Charles Tung

June 2, 2021

Student ID: 9721184

Foundations of Programming, Python

Assignment 08

[charleslautung/ITFnd100-Mod08: Files For Mod8 (github.com)](https://github.com/charleslautung/ITFnd100-Mod08)

# Assingment08\_Charles.py

## Introduction

Homework Assignment08 of the Foundations of Programming, Python class involved creating a Assignment08\_Charles.py script. Assignment08\_Charles.py script that is used to demonstrate the use of classes and methods in Python. The Assignment08.py script is used to track the product name and its associated costs. A product class was created within the script store the product name and price. A Class Processer was created to process data used in the product class. Likewise a IO class was created for input and output of the script.

## Show Current Data

The below screen shot in figure 1 show the current data being shown (Option 1):

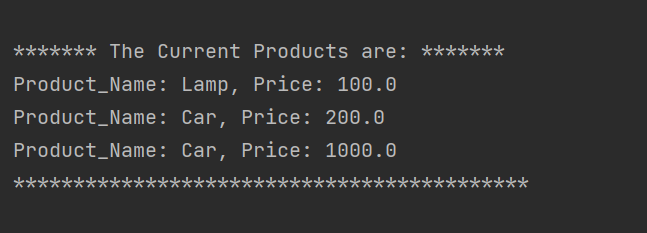


Figure 1: Screen Shot of PyCharm Showing the Current Data

## Add a New Product

The below screen shot in figure 2 shows the (Option 2) adding a new item:

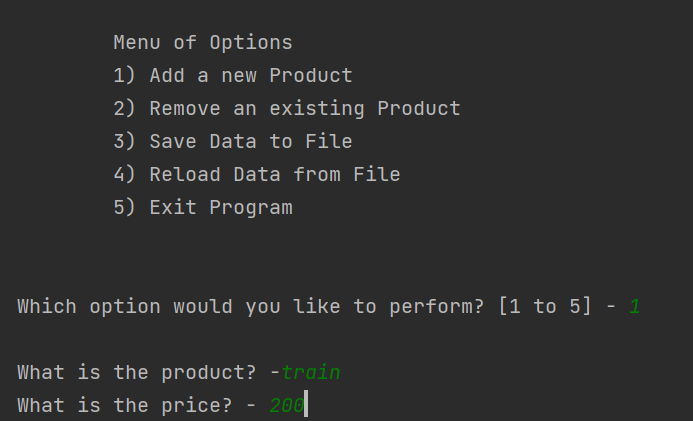


Figure 2: Screen Shot of PyCharm showing Option 1 (Add a new Item)

The below screen shot in figure 3 shows (Option 1) and the new item that was added below:

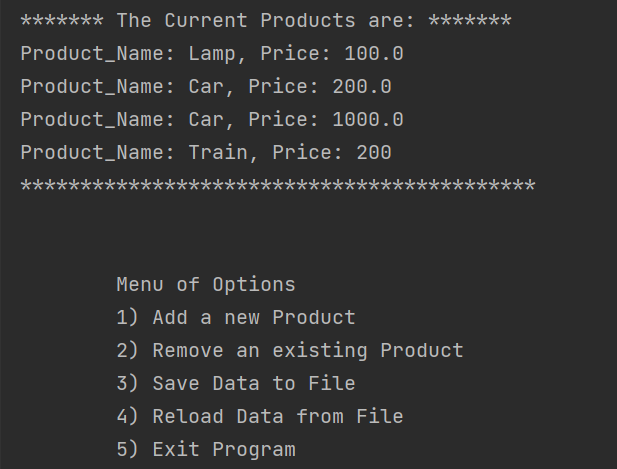


Figure 3: Screen Shot of PyCharm showing Option 1 and the additional “Train” Product

## Save Data to File:

Option 3 was selected in figure 4 below which saved the file to a text file as shown below:

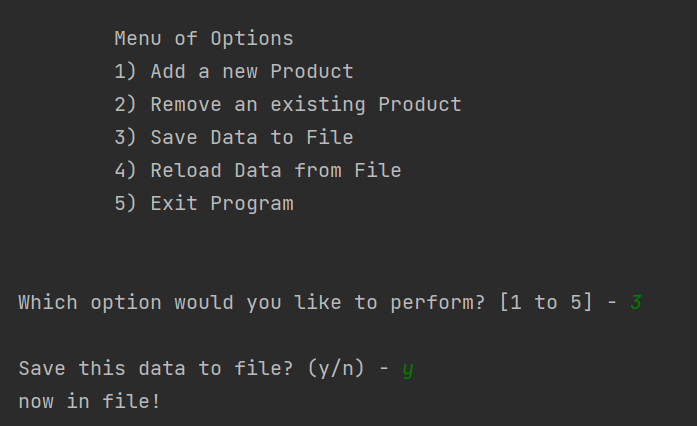


Figure 4: Screen Shot of PyCharm showing Option 3 and Saving the Data to a File

Below shows the text file Products.txt output in Figure 5 below and the additional train from the action shown in Figure 4 above:

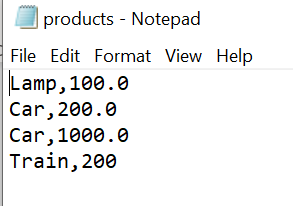


Figure 5: Screen Shot of Products.txt

## Remove Product:

The script can also remove a task. Below Figure 6 shows the task Eat removed as shown below:

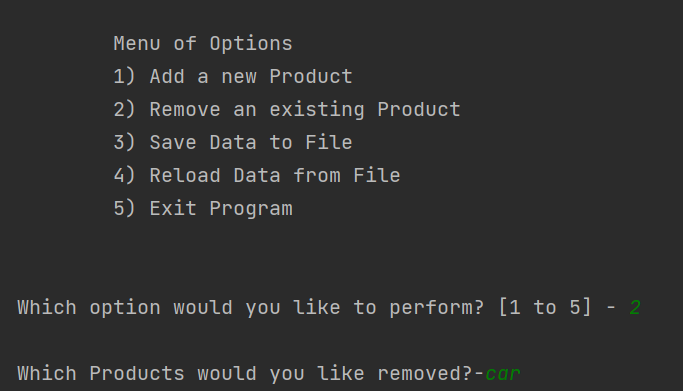


Figure 6: Screen Shot of Pycharm Showing the Task “Eat” Being Removed.

## Windows Command Screen Output

Below shows the screen shot (figure 7) from the windows command screen showing that the identical image from the PyCharm screenshot in Figure 1:

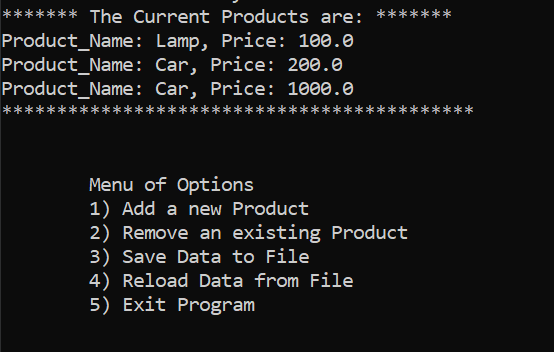


Figure 7: Windows Command Output of Assigment08\_Charles.py Script

## Source Code

The source code for Assigment08\_Charles.py was as shown in the PyCharm terminal as 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  
# CLTUNG, June 2, 2021,Modified code to complete assignment 8  
# CLTUNG, June 2, 2021, Added Product Class with Constructor and Getter/Setter  
# for Product Name and Price  
# CLTUNG, June 2, 2021, Added Process Class with Constructor and Getter/Setter  
# to Process Data  
# CLTUNG, June 2, 2021, Added IO Class for Input and Output  
# CLTUNG, June 2, 2021, Added Main Event Loop  
# ------------------------------------------------------------------------ #  
  
# Data -------------------------------------------------------------------- #  
strFileName = 'products.txt'  
lstTable = []  
strChoice = ""  
strStatus = ""  
  
class Product:  
 *"""Stores data about a product:  
  
 properties:  
 prod\_name: (string) with the products's name  
 price: (float) with the products's standard price  
 methods:  
 property: Getter for Product Name  
  
 changelog: (When,Who,What)  
 RRoot,1.1.2030,Created Class  
 <Your Name>,<Today's Date>,Modified code to complete assignment 8  
 """* # --Constructor--  
 def \_\_init\_\_(self, prod\_name, price):  
 self.\_\_prod\_name = prod\_name  
 self.\_\_price = price  
  
