

# Classes and Objects

## Introduction — What a Challenge!

This module covered scripting custom classes and object-oriented programming. As I worked through the module content, I felt like I understood. This assignment was the logical next step from the exercises the previous week and builds upon what I also thought I understood. What a humbling struggle!! I was able to produce code that functions, albeit with a logic error. Admittedly, my final code is lacking but time is of the essence and this is how we learn — through practice. I read, read, and read and did the same thing expecting different results. That said, I made some custom classes to handle data, file processing tasks (reading from and writing to a text file), and to handle the input and output of data from a user.

## Personal Pain Points

To err is to be human. One very obvious, early mistake I made was not including the parentheses in the statement that creates an instance of a class. For instance:  
`testObj1 = class which should be typed testObj1 = Class()`. Since your eyes get tired combing and debugging your way through non-functioning code willing the answer to reveal itself, it is easy to miss those errors of omission. Luckily, I was able to find the answer and move on from that error.

As my script grew in complexity, I started forgetting things I knew as I worked to track the logical flow of information and lost sight of fundamentals — like remembering how the write method functions. I kept triggering errors by trying to write an object to file instead of first converting that object to a string before writing it to file.

The next pain point was validating attribute data in a class. I knew I wanted to ensure the value passed to an attribute was of the right type but I was having trouble sorting that validation piece. I eventually got there, albeit in simplistic and an unsophisticated way. See Code Snippet 1 below for my basic attribute validation of a class attribute called

product\_price.


```
@product_price.setter
def product_price(self, value):
    if float(value) > 0:
        self.__product_price = value
    elif float(value) <= 0:
        raise Exception("The price must be greater than zero(0).")
```

Code Snippet 1

## Final Code

### PyCharm

Figure 1 shows the script running in PyCharm. You can see the logic error occur when the user selects the option (1) that displays current data. What it shows is the data in the text file at the time the program starts but it does not show data as it is added to the list. However, when you see the result of the program in Figure 3, you can see that an appended list is written to file. To demonstrate this I added two different products when I ran the script in PyCharm and Terminal.



```
Run: Assignment08
/Users/corpuscorvus/Documents/_PythonClass/Assignment08/venv/bin/python /Users/corpuscorvus/Documents/_PythonClass/Assignment08/Assignment08.py

Menu Options
1) See Current Data
2) Add Data
3) Save & Quit

Which option would you like to perform? [1, 2, or 3] - 1

-----The current products in the list-----
Test 2, 0.35
Quantum, 9999.00
-----

Menu Options
1) See Current Data
2) Add Data
3) Save & Quit

Which option would you like to perform? [1, 2, or 3] - 2

Enter a product name: Nuka cola
Enter the price of that product: 1000

Menu Options
2) Add Data
3) Save & Quit

Which option would you like to perform? [1, 2, or 3] - 1

-----The current products in the list-----
Test 2, 0.35
Quantum, 9999.00
-----

Menu Options
1) See Current Data
2) Add Data
3) Save & Quit

Which option would you like to perform? [1, 2, or 3] - 3

Data saved. Goodbye!

Process finished with exit code 0
```

Figure 1. Script run in PyCharm with logic error.

## Command Prompt

Figure 2, shows the script running in Terminal.app. You can see the menu presented to the user, the script prompting the user for a menu choice, the display of the current data in the list (read from a file), user input capturing a new list item, and finally the script saving and closing the list. You can see the text file result in Figure 3.

```
The default interactive shell is now zsh.
To update your account to use zsh, please run `chsh -s /bin/zsh`.
For more details, please visit https://support.apple.com/kb/HT208050.
[corpus-corvus:~ corpuscorvus$ cd /Users/corpuscorvus/Documents/_PythonClass/Assign]
gment08
[corpus-corvus:Assignment08 corpuscorvus$ python3 Assignment08.py]

    Menu Options
    1) See Current Data
    2) Add Data
    3) Save & Quit

Which option would you like to perform? [1, 2, or 3] - 1

_____The current products in the list_____
Test 2, 0.35
Quantum, 9999.00
Nuka Cola, 1000.00
-----

    Menu Options
    1) See Current Data
    2) Add Data
    3) Save & Quit

Which option would you like to perform? [1, 2, or 3] - 2

    Enter a product name: bobblehead
    Enter the price of that product: 15

    Menu Options
    1) See Current Data
    2) Add Data
    3) Save & Quit

Which option would you like to perform? [1, 2, or 3] - 3

Data saved. Goodbye!
corpus-corvus:Assignment08 corpuscorvus$
```

