Matthew Poehler

February 21, 2021

Foundations of Programming: Python

Assignment 06

CD Inventory Script

Introduction

In this document I will demonstrate the steps taken to modify the CD Inventory script provided by Professor Biesinger by implanting the new information from the Module 06 material of the Foundations of Programming: Python course. This will include brief overview of functions and classes as well as doc strings. These are the new concepts that make up the modifications to be made in the CD Inventory script. I will conclude with images of the completed script executing specific actions in Spyder IDE and Anaconda Prompt

Functions

According to the material in both Module 05 and Module 06, functions are a group of statements that when defined and given a name, can be executed when the defined name is called in the program. Functions improve script organization and help with the readability of larger programs by isolating the data processing inside the functions, allowing the function name to be used multiple times rather multiple lines of code being repeated throughout the program. Listing 1 is an example of a function being defined in a script.

```
14  def eqResults(value1, value2):
15      summ = int(value1 + value2)
16      diff = int(value1 - value2)
17      prod = int(value1 * value2)
18      quot = int(value1 / value2)
19      return (summ, diff, prod, quot)
```

Listing 1 - Function used in LAB06-B.py from Module 06 material, demonstrates how a function can look in a script

Parameters (Arguments)

The purpose of functions is to do some type of processing when called in the main body of the script. This requires some data to be passed into the function. Parameters, or arguments as they are more commonly called according to the material in Module06, is the term for the values that are used to pass in data in the function to then be processed. Parameters mostly come in the form of variables according to the material in Module 06. Listing 2 demonstrates what this looks like.

```
29  results = eqResults(intNumA, intNumB)
```

Listing 2 - Line from LAB06-B.py that contains the function 'eqResults' with two variables being passed in as the argument

Return Values

After an argument or arguments are passed in, the function processes the statements contained inside and then spits out the desired results for the main body of the script to use. According to the material in Module 06 these results are called return values, which can be consumed immediately in the script or stored in variable to be used later in the program if need be. Listing 1 shows that in the “eqResults” function the return values is a tuple of data. Listing 2 shows that the return values are being stored in the “results” variable.

Doc Strings

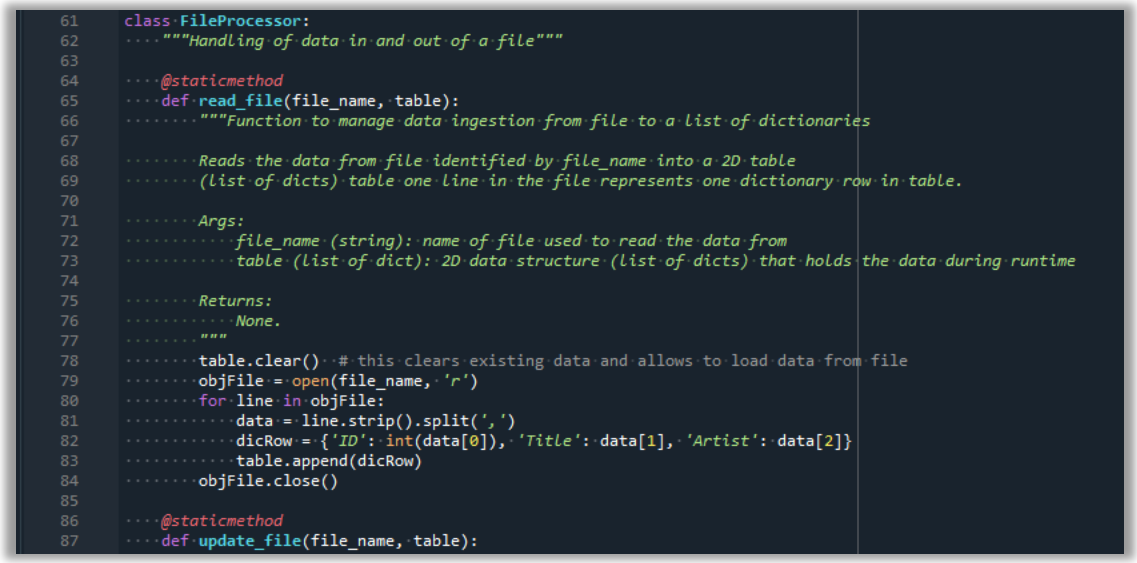
According to the material in Module 06 it is a common practice to include a description of each function that is in a program. This description is known as a docstring, Docstrings are headers at the beginning of the function and are contained within triple quotes (“”” “””). Listing 3 shows what this looks like inside a program.

```
23      def add_Inv(data_list):
24          """Function to add user data into 2D list of dictionaries as inventory management
25
26          Args:
27              datalist (return value of get_data() function): contains the user input as a list
28
29          Resturns:
30              dicRow (dict): user data in a dictionary
31              lstTbl (2D list of dict): append 2D list of dicts with recent user data
32          """
33          intID = int(data_list[0])
34          dicRow = {'ID': intID, 'CD Title': data_list[1], 'Artist': data_list[2]}
35          lstTbl.append(dicRow)
36          return lstTbl
```

