File - C:\Users\camer_PythonClass\Assignment08\main.py

```
 1 # ------------------------------------------------------------------------ #
 2 # Title: Assignment 08
 3 # Description: Working with classes
 4 #
 5 # ChangeLog (Who,When,What):
 6 # RRoot,1.1.2030,Created started script
 7 # RRoot,1.1.2030,Added pseudo-code to start assignment 8
 8 # CMathews,8.29.2021,Modified code to complete assignment 8
 9 # ------------------------------------------------------------------------ #
10
11 # Data -------------------------------------------------------------------- #
12 strFileName = 'products.txt'
13 objFile = None
14 lstOfProductObjects = []
15 dicRow = {}
16 lstTable = []
17 strChoice = ""
18 strProduct = ""
19 fltPrice = ""
20 strStatus = ""
21
22 # --- Make the Class ---
23 class Product:
24     """Stores data about a product:
25
26     properties:
27         product_name: (string) with the product's name
28         product_price: (float) with the product's standard price
29     methods:
30         __str__: (string) product_info (product name and price)
31     changelog: (When,Who,What)
32         RRoot,1.1.2030,Created Class
33         CMathews,8.29.2021,Modified code to complete assignment 8
34     """
35
36     # -- Constructor --
37     def __init__(self, product_name, product_price):
38         try:
```

```
39              self.product_name = product_name
40              self.product_price = product_price
41          except Exception as e:
42              raise Exception("\nError")
43
44      def __str__(self):
45          return str(self.product_name) + "," + str(self.__product_price)
46
47      # -- Properties --
48      @property
49      def product_name(self):
50          return str(self.__product_name).title()
51
52      @product_name.setter
53      def product_name(self, value):
54          if str(value).isnumeric() == False:
55              self.__product_name = value
56          else:
57              raise Exception("\nProduct names cannot be numbers.")
58
59      @property
60      def product_price(self):
61          return str(self.__product_price)
62
63      @product_price.setter
64      def product_price(self, value):
65          if str(value).isnumeric() == True:
66              self.__product_price = value
67          else:
68              raise Exception("\nProduct prices must be numbers.")
69
70
71  # Processing  ------------------------------------------------------------- #
72  class FileProcessor:
73      """Processes data to and from a file and a list of product objects:
74
75      methods:
76          save_file(file_name, list_of_product_objects):
```

```
        read_file(file_name): -> (a list of product objects)

    changelog: (When,Who,What)
        RRoot,1.1.2030,Created Class
        CMathews,8.29.2021,Modified code to complete assignment 8
    """

    @staticmethod
    def save_file(file_name, lstOfProductObjects):
        """Writes dictionary data to text file

        :param: file_name: (object) of text file
        :param: lstOfProductObjects: (list) of dictionary rows
        :return: (list) of dictionary rows, (string) message
        """
        file = open(file_name, "w")
        for row in lstOfProductObjects:
            file.write(row.__str__() + "\n")
        file.close()
        print("\nYour data has been saved!")

    @staticmethod
    def read_file(file_name):
        """Reads data from a file into a list of dictionary rows

        :param: file_name: (string) with name of file:
        :param: lstOfProductObjects: (list) to be filled with file data:
        :return: (list) of dictionary rows
        """
        lstOfProductObjects.clear()
        try:
            file = open(file_name, "r")
            for line in file:
                product, price = line.split(",")
                row = {"Product": product.strip(), "Price": price.strip()}
                lstOfProductObjects.append(row)
            file.close()
        except:
```

```
115             print("\nThere is no current data.")
116         return lstOfProductObjects
117 
118 # Presentation (Input/Output)  -------------------------------------------- #
119 class IO:
120 
121     @staticmethod
122     def print_menu():
123         print("""
124                 Menu:
125                 1 - Show Current Data
126                 2 - Add New Data
127                 3 - Save Data
128                 4 - Exit
129                 """)
130 
131     @staticmethod
132     def input_menu():
133         choice = str(input("Which option would you like to perform? ").strip())
134         return choice
135 
136     @staticmethod
137     def show_data():
138         """ Shows current data in list
139 
140         :param: lstOfProductObjects: (list) of product names and prices
141         :return: ()
142         """
143         if lstOfProductObjects != []:
144             print("\nHere is the current data: ")
145             for row in lstOfProductObjects:
146                 row = row.__str__()
147                 print(row.strip())
148         else:
149             print("\nThere is no current data.")
150 
151     @staticmethod
152     def input_yes_no_choice():
```

```
153         """Gets a yes or no response from the user
154 
155         :return: (string)
156         """
157         choice = str(input())
158         choice = choice.strip().lower()
159         return choice
160 
161     @staticmethod
162     def input_press_to_continue(optional_message=''):
163         """Pause program and show message before continuing
164 
165         :param: optional_message:
166         :return:
167         """
168         print(optional_message)
169         input('Press Enter to continue.')
170 
171     @staticmethod
172     def add_product_to_list():  # Get user input and add it to the list.
173         """Adds user data to the dictionary list
174 
175         :param: product_name: (string) with the product's name
176         :param: product_price: (float) with the product's standard price
177         :param: lstOfProductObjects: (list) of dictionary rows
178         :return: (list) of dictionary rows
179         """
180 
181         product_name = str(input("Product Name: ").strip())
182         while True:
183             try:
184                 product_price = float(input("Price: "))
185                 if type(product_price) == float:
186                     product_details = product_name, product_price
187                     lstOfProductObjects.append(product_details)
188                     return product_details
189                 else:
190                     continue
```

```
            except ValueError:
                print("Product prices must be numbers.")


# Main Body of Script  ------------------------------------------------- #
try:
    lstOfProductObjects = FileProcessor.read_file(strFileName)  # Load file data.
    IO.show_data()
    while True:
        IO.print_menu()  # Show menu options.
        choice = IO.input_menu()
        if choice == "1":  # Show current data.
            IO.show_data()
            continue
        elif choice == "2":  # Add data to list.
            IO.add_product_to_list()
            print("\nYour data has been added!")
            IO.input_press_to_continue()
            continue
        elif choice == "3":  # Save data to file.
            FileProcessor.save_file(strFileName, lstOfProductObjects)
            IO.input_press_to_continue()
            continue
        elif choice == "4":  # Exit program.
            print("\nAre you sure you want to exit? Unsaved changes will be lost. (Y/N): ")
            strChoice = IO.input_yes_no_choice()
            if strChoice.lower() == "y":
                print("\nGoodbye!")
                break
            else:
                IO.input_press_to_continue()
            continue
        else:  # All non-menu selections.
            print("\nPlease choose from the available options.")

except Exception as e:  # All non-custom errors.
    print("\nError!")
    print(e, e.__doc__, type(e), sep='\n')
```

```
229
230
231
232
```