Kyle Biondich

8/29/2023

IT FDN 110 A

Assignment 08

Class Funs

Introduction

Week 8 of the course introduced classes, class attributes, setters and getters. The following paragraphs outline the steps I went through to finish adding code to the starter assignment file for creating the product class constructor, parameters, and methods, as well as the file processor and IO methods for reading, writing, and saving data to a text file.

Intended Outcome

The intended outcome of this week is to present a menu of choices to the user, read product names and prices from a text file, add a new item to the list of products, save the list of items to a text file, and then exit out of the application.

```
PS D:\DEV\uw\assignments\_PythonClass\Assignment08> python .\_Assigment08-Starter.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] -
```

Figure 1: Intended Outcome: Menu Choices

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

******* The current items in the list are: *******
Cats (500.0)
Dogs (500.0)
House (400.0)
*******************************************


        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] -
```

Figure 2: Intended Outcome: Read Current Items

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

What is the product name? - boat
What is the product price? - 1200


        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] -
```

Figure 3: Intended Outcome: Add a new item

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


        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] -
```

Figure 4: Intended Outcome: Save data to file

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

PS D:\DEV\uw\assignments\_PythonClass\Assignment08>
```

Figure 5: Intended Outcome: Exit Program

Adding Code to the Product Class

The first task was to add the code for the constructor for the products class that contained properties product_name and product_price. This was done by first creating the initialization method and passing in self, product_name, and product_price, then setting them to the object upon initialization, as can be seen in lines 27 through 31 in Figure 6. Next, the properties were defined along with their setter methods, lines 33 through 54 in Figure 6. And finally the to_string method was created in a methods subsection, lines 56 through 62 in Figure 6.

```
11  # Data ---------------------------------------------------------------------- #
12  strFileName = 'products.txt'
13  lstOfProductObjects = []
14
15  class Product:
16      """Stores data about a product:
17
18      properties:
19          product_name: (string) with the product's  name
20
21          product_price: (float) with the product's standard price
22      methods:
23      changelog: (When,Who,What)
24          RRoot,1.1.2030,Created Class
25          K.Biondich,8-25-23,Modified code to complete assignment 8
26      """
27      # TODO: Add Code for Product class (Constructor, Properties, & Methods)
28      # Constructor
29      def __init__(self, product_name, product_price):
30          self.__product_name = product_name
31          self.__product_price = product_price
32
33      # Properties
34      @property
35      def product_name(self):
36          return str(self.__product_name).title()
37
38      @product_name.setter
39      def product_name(self, value):
40          if str(value).isnumeric() == False:
41              self.__product_name = value
42          else:
43              raise Exception("Product names cannot be numbers")
44
45      @property
46      def product_price(self):
47          return float(self.__product_price)
48
49      @product_price.setter
50      def product_price(self, value):
51          if str(value).isnumeric() == True:
52              self.__product_price = value
53          else:
54              raise Exception("Product prices must be numbers")
55
56      # Methods
57      def to_string(self):
58          ''' Returns object data as a string
59              Product Name + Product Price
60          '''
61          object_data = self.__product_name + ',' + str(self.__product_price)
62          return object_data
63
64  # Data ---------------------------------------------------------------------- #
```

Figure 6: Product Class

File Processor Class

The next class is the FileProcessor class, where functions pertaining to data processing into and out of a file are contained. The first method, read_data_from_file, lines 80-94 in Figure 7, reads data from a file into a list of objects. The next method, save_data_to_file, lines 97-108 in Figure 7, writes data from a list of objects to a products.txt file.

```
66  # Processing  ------------------------------------------------------------ #
67  class FileProcessor:
68      """Processes data to and from a file and a list of product objects:
69
70      methods:
71          save_data_to_file(file_name, list_of_product_objects):
72
73          read_data_from_file(file_name): -> (a list of product objects)
74
75      changelog: (When,Who,What)
76          RRoot,1.1.2030,Created Class
77          K.Biondich,8-25-23,Modified code to complete assignment 8
78      """
79      # TODO: Add Code to process data from a file
80      @staticmethod
81      def read_data_from_file(file_name):
82          ''' Reads data from a file into a list of objects
83
84          :param file_name: (string) with name of file:
85          :return: (list) of Product Class objects
86          '''
87          list_of_product_objects = []
88          file_obj = open(file_name, "a")
89          file_obj = open(file_name, "r")
90          for row in file_obj:
91              file_data = row.split(",")
92              product = Product(file_data[0].strip(),file_data[1].strip())
93              list_of_product_objects.append(product)
94          return list_of_product_objects
95
96      # TODO: Add Code to process data to a file
97      @staticmethod
98      def save_data_to_file(file_name, list_of_objects):
99          ''' Writes data from a list of objects to a file
100
101         :param file_name: (string) with name of file:
102         :param list_of_objects: (list) of objects:
103         :return: nothing
104         '''
105         file = open(file_name, 'w')
106         for row in list_of_objects:
107             file.write(row.to_string() + '\n')
108         file.close()
109
110 # Processing  ------------------------------------------------------------ #
```