Figure 1. Assignment script run in the command prompt (Mac Terminal)

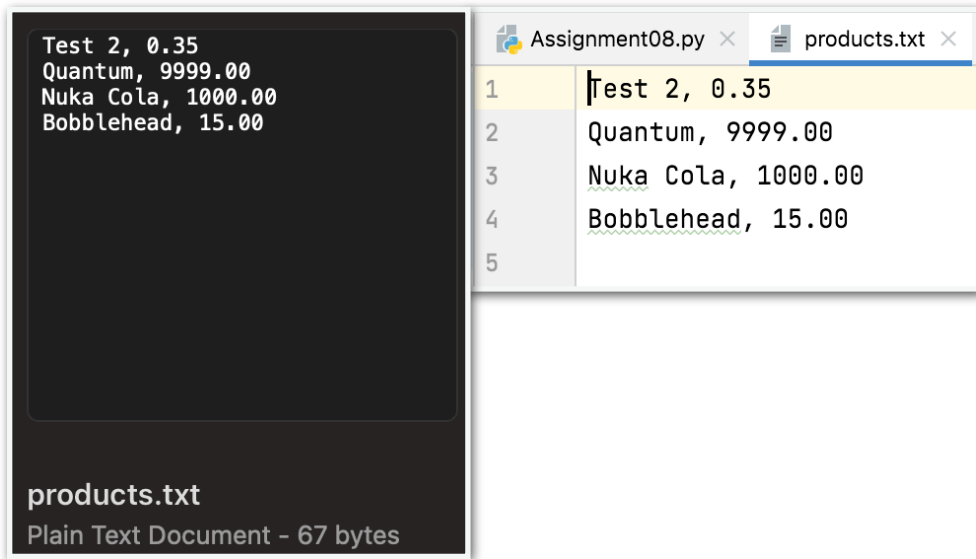


Figure 3. The script results written to text file as visualized in Finder (left) and PyCharm (right).

## Summary

This module was difficult. I rose to the challenge and worked hard on my code but unfortunately did not get to a fully functioning code in that I am not able to display the current state of the list correctly. It displays a snapshot in time only. I know this is occurring because I set a variable that is passed the value of a class instance from the read file portion of my script. This locks in the value. However, I am able to write and appended list to file.

There were some bright spots despite all the frustration. I learned that you can split your PyCharm window between scripts for comparison. I found this really helpful as I made scratch code for each portion of my script. I tested out the logic of the sections in a piecemeal fashion. My final code is a conglomeration of those tested sections. And, while I didn't sort out my issues with the current data display portion of the code, I did enjoy researching it and testing out code. Everyone solves problems differently and I found a way that I liked — breaking out each section of code into smaller scratch files and marrying them together.

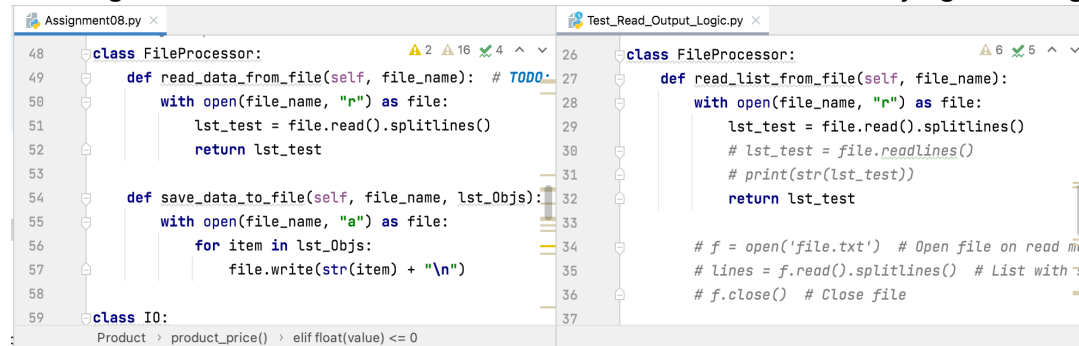


Figure 4. PyCharm's hand way to view two scripts side by side.