**Name:** Thomas Caetano

**Date:** 12-12-2021

**Course:** PYTHON FUNDAMENTALS – UW (IT FDN 110 B)

INTRODUCTION

Lab09\_A focuses on the applicability of python modules and how it’s classes, functions and methods are called within the body of a main programme. The main assignment consists of tying all the code together. The modules are imported in the beginning of the code and they serve to replace several phases or methods of the code by referencing the classes and functions inside the modules.

GitHub Link: <https://github.com/tcaetano1970/Assignment_09>

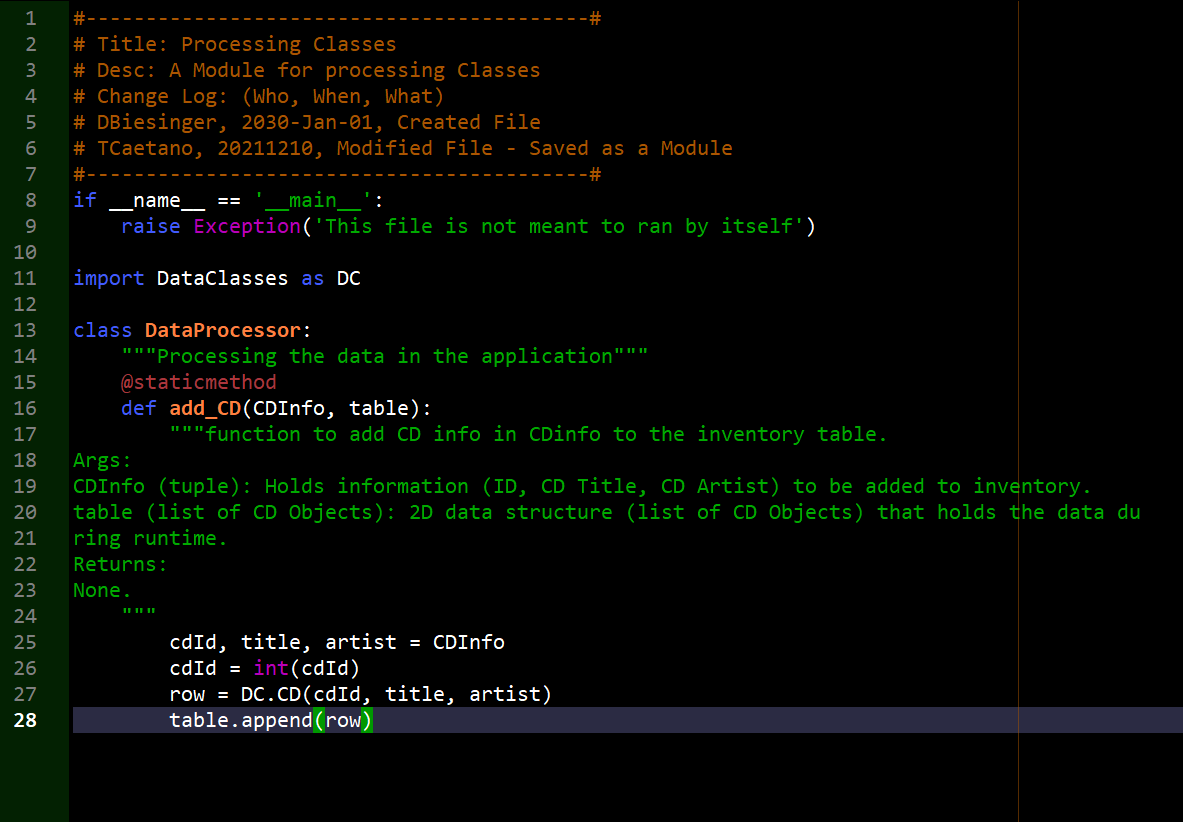
LAB 09-A  
In this Lab, you’ll create multiple modules that work together. We’ll re-implement the functionality of last module’s  
assignment using modules. You’ll also add a test harness to test your modules.  
Note: You do not need to type all the code. You can re-purpose code from last week’s assignment.  
Important: create a sub-directory Mod09\_A in your \_FDProgramming directory for this. (will get important in the next  
step!)

