Kegan Sanchez

December 7, 2020

Foundations of Programming (Python)

Assignment 08

<https://github.com/kegans/IntroToProg-Python-Mod08>

Objects and Classes

# Introduction

This week, I modified a program that uses objects and classes to read and write to a file. This assignment included new components such as constructors, classes, methods, the self keyword, properties, attributes, and exception handling.

# Creating the Program

In addition to reviewing the module notes provided by the instructor, Randal Root, I began by reviewing the starter .py file and attempting to determine what I would need to incorporate within this program. I had a harder time grasping the subject matter of this module enough to apply it effectively, so my ability to complete the assignment on my own and before the deadline was limited.

## The Product Class

Since I didn’t get a good enough grasp on the topics to apply them, I used the assignment answer code and review video to supplement my knowledge. The entire section devoted to handling the data was predominantly made up of a Product class which stored data about a product in a list of product objects (Figure 1).

*# Data -------------------------------------------------------------------- #*

*strFileName = 'products.txt'*

*lstOfProductObjects = []*

*class Product:*

*"""Stores data about a product:*

*properties:*

*product\_name: (string) with the product's name*

*product\_price: (float) with the product's standard price*

*methods:*

*changelog: (When,Who,What)*

*RRoot,1.1.2030,Created Class*

*KSanchez,2020.12.09,Modified code to complete assignment 8*

*"""*

*# TODO: Add Code to the Product class (Done)*

*# -- Constructor --*

*def \_\_init\_\_(self, product\_name: str, product\_price: float):*

*"""Set name and price of a new object"""*

*# -- Attributes --*

*try:*

*self.\_\_product\_name = str(product\_name)*

*self.\_\_product\_price = float(product\_price)*

*except Exception as e:*

*raise Exception("Error setting initial values:\n" + str(e))*

*# -- Properties --*

*# product name*

*@property*

*def product\_name(self):*

*return str(self.\_\_product\_name)*

*@product\_name.setter*

*def product\_name(self, value: str):*

*if str(value).isnumeric():*

*self.\_\_product\_name = value*

*else:*

*raise Exception("Names cannot be numbers.")*

*# product price*

*@property*

*def product\_price(self):*

*return float(self.\_\_product\_price) # cast to float*

*@product\_price.setter*

*def product\_price(self, value: float):*

*if str(value).isnumeric():*

*self.\_\_product\_price = float(value) # cast to float*

*else:*

*raise Exception("Prices must be numbers.")*

*# -- Methods --*

*def to\_string(self):*

*"""alias of \_\_str\_\_(), converts product data to string"""*

*return self.\_\_str\_\_()*

*def \_\_str\_\_(self):*

*"""Converts product data to string"""*

*return self.\_\_product\_name + "," + str(self.product\_price)*

*# Data -------------------------------------------------------------------- #*

***Figure 1: Data – Product Class***

## The FileProcessor Class

After reviewing the Product class, I looked at the processing portion of the program, which was also made up of a single class called FileProcessor which handled data to and from a file and a list of product objects (Figure 2).

# 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):

read\_data\_from\_file(file\_name): -> (a list of product objects)

changelog: (When,Who,What)

RRoot,1.1.2030,Created Class

KSanchez,2020.12.09,Modified code to complete assignment 8

"""

# TODO: Add Code to process data from a file (Done)

@staticmethod

def save\_data\_to\_file(file\_name, list\_of\_product\_objects):

"""Writes data to a file from a list of product rows

:param file\_name: (string) with name of file

:param list\_of\_product\_objects: (list) of product objects data saved to file

:return: (bool) with status of success status

"""

success\_status = False

try:

file = open(file\_name, "w")

for product in list\_of\_product\_objects:

file.write(product.\_\_str\_\_() + "\n")

file.close()

success\_status = True

except Exception as e:

print("There was a general error!")

print(e, e.\_\_doc\_\_, type(e), sep="\n")

return success\_status

# TODO: Add Code to process data to a file (Done)

@staticmethod

def read\_data\_from\_file(file\_name: str):