Listing 3 - Function in CDInventory.py script, demonstrating the use of docstrings

Classes

Similar to how functions organize lines of code, Classes organize functions depending on the contents of each function. According to the material in Module 06 classes are way of grouping functions, variables and constants. Listing 4 contains an image of what this looks like in a program.



```
class FileProcessor:
    """Handling of data in and out of a file"""

    @staticmethod
    def read_file(file_name, table):
        """Function to manage data ingestion from file to a list of dictionaries

        Reads the data from file identified by file_name into a 2D table
        (list of dicts) table one line in the file represents one dictionary row in table.

        Args:
            file_name (string): name of file used to read the data from
            table (list of dict): 2D data structure (list of dicts) that holds the data during runtime

        Returns:
            None.
        """
        table.clear()  # this clears existing data and allows to load data from file
        objFile = open(file_name, '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()

    @staticmethod
    def update_file(file_name, table):
```

Listing 4 - A portion of a class in the CDInentory.py script.

CD Inventory Script

The assignment tasked for this section of the Foundations of Programming: Python course is to modify the CD Inventory script provided by Professor Biesinger. These modifications include adding functions and the appropriate information attached to them to complete the program that asks a user to input the ID, CD Title and Artist name of any album of their choosing. The program gives the option to add and delete from the inventory as well as gives the user the ability to load from the text file and save to the text file. Listings 3 and 4 are portions of this script. Figures 1, 2 and 3 are images of the program being executed in Spyder IDE and Figures 4 is the execution in Anaconda Prompt.