Figure 7: File Processor Class

IO Class

The IO class contains the methods that pertain to interaction with the user. The first, print_menu_items, lines 125-136 in Figure 8, produces the printed menu list when called. The next method input_menu_choice, lines 138-152 in Figure 9, gets the menu choice from the user. The next method,

print_current_list_items, lines 154 – 165 in Figure 8, shows the current items in the list of product objects. And lastly, the method input_product_data, lines 167 – 187 in Figure 9, gets data for a product object from the user.

```
112  # Presentation (Input/Output)  -------------------------------------------- #
     You, 2 minutes ago | 1 author (You)
113  class IO:
114      """  A class for performing Input and Output
115      methods:
116          print_menu_items():
117          print_current_list_items(list_of_product_objects):
118          input_product_data():
119      changelog: (When,Who,What)
120          RRoot,1.1.2030,Created Class:
121          K.Biondich,8-25-23,Modified code to complete assignment 8
122      """
123      # Add code to show menu to user (Done for you as an example)
124      @staticmethod
125      def print_menu_items():
126          """  Display a menu of choices to the user
127          :return: nothing
128          """
129          print('''
130          Menu of Options
131          1) Show current data
132          2) Add a new item.
133          3) Save Data to File
134          4) Exit Program      You, 4 days ago • finish assignment 08 …
135          ''')
136          print()  # Add an extra line for looks in the terminal window
137
138      # TODO: Add code to get user's choice
139      @staticmethod
140      def input_menu_choice():
141          ''' Gets the menu choice from a user
142          :return: string
143          '''
144          choice = ''
145          try:
146              choice = str(input("Which option would you like to perform? [1 to 4] - ")).strip()
147              if choice.isnumeric() == False:
148                  raise Exception("Please enter a number between 1 and 4")
149              print()  # Add an extra line for looks in the terminal window
150          except Exception as e:
151              print(e)
152          return choice
153
```

Figure 8: IO Class

```
154     # TODO: Add code to show the current data from the file to user
155     @staticmethod
156     def print_current_list_items(list_of_objects):
157         ''' Shows the current items in the list of objects
158         :param list_of_objects: (list) of objects:
159         :return: nothing
160         '''
161         print("******* The current items in the list are: *******")
162         for row in list_of_objects:
163             print(row.product_name + ' (' + str(row.product_price) + ')')
164         print("*******************************************")
165         print()  # Add an extra line for looks in the terminal window
166 
167     # TODO: Add code to get product data from user
168     @staticmethod
169     def input_product_data():
170         ''' Gets data for a product object
171         :return: (object) product
172         '''
173         name = None
174         price = 0.0
175         newproduct = None
176     # try:
177         name = str(input("What is the product name? - ")).strip()
178         while name.isnumeric() == True:
179             print("Product names cannot be numbers")
180             name = str(input("What is the product name? - ")).strip()
181         try:
182             price = float(input("What is the product price? - "))
183         except ValueError:
184             print("Product prices must be numbers")
185         print()  # Add an extra line for looks in the terminal window
186         newproduct = Product(product_name=name, product_price=float(price))
187         return newproduct
188 
189 # Presentation (Input/Output)  -------------------------------------------- #
190
```

Figure 9: IO Class

Main Body Of Script

The main body of the script contains the code for running the script and initializes the list of product objects that may be stored in the products.txt file. Line 195 in Figure 9 creates the lstOfProductObjects variable and calls the FileProcessor.read_data_from_file method, passing it the strFileName variable that was initialized at runtime. Lines 197 – 219 in Figure 9 starts a while loop to always present the user with available script options, presenting first the menu (line 199), then a variable to capture the user's choice (line 201), then option 1 (lines 203 – 205), printing a list of the current items in the list, then option 2 (lines 207-209) the option to save new data to the list, option 3 (lines 211-213) the option to save the list to the products.txt file, and finally option 4 (lines 215-216) the option to exit the script.

```
# 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
lstOfProductObjects = FileProcessor.read_data_from_file(strFileName)
# Show user a menu of options
while True:
# Show user a menu of options
    IO.print_menu_items()
# Get user's menu option choice
    strChoice = IO.input_menu_choice()
    # Show user current data in the list of product objects
    if strChoice.strip() == '1':
        IO.print_current_list_items(lstOfProductObjects)
        continue
    # Let user add data to the list of product objects
    elif strChoice.strip() == '2':
        lstOfProductObjects.append(IO.input_product_data())
        continue
    # let user save current data to file and exit program
    elif strChoice.strip() == '3':
        FileProcessor.save_data_to_file(strFileName, lstOfProductObjects)
        continue
    # Let user exit program
    elif strChoice.strip() == '4':
        break
    else:
        print('Please enter a number between 1 and 4')
        continue

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

Figure 9: Main

Observations

This was a difficult assignment. The most difficult part was in the “input_product_data” method, and trying to capture the exception of a string passed into the price variable that wouldn’t end the script. I eventually created something that works, but I’m not sure if it’s a good solution. I used a while loop for capturing the product name, but then had to use a try block to stop the “cannot convert string to float” error from exiting out of the script.

Summary

In summary, utilizing all the resources provided to the class and the online lecture, this paper outlines all the steps that were taken to create a python script that results in a successful execution of the intended outcome (Figure 1). Following the steps outlined above will allow for the audience to recreate the presented result.

References

Questions. (n.d.). Retrieved from Stack Overflow: https://stackoverflow.com/questions/736043/checking-if-a-string-can-be-converted-to-float-in-python