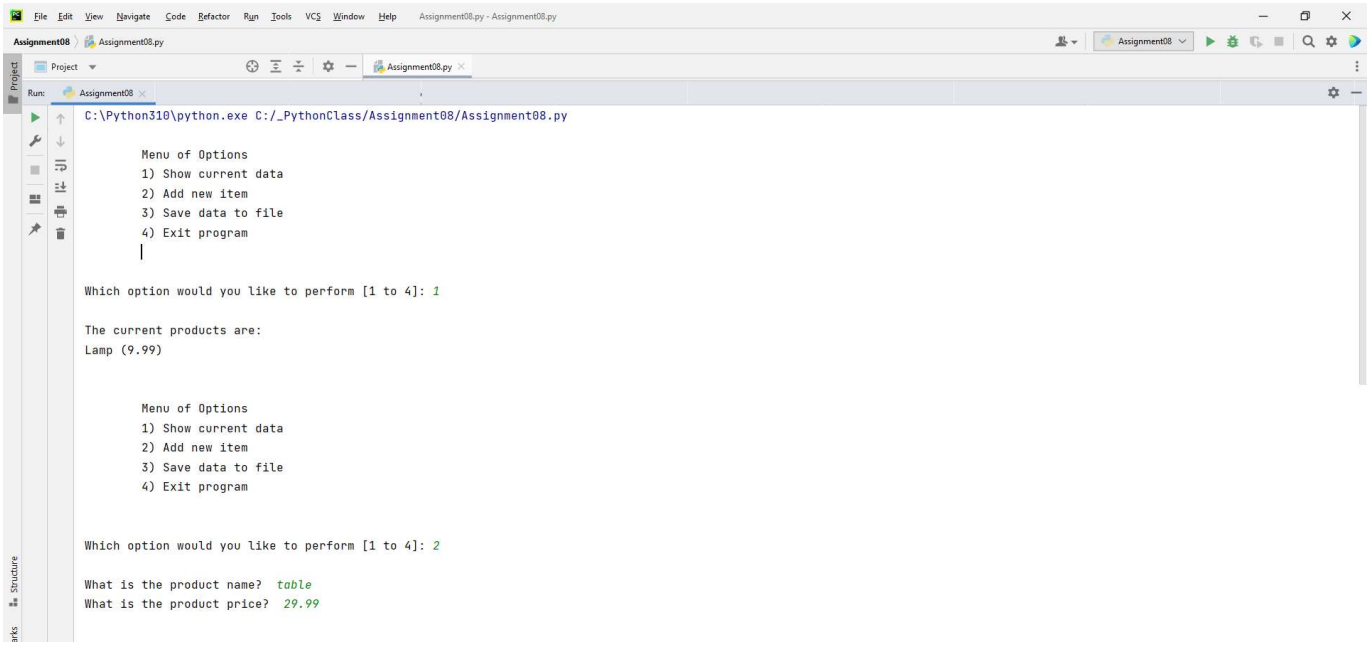


Figure 3: Screenshot of the script running in PyCharm

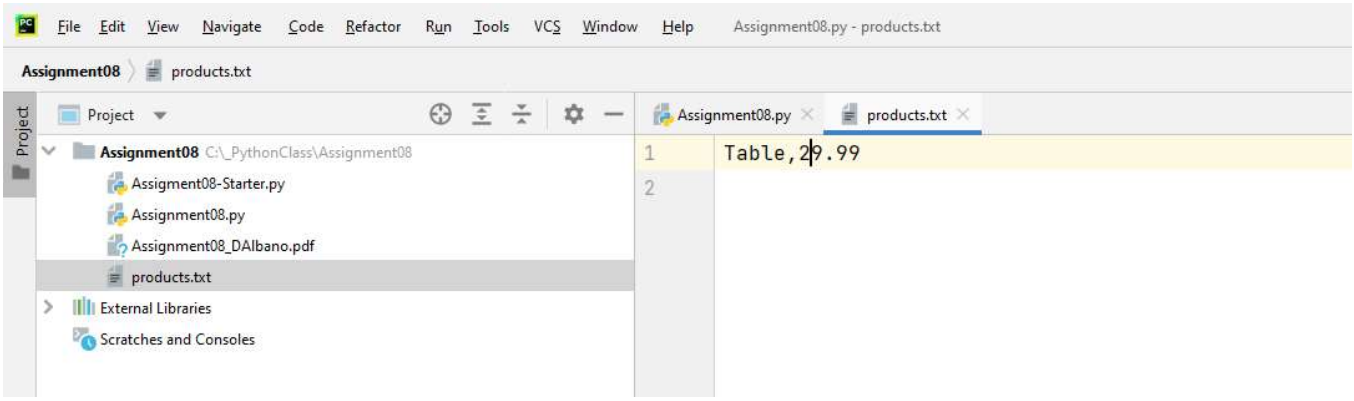
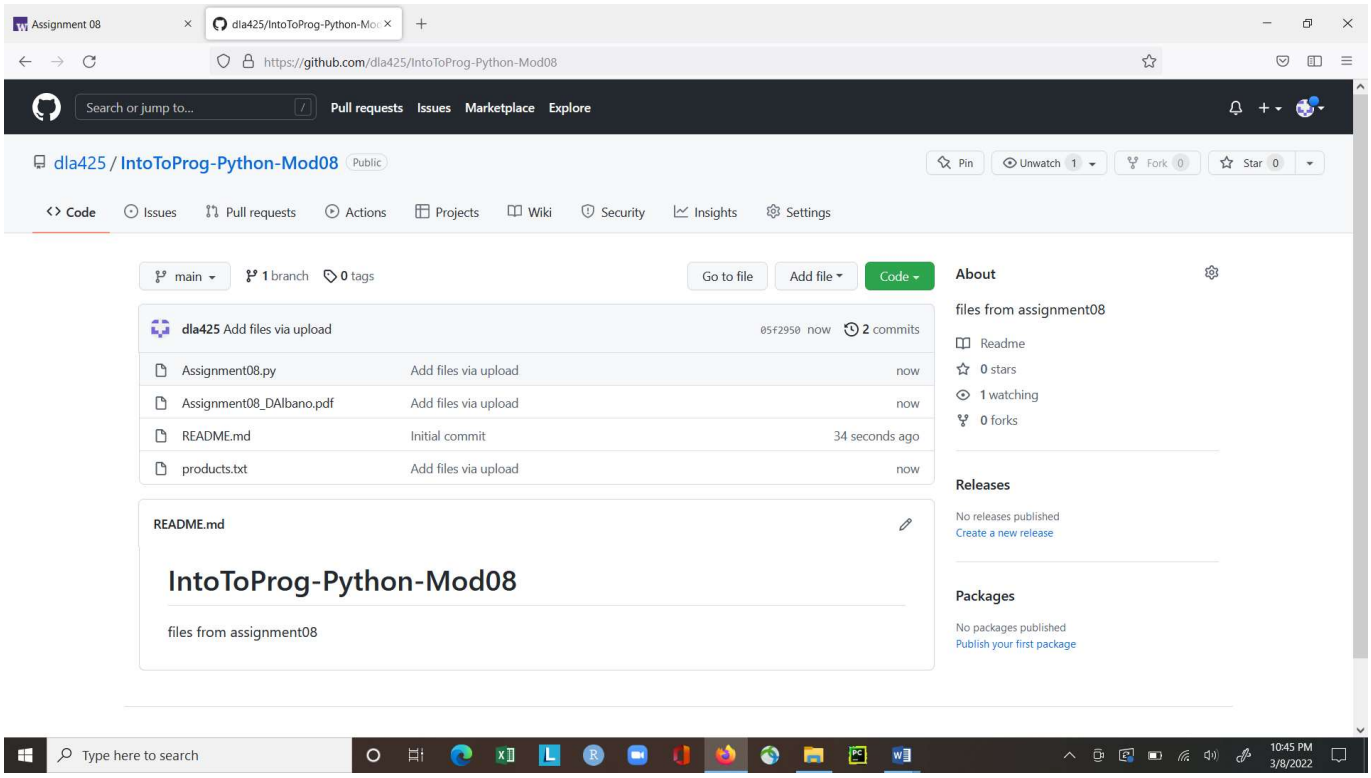


Figure 4: Screenshot of the products.txt file after running the program

Post Files to GitHub

After creating a GitHub repository named “IntoToProg-Python-Mod08” in my account, I uploaded the Assignement08 files and committed the changes.



Summary

To complete this assignment I needed to understand classes, as well constructors, attributes, properties and methods. It was important to also understand when to use the static method within a class and when not to. I found the examples in the course videos, textbook, and additional webpages to be extremely helpful in understanding the concepts needed to create a working program.