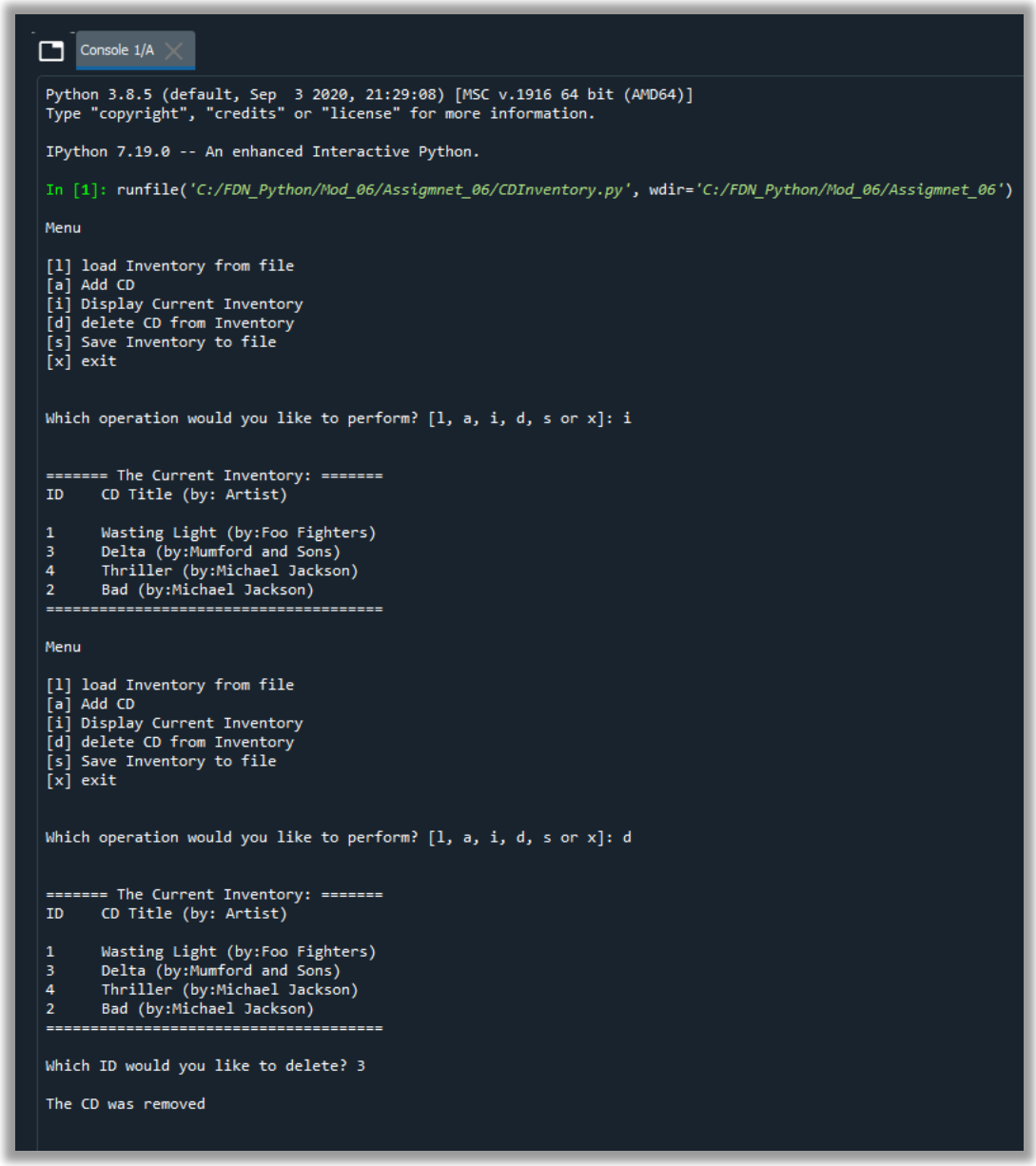


Figure 1 - CDInventory.py being executed in Spyder IDE, displaying the inventory and deleting an entry