1. Create a script called ‘TestHarness.py’  
   2. Create a script module ‘DataClasses.py’  
   3. Add the code in listing 6 to the DataClasses.py script.  
   4. Add code to your TestHarness.py script to test your DataClasses.py script  
   5. Create a script module ‘IOClasses.py’  
   6. Add the code in listing 7 to the IOClasses.py script  
   7. Add code to your TestHarness.py script to test your IOClasses.py script  
   8. Create a script module ‘ProcessingClasses.py’  
   9. Add the code in listing 8 to the ProcessingClasses.py script  
   10. Add code to your TestHarness.py script to test your ProcessingClasses.py script  
   11. Create a script ‘CDInventory.py’  
   12. Add code to your script to run the application (former main section)  
   13. Ensure that all test cases in the test harness work  
   14. Ensure that all functionality in the application work.

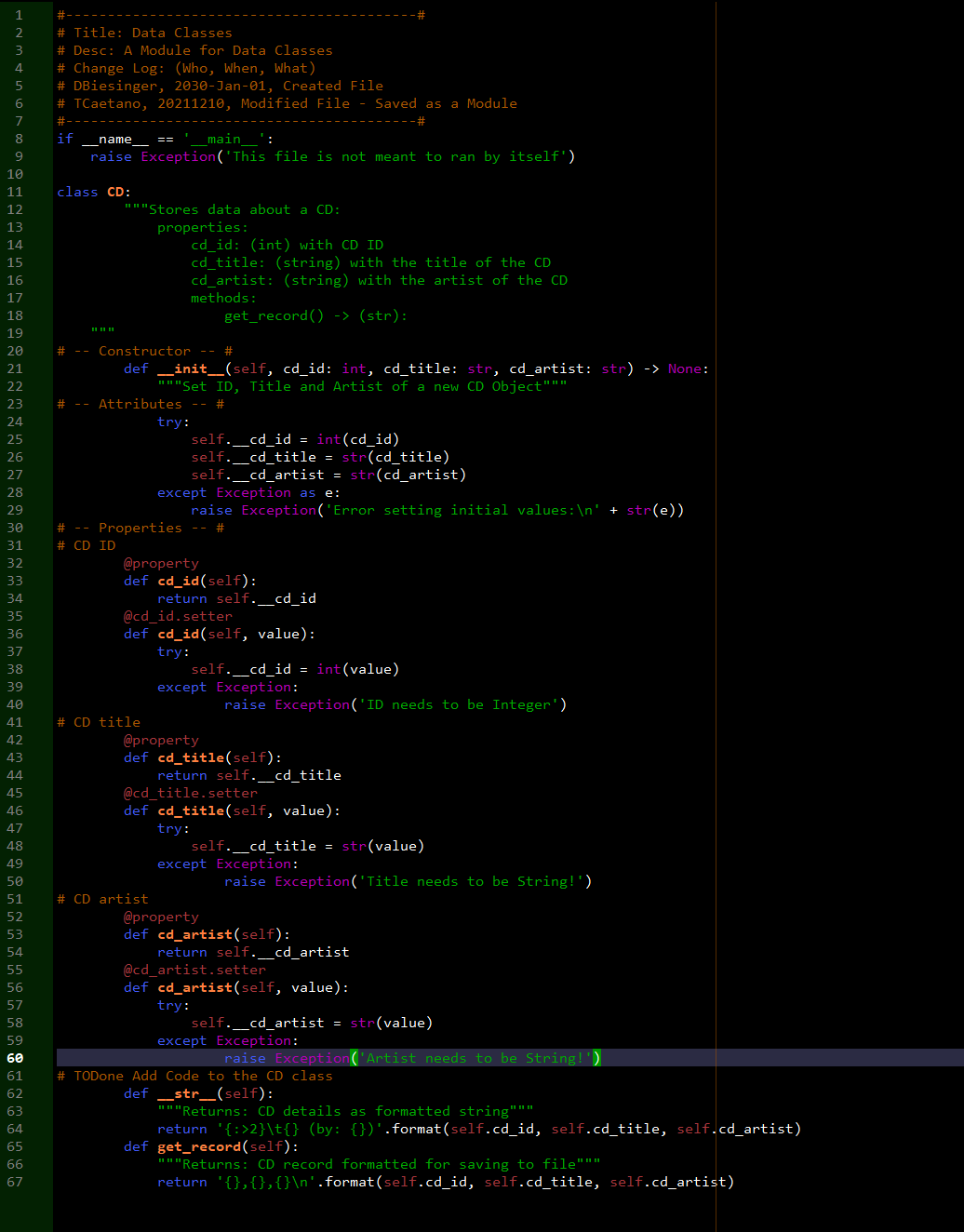
Modules Setup

*I used the /desktop as the directory to manage all files.*

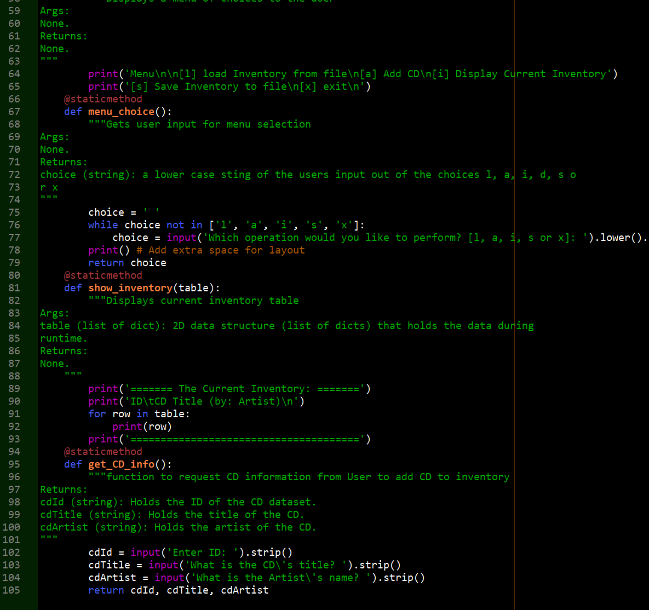
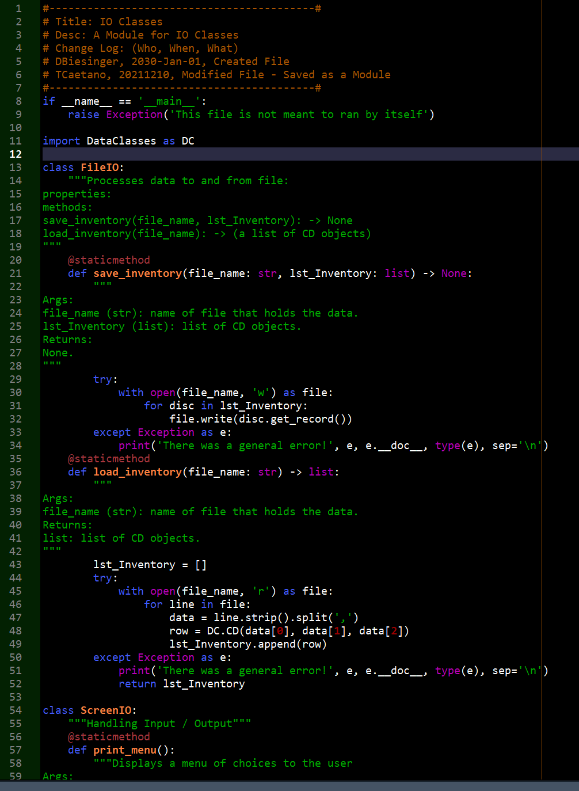
ProcessingClasses Module



