**Kate Pollock**

**December 4, 2021**

**Foundations of Programming: Python**

**Assignment08**

**GitHubURL <**<https://github.com/katepollock/IntroToProg-Python-Mod08>**>**

# **Classes**

# Introduction

In Module 8, I learned about classes as a way to group data and functions and objects to access the code in the class. I additionally learned about many of the other major concepts that make up a class such as constructors, setters and getters as well as attributes. I used my new knowledge to modify a script that utilizes a Product class to create product objects with name and price attributes.

# Planning my “Product ” Script

I began my scripts with a header and the global variable for my text file (***See Figure 1***)

*# ------------------------------------------------------------------------ #  
# 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  
# KPollock,12/3/21, Modified code to complete assignment 8  
# ------------------------------------------------------------------------ #  
  
# Data -------------------------------------------------------------------- #*strFileName = **'products.txt'**

***Figure 1 – Script Heading and Variable***

Product Script

My script contains 3 classes: Product, File Processor and Input/Output.

The Product class is displayed below in Figure 2. It has a constructor which automatically runs when I create an object from the class. The constructor has 2 parameters – name and price. I used the key word ‘self’ to indicate an object instance. I used an input from user method as a part of this class to request a name and price and instantiate an object. I used getters and setters for both name and price – raising an exception if the name contains numbers and if the price is not an integer or float. The\_\_str\_\_method overrides the default object and returns the object attributes as strings. ***See Figure 2*.**

**class** Product:  
 *"""Stores data about a product:  
  
 properties:  
 name: (string) with the products's name  
 price: (float) with the products's price  
 methods:  
 str method to change default to product name and product price  
 changelog: (When,Who,What)  
 RRoot,1.1.2030,Created Class  
 KPollock, 12/3/21, Modified code to complete assignment 8  
 """  
  
 # --Constructor--#* **def** \_\_init\_\_(self, name, price):  
 *# --Attribute--#* self.name = name  
 self.price = price  
  
 @staticmethod  
 **def** input\_from\_user():  
 name = input(**'What product would you like to add? '**).replace(**','**, **''**)  
 price = input(**'What is the price of the product? '**).replace(**','**, **''**)  
 print() *# extra line for looks* **return** Product(name, float(price))  
  
 *# --Properties--#* @property  
 **def** name(self): *# getter* **return** self.\_\_name  
  
 @name.setter  
 **def** name(self, value): *# setter* **if not** value.isnumeric():  
 self.\_\_name = value  
 **else**:  
 **raise** Exception(**"Names cannot be numbers"**)  
  
 @property  
 **def** price(self): *# getter* **return** self.\_\_price  
  
 @price.setter  
 **def** price(self, value): *# setter* **if** type(value) == float **or** type(value) == int:  
 self.\_\_price = value  
 **elif** type(value) == str **and** value.isnumeric():  
 self.\_\_price = float(value)  
 **else**:  
 **raise** Exception(**"Price must be a number"**)  
  
 *# ---Methods---#* **def** \_\_str\_\_(self):  
 **return** self.name + **','** + str(self.price)

***Figure 2 – Product Class code***

The file processing class saves data to a text file and reads data from the file. I used the static method as this class is focused on processing data. ***See Figure 3***.

*# 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): --> return None  
 read\_data\_from\_file(file\_name): -> return (a list of product objects)  
  
 changelog: (When,Who,What)  
 RRoot,1.1.2030,Created Class  
 KPollock, 12.3.2021, Modified code to complete assignment 8  
 """* @staticmethod  
 **def** save\_data\_to\_file(file\_name, list\_of\_product\_objects):  
 *""" Writes data from the list to the file using csv* **:param** *list\_of\_product\_objects:* **:param** *file\_name: (string) with name of file* **:return***: list of rows that was written to file  
 """* file = open(file\_name, **"wt"**)  
 **for** product **in** list\_of\_product\_objects:  
 file.write(str(product) + **'\n'**)  
 file.close()  
 **return None** @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 product\_objects  
 """* list\_of\_product\_objects = [] *# start with empty list* **try**:  
 file = open(file\_name, **"r"**)  
 **except** FileNotFoundError:  
 **return** list\_of\_product\_objects  
  
 **for** line **in** file:  
 name, price = line.strip().split(**','**)  
 list\_of\_product\_objects.append(Product(name, float(price)))  
 file.close()  
 **return** list\_of\_product\_objects

***Figure 3 – File Processing code***

The IO Class performs methods including printing the menu for the user, prompting input of a menu choice, and displaying the products stored in the list. ***See Figure 4***.

*# Presentation (Input/Output) -------------------------------------------- #***class** IO:  
 *"""Performs input and output tasks:  
  
 methods: print menu, input choice, print current products,  
 input new product and price  
  
 changelog: (When,Who,What)  
 RRoot,1.1.2030,Created Class  
 KPollock, 12.3.2021, Modified code to complete assignment 8  
 """  
  
 # Presentation (Input/Output) -------------------------------------------- #* @staticmethod  
 **def** print\_menu\_tasks():  
 *""" Display a menu of choices to the user* **:return***: nothing  
 """* print(**'''  
 Menu of Options  
 1) Display current data   
 2) Add a new Product  
 3) Save Data to File and Exit Program   
 '''**)  
 print() *# Add an extra line for looks* @staticmethod  
 **def** input\_menu\_choice():  
 *""" Gets the menu choice from a user* **:return***: string  
 """* choice = input(**"Which option would you like to perform? [1 to 3] - "**).strip()  
 print() *# Add an extra line for looks* **return** choice  
  
 @staticmethod  
 **def** print\_current\_products\_in\_list(products):  
 *""" Shows the current products in the list of dictionaries rows* **:param** *products: (list) of products you want to display* **:return***: nothing  
 """* print(**"\*\*\*\*\*\*\* The Current Products are: \*\*\*\*\*\*\*"**)  
 **for** product **in** products:  
 print(**f'{**product.name**} @ ${**product.price**:.2f}'**)  
 print(**"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"**)  
 print() *# Add an extra line for looks*

***Figure 4 – Input and Output code***

Main Body of Script

The main body of my script first reads the data from the file to update the list if needed. I then use a series of if-elif statements based on the user choice and call the appropriate methods to display, process the data and save the data to the file (***See Figure 5***).

*# 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***def** main():  
 lstOfProductObjects = FileProcessor.read\_data\_from\_file(strFileName)  
  
 **while True**:  
 IO.print\_menu\_tasks()  
 strChoice = IO.input\_menu\_choice()  
  
 **if** strChoice == **'1'**:  
 IO.print\_current\_products\_in\_list(lstOfProductObjects)  
  
 **elif** strChoice == **'2'**:  
 product = Product.input\_from\_user()  
 lstOfProductObjects.append(product)  
 print(product.name, **"has been added!"**)  
  
 **elif** strChoice == **'3'**:  
 FileProcessor.save\_data\_to\_file(strFileName, lstOfProductObjects)  
 print(**'Data has been saved. Goodbye!'**)  
 **break  
  
 else**:  
 print(**'Please enter choices [1 to 3]'**)  
  
main()

***Figure 5 – Main body of script***

# Results of Script

I ran the code in the command prompt and the results were as expected (***Figure 6***).

Text

Description automatically generated

***Figure 6: Output in Command Prompt***

# Summary

I have written the Python program above by utilizing the new concepts learned in Module 8 of this course. These concepts include classes, objects, methods, and attributes. I’m looking forward to learning about the concept of inheritance next week.