Eric Hoyle

02-20-21

Foundations in Programming; Python

Assignment\_06

# Introduction

In Module 6 I got my first introduction to the new concepts of classes, variable scope, defining functions and the use of arguments, and became better acquainted with SoC in programming. This week’s content was good because rather than spending hours struggling to get my programming to work as I intended, I was instead able to focus on structure and the interplay of classes and methods.

# Creating the Program (Refinement)

This week, the program was more or less handed to me and all I needed to do was to condense the applicable code in the main into functions and place them in their appropriate class; a seemingly simple task. However, it became clear rather quickly that deciding where to break code into function based on class is not always straight-forward. Indeed, some sections of code that I desperately wanted to keep together in a logical block spanned two or more classes. Take the section of code used to append intake a new CD into the inventory:

Text

Description automatically generated

Figure 1-Menu selection ‘a’ code in CDInventory.py.

Although keeping this block intact is tempting it clearly falls into two classes. Lines 138-141 belonging to the IO class while 142-143 belong to the data processing class. Subsequently I broke this block into two functions; add\_cd() and table\_append()), the former falling into the IO class and the latter the data processing class.

Similarly, the code block for deleting an item from inventory was consolidated into a function I defined as cd\_remove() and placed into the Data Processing class.

Text

Description automatically generated

Figure - The cd\_remove function of Data Processing Class

Lastly, the code block for writing data to the file was also placed into a the defined function “write\_file()” and in the class FileProcessor. Again, here it was a little difficult to decide whether this block should be broken down further into a function within DataProcessing and a function in the class FileProcessor because the data is being manipulated prior to writing to the file. But, this would seem to cause more discontinuity than is warranted so I left it intact.

Text

Description automatically generated

Figure - write\_file function of class FileProcessor

# Summary

Overall, the scope of this assignment seemed more manageable than last week but, as always, the Devil is in the details. Once I started dissecting the code and parsing it into functions it became difficult to know where to stop. Does each print statement need to be contained in an IO class function? That seems a bit perilous and overwrought, so I left much of the remaining starter code unaltered. I also made some minor adjustments to presentation of the inventory just because it seemed worthwhile to me.

Additionally, the concept of variable scope and how to best handle transferring output from one function to input in another remains a little elusive to me but so far everything seems to be working as intended so I think I’m safe for now.

In the end I see how advantageous compartmentalizing code onto functions and classes can be for purposes of maintenance and readability. At some point I imagine it makes much more sense to build some programs by defining classes and functions first and then building the main, supposing one has a vision for the scope of the program in advance and it isn’t a meandering through the woods as so far it has been for me.

Here it is in Spyder:

Text

Description automatically generated

Figure 4-In Spyder

And in Terminal:

Text

Description automatically generated

Figure 5-In terminal

# Appendix

Source code contextualization from [Planetb’s](http://www.planetb.ca/syntax-highlight-word) [[1]](#footnote-1) web page.

Text

Description automatically generated

A picture containing text

Description automatically generated

Graphical user interface, text, application

Description automatically generated

1. Retrieved 02-05-21 [↑](#footnote-ref-1)