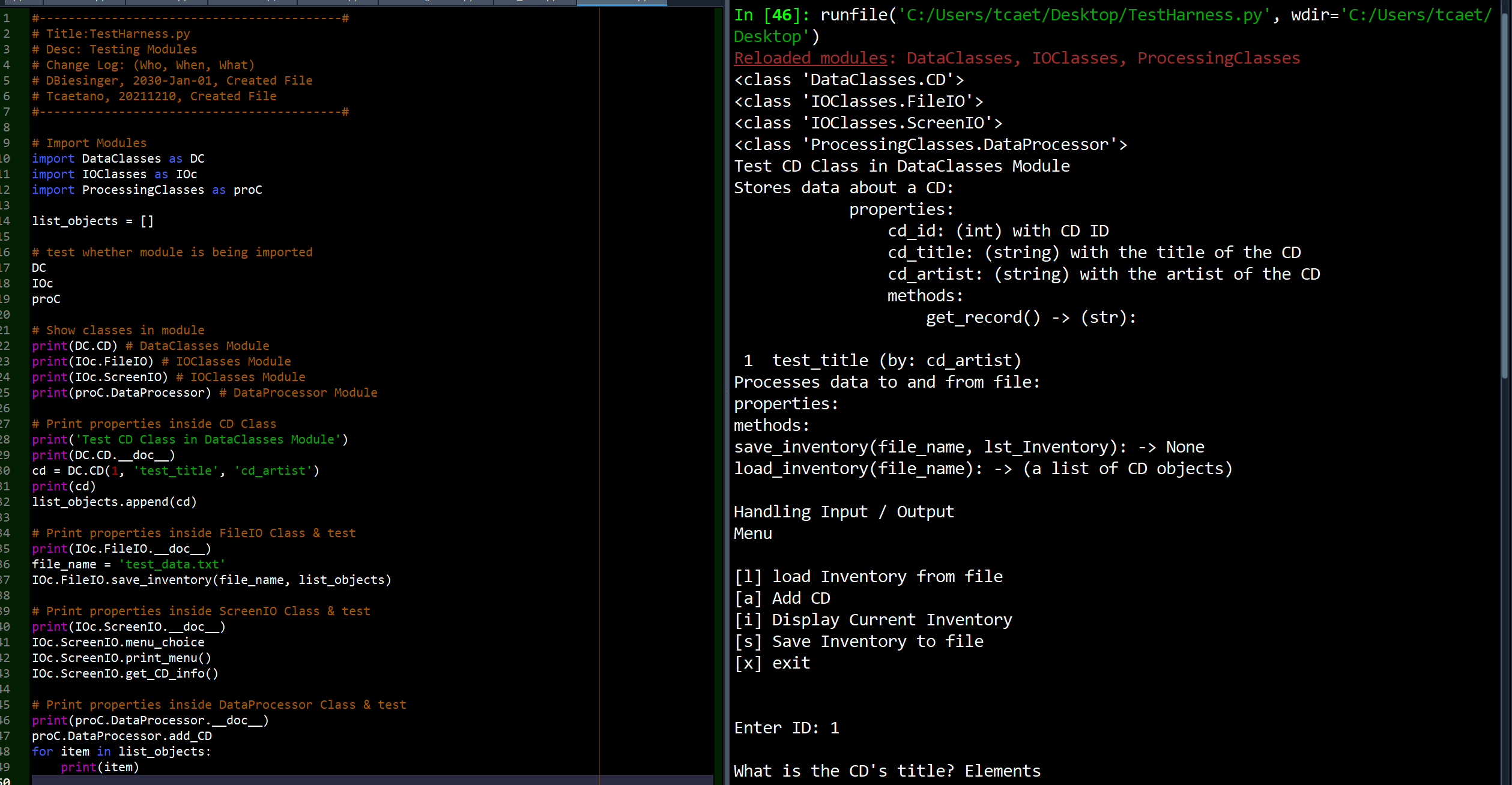
DataClasses Module

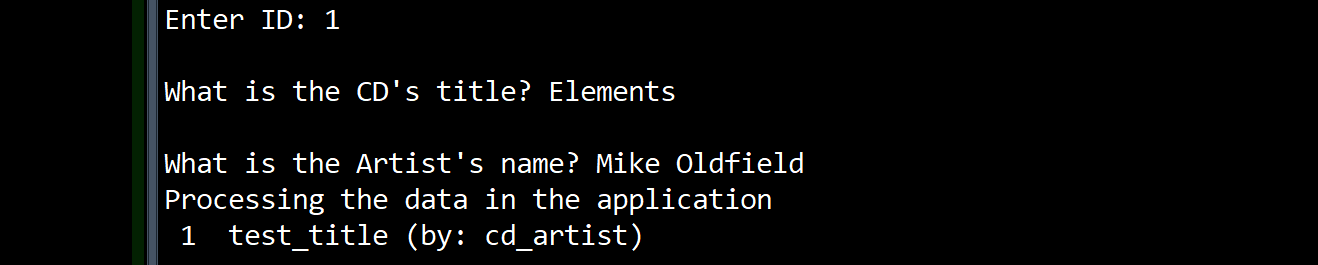


IOClasses Module



TestHarness Programme to test modules a





Graphical user interface, application, Word

Description automatically generated

It’s interesting to learn that python modules are nothing more than .py files that are called into a programme through the IMPORT command. I was under the impression that modules were something far more complicated. Modules contain classes and functions that may be called by using the module’s name followed by a function or command inside it. A module may also be assigned an alias after ‘importation.’ This makes it simpler to write code.