"""Reads data from a file into a list of product rows

:param file\_name: (string) with name of file

:return: (list) of product rows

"""

list\_of\_product\_rows = []

try:

file = open(file\_name, "r")

for line in file:

data = line.split(",")

row = Product(data[0], data[1])

list\_of\_product\_rows.append(row)

file.close()

except Exception as e:

print("There was a general error!")

print(e, e.\_\_doc\_\_, type(e), sep="\n")

return list\_of\_product\_rows

# Processing ------------------------------------------------------------- #

***Figure 2: Processing – FileProcessor Class***

## The IO Class

The next portion of the program handled the presentation processes, or input/output and was mostly comprised of code that was used in previous file handling program assignments (Figure 3).

# Presentation (Input/Output) -------------------------------------------- #

class IO:

# TODO: Add docstring (Done)

"""A class for performing Input and Output

methods:

print\_menu\_items():

print\_current\_list\_items(list\_of\_rows):

input\_product\_data():

changelog:

KSanchez, 2020.12.09, Modified code to complete assignment 8

"""

# TODO: Add code to show menu to user (Done)

@staticmethod

def print\_menu\_items():

"""Displays a menu of choices for the user."""

print("""

Menu of Options

1) Show current data

2) Add a new item

3) Save data to a file

4) Exit program

""")

print() # Adds an extra line for looks

# TODO: Add code to get user's choice (Done)

@staticmethod

def input\_menu\_choice():

"""Gets the menu choice from a user

:return: string

"""

choice = str(input("Which option would you like to perform? [1 to 4]: ")).strip()

print() # Adds an extra line for looks

return choice

# TODO: Add code to show the current data from the file to user (Done)

@staticmethod

def print\_current\_list\_items(list\_of\_rows: list):

"""Prints the current items in the list of rows.

:param list\_of\_rows: (list) of rows you want to display

"""

print("\*\*\*\*\*\*\*The current product items are:\*\*\*\*\*\*\*")

for row in list\_of\_rows:

print(row.product\_name + "(" + str(row.product\_price) + ")")

print("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

print() # Adds an extra line for looks

# TODO: Add code to get product data from user (Done)

@staticmethod

def input\_product\_data():

"""Gets data for a product object

:return: (Product) object with input data

"""

try:

name = str(input("Product name: ").strip())

price = float(input("Product price: ").strip())

print() # Adds an extra line for looks

p = Product(product\_name=name, product\_price=price)

except Exception as e:

print(e)

return p

# Presentation (Input/Output) -------------------------------------------- #

***Figure 3: Presentation – IO Class***

## The Main Body

The last portion of the program was the main body, which took each of these classes and ran them in a while loop with exception handling to catch any errors (Figure 4).

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

# TODO: Add Data Code to the Main body (Done)

# Load data from file into a list of product objects when script starts

try:

lstOfProductObjects = FileProcessor.read\_data\_from\_file(strFileName)

while True:

# Show user a menu of options

IO.print\_menu\_items()

# Get user's menu option choice

strChoice = IO.input\_menu\_choice()

if strChoice.strip() == "1":

# Show user current data in the list of product objects

IO.print\_current\_list\_items(lstOfProductObjects)

continue

elif strChoice.strip() == "2":

# Let user add data to the list of product objects

lstOfProductObjects.append(IO.input\_product\_data())

continue

elif strChoice.strip() == "3":

# let user save current data to file and exit program

FileProcessor.save\_data\_to\_file(strFileName, lstOfProductObjects)

continue

elif strChoice.strip() == "4":

break

except Exception as e:

print("There was an error! Check file permissions.")

print(e, e.\_\_doc\_\_, type(e), sep="\n")

