# Introduction

For assignment 8 this week we were given the task of modifying starter code provided by our instructor. Among the list of ‘TODO’S’ we were tasked with, we were to add code to specific classes, add functionality to allow a user to save and load to a .txt file and lastly add code to the main body of the program in addition to error handling. Additionally, we were also tasked with updating docstrings as well as ensuring the program functioned correctly. Below are the details taken to complete this week’s assignment.

# Assignment 08

With last week’s assignment still fresh in memory, I started week 8 with laser focus. I began by completing all labs beforehand as to better understand the assignment and to utilize my time more effectively. I didn’t want to waste a single moment focusing on specific non-essential elements of the code vs overall program functionality.

Assignment 8 was largely centered on fulfilling several TODO items - such as creating code for the classes: CD and FileIO, adding code to the presentation of the separation of concerns(SOC) as well as the body of the program, then finally ensuring the program worked .

At first, I was apprehensive as the starter code was very bare. However, given that the task was only to read and understand the pseudocode, then add code to make the application work, I was quickly able to conquer my insecurities and got straight to work.

I began by implementing what I’d learned (or what I believed I’d learned) about classes in week 8. I created code to fulfill the requirements of the TODO listed under the CD class; the requirement was that we add code so the inputs: cd\_id, cd\_title, cd\_artist would be stored in the class CD data. Just as an aside: I’m realizing just how abstract coding is.

Trying to visualize how all the moving pieces of a program mesh together without breaking it, is at times, overwhelming. Moreover, I’m learning how intimidating it is (or will be) to create a program from nothing without instruction. Thankfully, the pseudocode was easy to understand in terms of what needed to be done for class CD().

With the instructions given I wrote out the following code for Class CD().

|  |
| --- |
| Class CD ():  def \_\_init\_\_(self, obj):    self.add\_row = obj    def add\_row (cd\_id, cd\_title, cd\_artist, table):    try:  int\_id = int(cd\_id)  dicRow = {'ID': int\_id, 'Title': cd\_title, 'Artist': cd\_artist}  table.append(dicRow)  except ValueError:  print('\n' + 'Please re-check entry for ID - a non-integer number was entered or the position was left empty.' + '\n') |

For the code shown above (and I apologize if I’m reading this incorrectly) I started with the class: CD(), then defined the constructor and created its attribute: self.add\_row = obj. After, I created a function, which was easy-going as the code used has been repeatedly applied since assignment 5.

I updated the variables and the function arguments, so they reflected the requirements of the TODO item. Then I passed them off as the properties of class CD. Lastly, with the variables changed, I applied my instructor’s advice and updated my previous exception, so it incorporated the dicRow table appendage in the try, except error handling. Doing this, the program will not crash whenever an input of string or 0 are accidentally passed in the ID call.

The next tasks given were to add code to process data from a file and to a file, in the class FileIO. I’ll be completely honest and state outright: I’m uncertain that I did this correctly. Nevertheless, I rather try and fail and learn than not even attempt to understand.

I started off with what I already knew about saving and loading from .txt files. Since we already had code readily available from our prior assignments, I took and applied the appropriate code blocks for writing and reading to .txt files and added them to the class FileIO.

|  |
| --- |
| class FileIO:  def \_\_init\_\_(self, obj):    self.save\_inventory = obj  self.load\_inventory = obj    # @property  def save\_inventory(self, table):  objFile = open(strFileName, 'w')  for row in table:  lstValues = list(row.values())  lstValues[0] = str(lstValues[0])  objFile.write(','.join(lstValues) + '\n')  objFile.close()  # @property  def load\_inventory(self, table):  table.clear()  objFile = open(strFileName, 'r')  for line in objFile:  data = line.strip().split(',')  dicRow = {'ID': int(data[0]), 'Title': data[1], 'Artist': data[2]}  table.append(dicRow)  objFile.close() |

Then, as shown above, I followed more or less the same process as I did with class CD(): and defined the constructor and its two attributes, in this case: self.save\_inventory = obj and self.load\_inventory = obj then imported the two code-blocks for saving and loading inventory, updating the function call name to reflect the attributes of the class. Again, I will state: I’m uncertain if I did this correctly as the TODO item called for the save / load functionality of the program to be methods and not merely functions. Moreover, when either load or save call is made in the program, bizarrely the program doesn’t accept the property. BUT with those commented out, the program runs fine.

Lastly, for the remainder of Assignment 8, the TODO items were largely centered on adding functionality to the class IO as well as the main body of the program.

For this, I largely imported code from the prior assignments, adjusted the language to factor in that the code was not calling for delete functionality of dicRow by omitting that code block as well as updating the print statements and lastly read through the imported code, updating function calls to reflect the changes to the classes, method calls and functions as to allow the program to work.

In addition, I edited the pseudocode to reflect that the code had been changed so it read better (if that makes sense), added my contribution to the script and error checked the program to make sure crashing was not feasible. And as you’ll see below the program runs perfectly:

![Graphical user interface, text

Description automatically generated]()

![Text

Description automatically generated]()

**Fig 1. – CMD and Spyder showing execution of Assignment 8 code.**

# Conclusion

With the code running and all TODO tasks completed, I believe all prerequisites have been fulfilled to mark this assignment completed. I’m certain there are still things which will need to be improved upon or which due to my still being relatively new to the more complex aspects of Python will have overlooked, nevertheless, I’m looking forward to hearing the feedback to this assignment so I can continue to grow as a programmer. Looking forward to the start of week 9 and two more weeks till the conclusion of the intro to Python course. Extremely excited.

# Appendix.

**Fig 1. – CMD and Spyder showing execution of Assignment 8 code.**