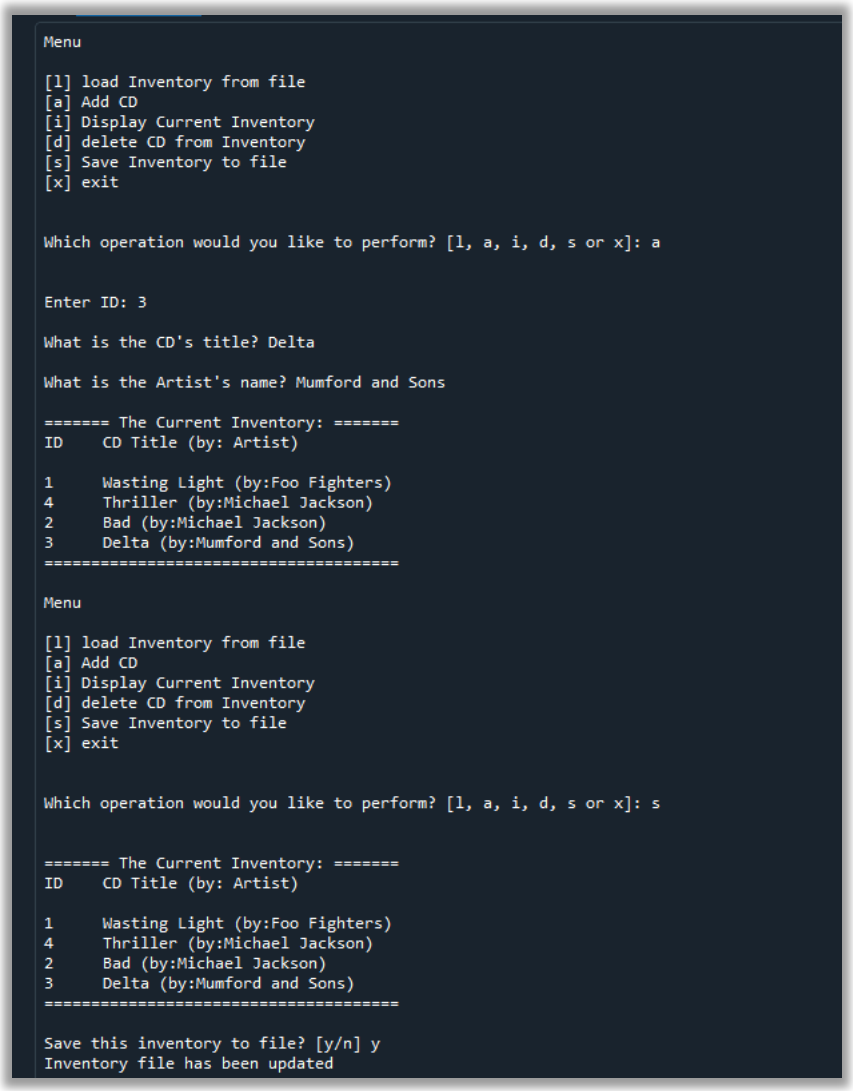


Figure 2 - CDInventory.py being run, adding an entry and saving updated inventory to a text file