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

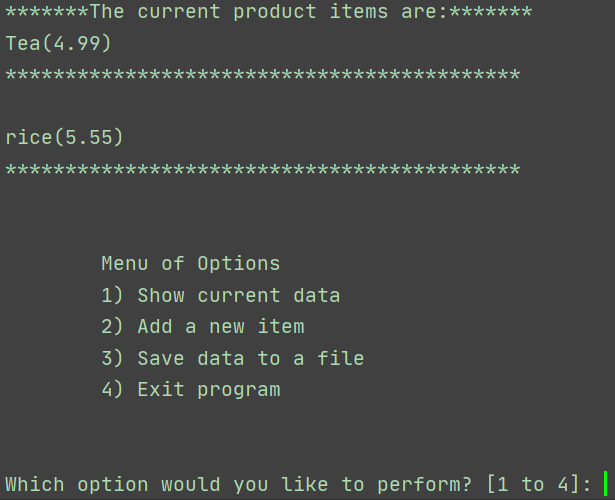
***Figure 4: Main Body***

# Testing

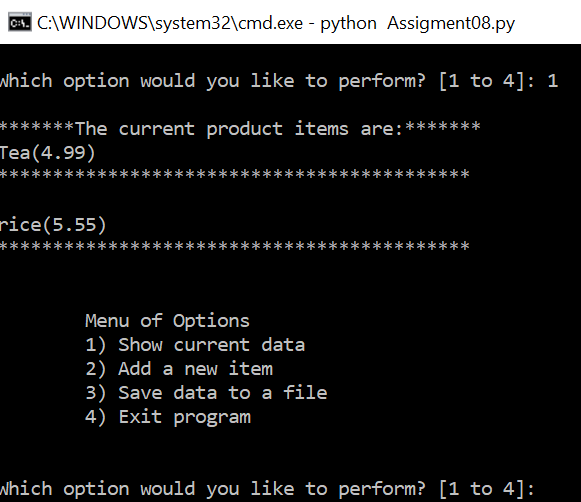
When I began creating this program, I completed all coding and testing primarily in the PyCharm IDE. To complete this assignment, I also tested in the command terminal.

## PyCharm and Command Terminal

I tested in PyCharm and then opened a command terminal, changed the directory to my assignment folder, and ran portions of my Assignment07.py script file. I received the desired output here, as well (Figures 5 and 6).

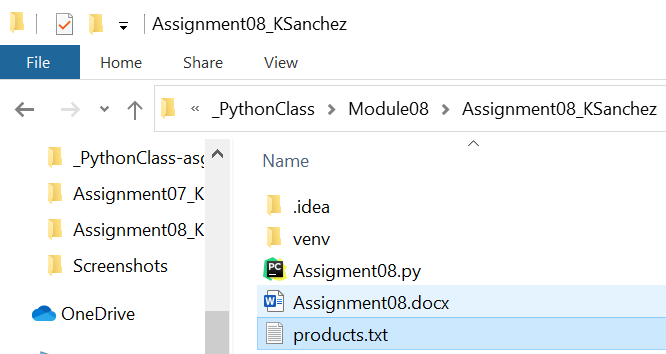


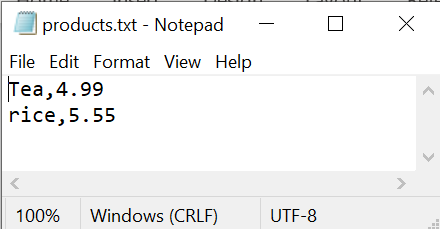
***Figure 5 : PyCharm Testing***



***Figure 6: Command Terminal Testing***

Finally, I checked that the text document was saved in my assignment folder (Figure 7).





***Figure 7: Verifying Data File Exists***

# Summary

After modifying the starter file and using the assignment answer to complete this assignment, I reflected on what I had learned. I not only gained exposure to objects and classes, but now have a better understanding of how they can be incorporated into a single program. It was difficult to fully understand the content with portions explained in notes or videos, but helps when I can see it applied together. I will keep this in mind for future projects and continue looking for ways to enhance my ability to learn and apply new concepts.