In the above example three modules are created and assigned names ending with a .py file extension.

The TestHarness.py programme then imports these three modules with aliases in the beginning of the programme. Notice how the module names are long and once they are assigned aliases, they become much simpler and easier to type.

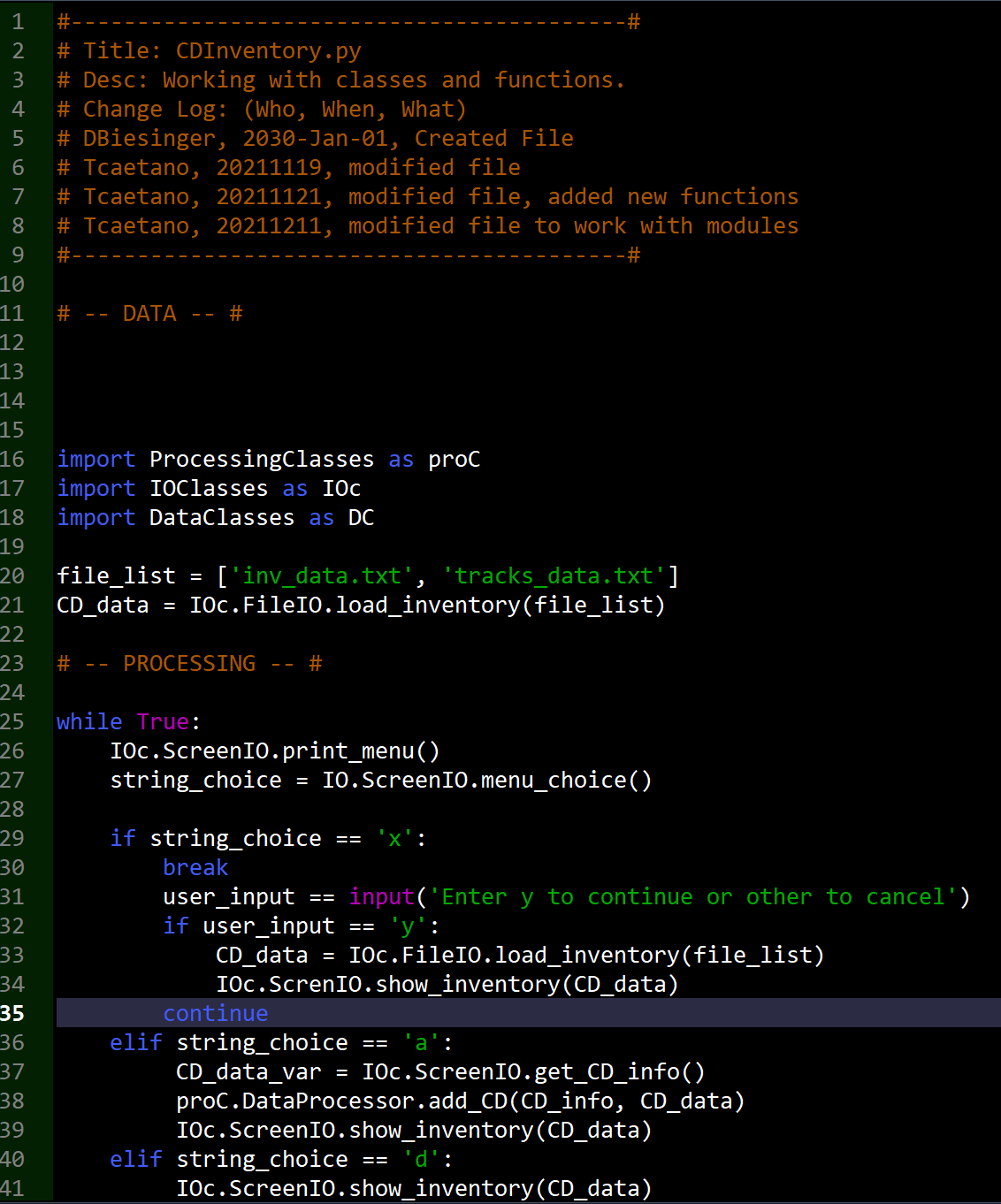
Importing the modules creates a reference to the three programmes lying outside the the TestHarness code albeit still inside the same directory and path. It is crucial to have the modules inside the same PATH as the original programme.

