Alison Langer

August 30, 2021

Foundations of Programming: Python

Assignment 08

Github Link: https://github.com/alilan03/IntroToProg-Python-Mod08

Working With Classes: A Python Script

## **Introduction**

In this paper I will discuss the process of creating a to do list python script for assignment 08. The assignment requested a python script file that uses classes to store data about a product. This included five main concepts of user input/output, file I/O, error handling, classes and defining methods. This paper will begin with the initial file creation and will follow through to the final completion of the assignment with the functioning code.

### **Creating a Script File**

The first step for this assignment was to create a folder called “Assignment08” in the C: drive of the computer as a subfolder of the “\_PythonClass”. (Fig 1.1)

Graphical user interface, application

Description automatically generated

***Fig 1.1 Shows the “Assignment08” folder in the “\_PythonClass” folder***

The next step was to add the “Assignment08-Starter.py” file to the “Assignment08” folder.

Graphical user interface, application

Description automatically generated

***Fig 1.2 Shows the contents of the “Assignment08” folder***

The first step I took in writing the code for this assignment was creating a header to give a description of the project as well as provide a change log for updates to the code. (Fig 1.3)

Text

Description automatically generated

***Fig 1.3 The header for the “Assignment08” file***

### **Writing the Python Code**

To begin my project, I started by creating some initial fields for the product class as well as a constructor for the product object. This constructor would initialize the object’s fields to the values provided by the user. (Fig 2.1)

A picture containing text, screen, screenshot

Description automatically generated

***Fig 2.1 Shows the constructor for the product class***

Next I worked on the “FileProcessor” class by adding two file I/O methods. The first method was called “save\_data\_to\_file” which wrote the current data in the program to a specified file. This was achieved by opening the file in write mode and then looping through each product object and writing its attributes to the file. (Fig 2.2)

Text

Description automatically generated

***Fig 2.2 Shows the “save\_data\_to\_file” method in the “FileProcessor” class***

The next “FileProcessor” class was called “read\_data\_from\_file” which read the data from the file and added it to the list in the program. This was achieved by opening the file in read mode before looping through each item in the file and creating a new product object with the given attributes and then adding the new object to the list. (Fig 2.3)

Text

Description automatically generated

***Fig 2.3 The “read\_data\_from\_file” method in the “FileProcessor” class***

The next class in the program was called “IO” which handled all user input or output. The first portion of this task was to write a docstring for the class. (Fig 2.4)

Text

Description automatically generated

***Fig 2.4 The docstring for the “IO” class***

The first method in the “IO” class was called “show\_menu” which printed out the menu lines. (Fig 2.5)

Text

Description automatically generated

***Fig 2.5 The “show\_menu” method in the “IO” class***

The second I/O method was called “get\_choice” which asked the user to choose a menu option and then returned their choice. (Fig 2.6)

Text

Description automatically generated

***Fig 2.6 The “get\_choice” method in the “IO” class***

The third IO class was called “sow\_current\_data” which printed out all the products stored in the program’s list. This method would print out a counter with the product object’s attributes. (Fig 2.7)

Text

Description automatically generated

***Fig 2.7 The “show\_current\_data” method in the “IO” class***

The final method in the IO class was called “new\_product” which gathered the appropriate input to create a new product object and then added it to the program’s list. (Fig 2.8)

Text

Description automatically generated

***Fig 2.8 The “new\_product” method in the “IO” class***

The final portion of code was the main body of the script. This began with first calling the “read\_data\_from\_file” method to gather the initial data from the file. From there, the script enters a while loop that continues until the user chooses to exit the program. Based on the user’s menu choice, the program took a different branch of the if-elif-elif-else tree. Menu option 1 printed the current contents of the list. If there were no contents in the list, a special message was displayed for error handling. Option 2 would create a new product object. Option 3 would save the current data to the file upon the user’s request. Finally, an “else” branch was added for error handling if the user tried to choose a non-existent menu option. (Fig 2.9 and 2.10)

Text

Description automatically generated

***Fig 2.9 Main body script***

Text

Description automatically generated

***Fig 2.10 Main body script continued***

### **Running the Script**

The final portion of the assignment was to run the Python script in both PyCharm (Fig 3.1 - Fig 3.3) as well as a shell window (Fig 3.4 – Fig. 3.6) and record the running functionality.

Text

Description automatically generated

***Fig 3.1 Menu option 1: displaying the current data in the list***

Text

Description automatically generated

***Fig 3.2 Menu option 2: adding a new product***

Text

Description automatically generated

***Fig 3.3 Menu option 3: saving the current data and exiting the program***

Text

Description automatically generated

***Fig 3.4 Menu option 1: displaying the current data in the list***

Text

Description automatically generated

***Fig 3.5 Menu option 2: adding a new product***

Text

Description automatically generated

***Fig 3.6 Menu option 3: saving the current data and exiting the program***

### **Summary**

In this paper discussed the process of creating the to do list script in Python for assignment 08. This script uses classes to store data about a list of products. This included five main concepts of user input/output, file I/O, error handling, classes and defining methods Input is useful for obtaining data from the user to utilize in the program. Output is useful for displaying messages or prompts to the user. File I/O can be used to either read or write to a file. In this case we read initial data from a file and then wrote the current data to the file upon the user’s request. Error handling was demonstrated through various if-else statements that prevented unwanted results or program crashes. Classes and their respective methods were used to organize and separate the file processing actions, I/O and the product object class. Throughout my paper I discussed the steps and logic behind each of my decisions while coding the Python script for this assignment and concluded with the final display of the script running in a shell window.