 # -- Properties --  
 # Product Name  
 @property  
 def prod\_name(self): # getter for product name  
 return str(self.\_\_prod\_name).title()  
  
 @prod\_name.setter  
 def prod\_name(self, value): # setter for product name  
 if str(value).isnumeric() == False:  
 self.\_\_prod\_name == value  
 else:  
 raise Exception("Names cannot be numbered")  
  
 # Price  
 @property  
 def price(self): # getter for price  
 return str(self.\_\_price)  
  
 @price.setter # setter for price  
 def price(self, value):  
 if value.isnumeric() == True:  
 self.\_\_price = value  
 else:  
 print("Need Enter Number. Not String")  
  
 # *TODO: Add Code to the Product class*# Data -------------------------------------------------------------------- #  
  
# Processing ------------------------------------------------------------- #  
class Processor:  
 *"""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  
 <Your Name>,<Today's Date>,Modified code to complete assignment 8  
 """* @staticmethod  
 def read\_data\_from\_file(file\_name):  
 *""" Reads data from a file into a list of dictionary rows  
 :param file\_name: (string) with name of file:  
 :return: (list) of objects  
 """* list\_of\_objects = []  
 file = open(file\_name, "r")  
 for line in file:  
 data = line.split(",")  
  
 prod\_name = data[0].strip()  
 price = float(data[1].strip())  
 list\_of\_objects.append(Product(prod\_name, price))  
  
 file.close()  
 return list\_of\_objects  
  
 @staticmethod  
 def add\_data\_to\_list(list\_of\_objects, prod\_name, price):  
 *"""Adds data to a list of dictionary row  
 :param list\_of\_objects (list) of objects adding data to it  
 :param prod\_name: (string) with name of product  
 :param price: (string) with price  
 """* list\_of\_objects.append(Product(prod\_name, price))  
  
 @staticmethod  
 def remove\_data\_from\_list(list\_of\_objects, strKeyToRemove):  
 *"""Removes data from a list of dictionary rows  
 :param list\_of\_objects: (list) of objects removing from it  
 :param strKeyToRemove: Product to Remove  
 :return: Revised list of objects  
 """* sucess\_status = False  
 prod\_number = 0  
 for Product in list\_of\_objects:  
 if Product.prod\_name == strKeyToRemove:  
 del list\_of\_objects[prod\_number]  
 sucess\_status = True  
 prod\_number += 1  
 return list\_of\_objects, 'Success'  
  
 @staticmethod  
 def write\_data\_to\_file(file\_name, list\_of\_objects):  
 f = open(file\_name, "w")  
 for Product in list\_of\_objects:  
 f.write(str(Product.prod\_name + ',' + Product.price + '\n'))  
 f.close()  
 print("now in file!")  
 return list\_of\_objects  
  
# Processing ------------------------------------------------------------- #  
  
# Presentation (Input/Output) -------------------------------------------- #  
class IO:  
 # Print Doc String  
 print(Product.\_\_doc\_\_)  
  
 """ Performs Input and Output Products """  
  
 @staticmethod  
 def print\_menu\_Products():  
 *""" Display a menu of choices to the user  
  
 :return: nothing  
 """* print('''  
 Menu of Options  
 1) Add a new Product   
 2) Remove an existing Product  
 3) Save Data to File   
 4) Reload Data from File  
 5) Exit Program  
 ''')  
 print() # Add an extra line for looks  
  
 @staticmethod  
 def input\_menu\_choice():  
 *""" Gets the menu choice from a user  
 :return: string  
 """* try: # Error Handling for User Entry  
 choice = str(input("Which option would you like to perform? [1 to 5] - ")).strip()  
 except ValueError:  
 print("Invalid Entry. Please select a valid choice")  
 print() # Add an extra line for looks  
 return choice  
  
 @staticmethod  
 def print\_current\_Products\_in\_list(lstTable):  
 *""" Shows the current Products in the list of dictionaries rows  
 :param lstTable: (list) of objects you want to display  
 :return: nothing  
 """* print("\*\*\*\*\*\*\* The Current Products are: \*\*\*\*\*\*\*")  
 for Product in lstTable:  
 print("Product\_Name: {}, Price: {}".format(Product.prod\_name, Product.price))  
 #print(row["Product\_Name"] + " (" + row["Price"] + ")")  
 print("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")  
 print() # Add an extra line for looks  
  