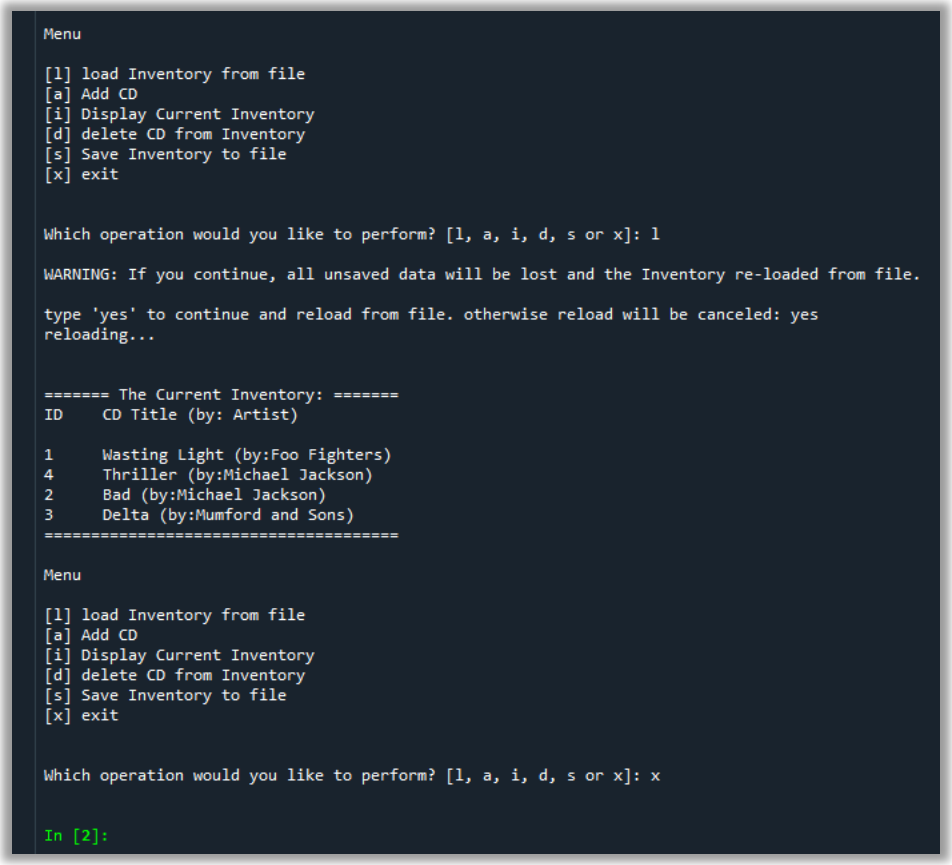


Figure 3 – CDInventory.py being run, loading inventory from text file and exiting

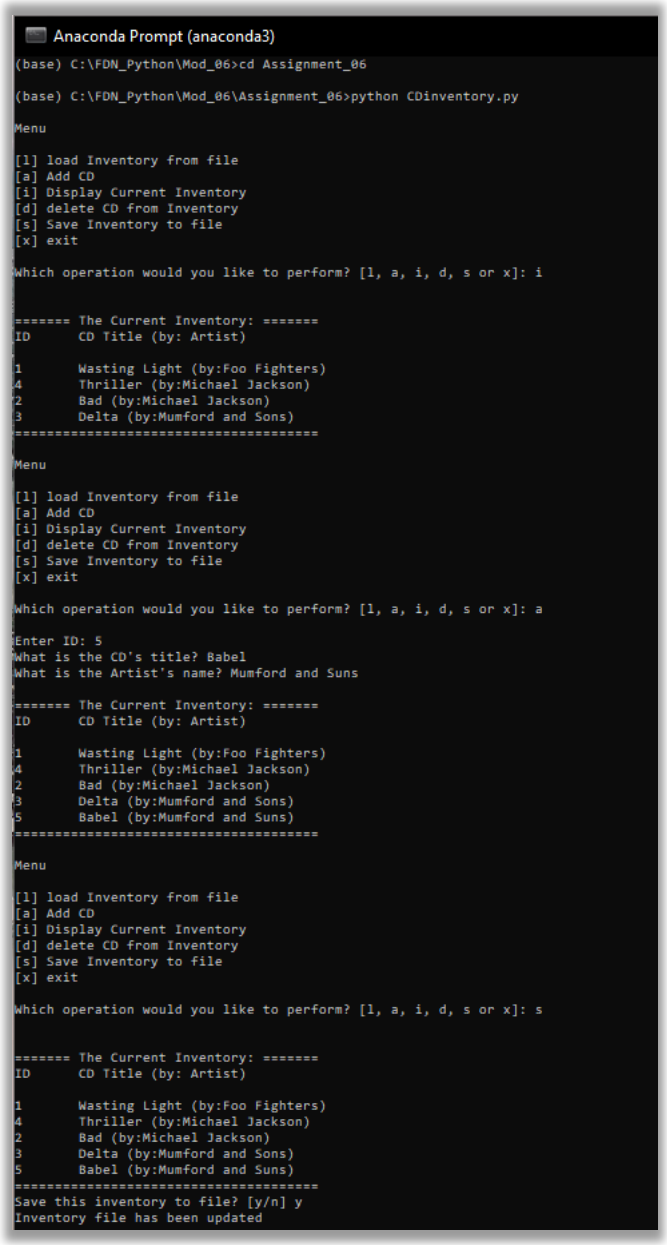


Figure 4 - CDInventory.py being executed in Anaconda Prompt

Summary

Functions are very helpful in organizing the inner workings of a program but take practice to understand fully and I am excited to continue to add to the versatility of them throughout the rest of this course. The material in Professor Biesinger's Module 06 is a great source of information to help in the understanding of these new additions to the CD Inventory Script. I am really looking forward to the next modules in the course.