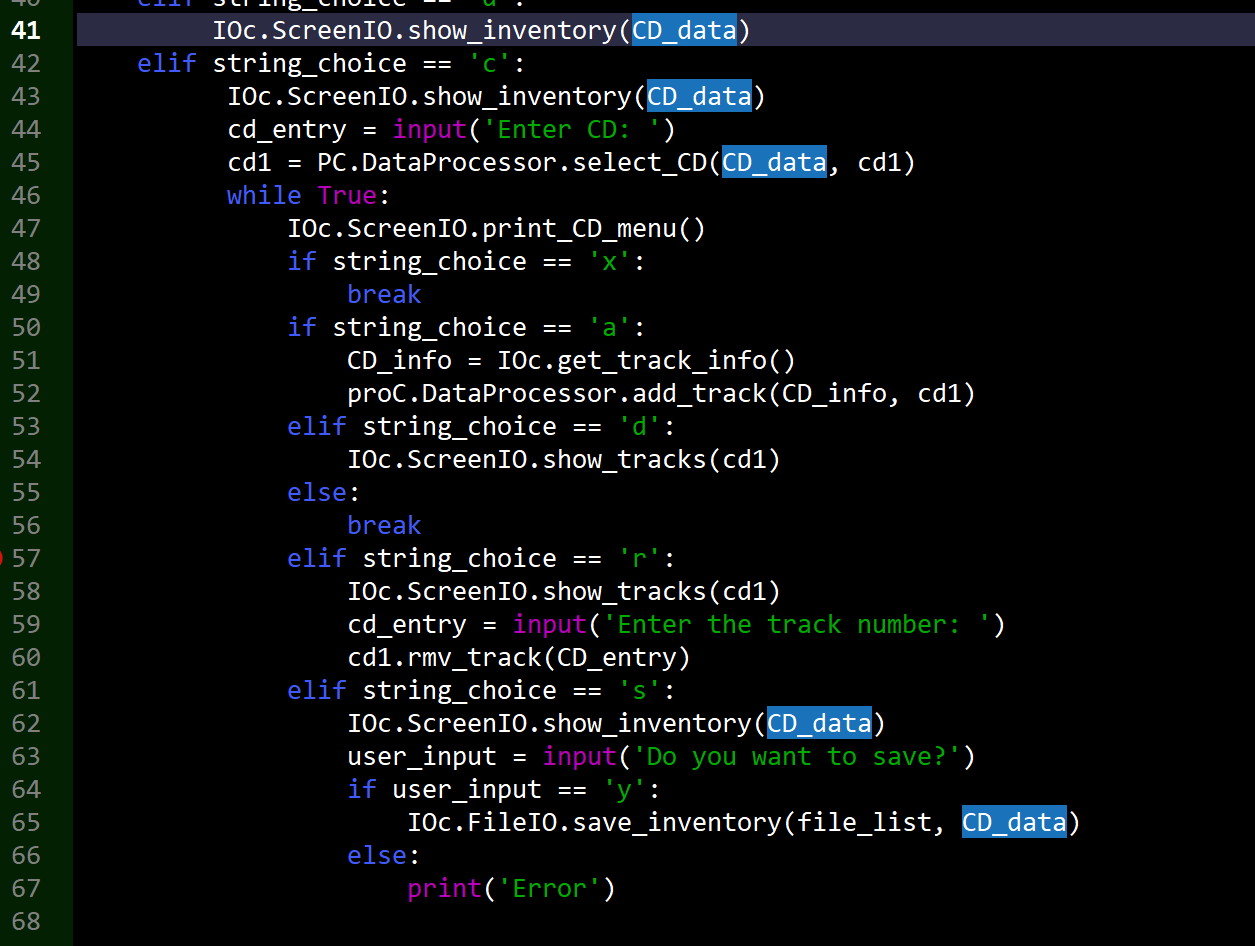
At a very basic level I wanted to call the modules by themselves to show that they are successfully imported.