 @staticmethod  
 def input\_yes\_no\_choice(message):  
 *""" Gets a yes or no choice from the user  
 :return: string  
 """* return str(input(message)).strip().lower()  
  
 @staticmethod  
 def input\_press\_to\_continue(optional\_message=''):  
 *""" Pause program and show a message before continuing  
 :param optional\_message: An optional message you want to display  
 :return: nothing  
 """* print(optional\_message)  
 input('Press the [Enter] key to continue.')  
  
 @staticmethod  
 def input\_new\_product\_and\_price():  
 *"""Gets data for a dictionary rows  
 :return: (tuple) of string with Products and Price  
 """* prod\_name = str(input("What is the product? -")).strip()  
 price = input("What is the price? - ").strip()  
  
 print() # Add an extra line for looks  
 return prod\_name, price  
  
 @staticmethod  
 def input\_product\_to\_remove():  
 *"""Gets a Product from the user to remove  
 :return: (string) Product to remove  
 """* prod\_name = str(input("What product would you like to remove? - ")).strip()  
 return prod\_name  
  
# Main Body of Script ---------------------------------------------------- #  
# Step 1 - When the program starts, Load data from ToDoFile.txt.  
lstTable = Processor.read\_data\_from\_file(strFileName) # read file data  
  
  
# Step 2 - Display a menu of choices to the user  
while (True):  
 # Step 3 Show current data  
 IO.print\_current\_Products\_in\_list(lstTable) # Show current data in the list/table  
 IO.print\_menu\_Products() # Shows menu  
 strChoice = IO.input\_menu\_choice() # Get menu option  
  
 # Step 4 - Process user's menu choice  
 if strChoice.strip() == '1': # Adds a New Products  
 tplData = IO.input\_new\_product\_and\_price() # Outputs Tuple with New Product & Priority  
 Processor.add\_data\_to\_list(lstTable, tplData[0], tplData[1]) # Adds Data to List  
 IO.print\_current\_Products\_in\_list(lstTable)  
 print("Added New Product Successfully")  
 continue # to show the menu  
  
 elif strChoice == '2': # Removes an Existing Product  
 strKeyToRemove = input("Which Products would you like removed?-")  
 blnItemRemoved = Processor.remove\_data\_from\_list(lstTable, strKeyToRemove)  
 continue # to show the menu  
  
 elif strChoice == '3': # Save Data to File  
 strChoice = IO.input\_yes\_no\_choice("Save this data to file? (y/n) - ")  
 if strChoice.lower() == "y":  
 Processor.write\_data\_to\_file(strFileName, lstTable) # Writes Data to File  
 IO.input\_press\_to\_continue(strStatus)  
 else:  
 IO.input\_press\_to\_continue("Save Cancelled!")  
 continue # to show the menu  
  
 elif strChoice == '4': # Reload Data from File  
 print("Warning: Unsaved Data Will Be Lost!")  
 strChoice = IO.input\_yes\_no\_choice("Are you sure you want to reload data from file? (y/n) - ")  
 if strChoice.lower() == 'y':  
 lstTable.clear()  
 lstTable = Processor.read\_data\_from\_file(strFileName) # Outputs List and Sucess  
 IO.print\_current\_Products\_in\_list(lstTable)  
 else:  
 IO.input\_press\_to\_continue("File Reload Cancelled!")  
 continue # to show the menu  
  
 elif strChoice == '5': # Exit Program  
 print("Goodbye!")  
 break # and Exit

# --------------------------------------------------------------------------

Figure 8: Screenshot of Source Code to Assigment06\_Charles.py Written in PyCharm’s Terminal

The main event loop is the While Loop that calls out the IO Classes. The IO Classes then calls out the Processor Class which will call out the Product Class. The Classes were set up to help better organize the script source code by calling out objects.

## Summary

This paper summarizes some of the Assigment08\_Charles.py script. The Assignment08.py script is used to track the product name and its associated costs. A product class was created within the script store the product name and price. A Class Processer was created to process data used in the product class. Likewise a IO class was created for input and output of the script.