The method used to call a function or class inside a module is by using the name of the module followed by that function, for example, IOc.FileIO calls the function FileIO inside the IOc module – to further move in the hierarchy order, meaning calling a method inside the function itself will require another addition to this line; IOc.FileIO.Save\_Inventory().

At the end of this code, I was able to call code from inside the module that actually displays the menu of options in the CDInventory code and ask for user input. Given that I did not want to rely on appendix solutions to bail me out I decided to leave it as it is now. Going through course literature was helpful but I am still having a hard time bringing it all together.

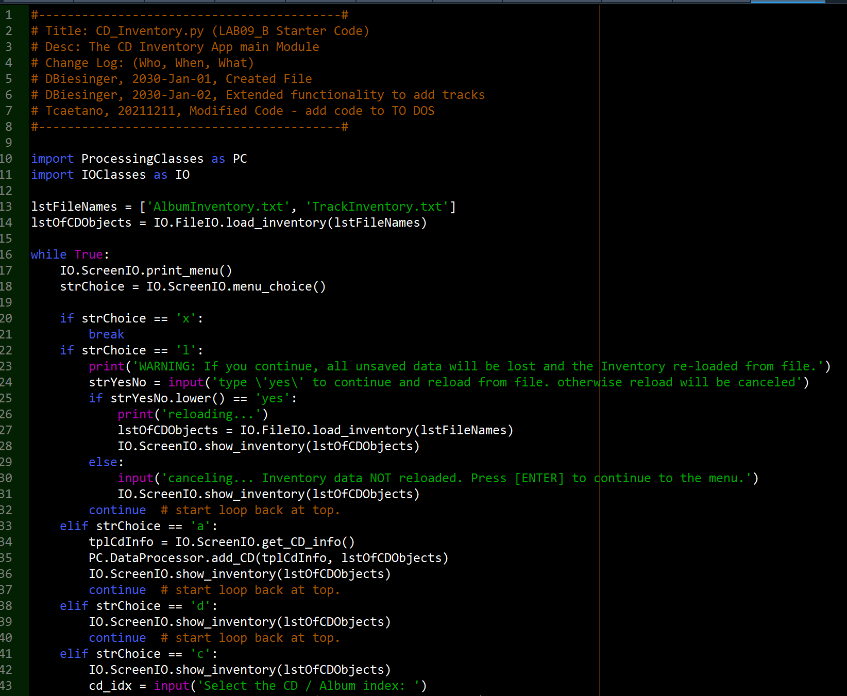
CDInventory.py





This is a challenging one and it becomes the assignment itself. The main functions in this code must be replaced with the code in the modules or by calling the functions in the modules which perform the same tasks. I understand the concept and logic behind it but having difficulty in implementing this namely which sections of the module should be called. Will continue working on this.

LAB 09-B:  
I included a LAB 09-B starter code. Solve the TODOs to make the application work as described above.  
(A sample solution is in the appendix, peek at your own risk)  
The included TestHarness should produce an output like this:



Lab09\_B continues as the main code assignment\_09

