# Natalie E. Tolliver

6/14/2020

ITNFDN 100A

Assignment 09

<https://tollivne.github.io/IntroToProg-Python-Mod09/>

Separating Concerns into Modules

## Introduction

The purpose of this script is to read data from an existing employee database, prompt the user for a new Employee ID, First Name, and Last Name and add it to the database. The user then has the option of displaying the current data that is in memory, adding more items, saving the data, or exiting the program. The user experience is quite similar to that of the last script (Assignment 8). The primary difference is not so much in the user experience as it is the methodology used in the script. Whereas previously, the functions and classes were placed in the main code, for this script, the functions and classes are separated into different files and the main program is used to call these “modules.” The modules are “imported” in the beginning of the main program and then when they need to be used, the syntax uses the module name (or shortcut name) and then the name of the function and then the name of the field, if referencing an individual variable. The output of the program is shown running in both PyCharm and the CMD window.

# Learning Objectives

The primary purpose of this exercise was to display knowledge of the following concepts.

## What is the difference between a class and module?

A module is a file that is separate from the main program that contains classes and functions. They help you to organize your functions and classes into “separation of concerns” for ease in reuse of code and in reading the code. For example, you can create modules to perform Input/Output (or Presentation), modules to perform Processing, and modules about Data.

## What is the "main" module?

The main module is the one that calls the other modules by “importing” the “modules.” They are then referred to later in the script by calling the modules using the module name and then the names of the classes or functions. You can create your own custom modules or download a myriad of available modules from the internet. Modules are called using the following syntax.

Import ModuleName

To call a stand alone function: ModuleName.FunctionName()

To create an object from a class: ObjectName = ModuleName.ClassName()

To call a function within a class: ModuleName.ClassName.FunctionName()

## What is the "\_\_name\_\_ " System Variable?

The \_\_name\_\_ is used to verify to the programmer that we are working with the main program and not one of the modules. It returns the name of the file that you are working with unless it is the interactive module and in that case, it returns the string “\_\_main\_\_”. The \_\_name\_\_ variable is often used along with error handling to test to see if it returns “\_\_main\_\_.” If it does not return main, then the program can print out a statement telling the user that module is not meant to be run as a standalone program. If it returns the name “\_\_main\_\_” then you can have the program print out a statement like “This is the main program.”

## How do you connect one module to another?

You connect modules by using the import statement in your main program. The Python interpreter then looks in the same folder as your main program to find the module by that name. If it does not find it there, then it looks in folders specified by the environment variables.

## What is class inheritance?

When a class refers to another class, it inherits all the parameters from that module. For example, if you create a class called “Person” with first name and last name fields, then you create a class called “Employee”, the employee class “inherits” the fields first name and last name from the “Person” class. When you inherit code from another class, it is called a parent – child relationship. It is represented in writing as shown below.

Child class -> Parent class

It can also be referred to synonymously as:

Derived class -> Base class

Sub class -> Super class

In the script in this project, the parent class is “Person” and the child class is “Employee.”

All classes automatically inherit from a class called “Object” whether or not you specify it. For this project I created an Employee class which inherited from the Person class as follows:

class Employee(Person)

## What are three types of UML diagrams?

Unified Modeling Language is a way of representing relationships between different modules. Three of the most common are

1. Class Diagrams which show the relationship between classes along with the properties and methods included in each class.
2. Use Case diagrams show how the software will be used by the human or other software. The person or software is referred to as the “Actor” which is represented by a stick figure of a human.
3. Composition Diagram shows the objects made from the classes and their relationships.

# Building the Test Harness

The test harness was built by importing the modules and performing very simple tasks such as reading in data and printing data. The purpose was to see if the imports and the calls function properly before proceeding to do more complex tasks with the functions and classes within the modules. The testing is done before creating the main module. We were given code for the different modules and had to link them properly, add code where indicated and run the test harness. In running it, I found the issues shown in Figure 1:

A screenshot of a social media post

Description automatically generated

Figure - Comparison between two different methods

The code on the right did not function properly. It called the “DataClasses” module and then called the “Employee” function. The Employee class has three fields. The first fiend is the hidden field employee\_id, the second field is first\_name, and the third field is last\_name.

This code in the right side of the figure is from the module called “Listing 11 shown below.

print(**"\*\*\*\*\*\*\* The current items employees are: \*\*\*\*\*\*\*"**)  
for row in list\_of\_rows:  
 print(str(DC.Employee(row).employee\_id)  
 + **","** + DC.Employee(row).first\_name  
 + **","** + DC.Employee(row).last\_name)  
print(**"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"**)  
print() *# Add an extra line for looks*

Running this code produces the error message shown in Figure 2:

A screenshot of a cell phone

Description automatically generated

Figure - Error message produced when running Listing11

The reason this code produces this error message is because it is calling the Employee Class of the DataClasses Module with only one parameter when it requires the other two, first name, and last name.

This code shown below is from the IOClases file and runs without error. Since it was not calling the Data Classes Module, it didn’t run into the same error.

print(**"\*\*\*\*\*\*\* The current items employees are: \*\*\*\*\*\*\*"**)  
for row in list\_of\_rows:  
 print(str(row.employee\_id)  
 + **","** + row.first\_name  
 + **","** + row.last\_name)  
print(**"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"**)  
print() *# Add an extra line for looks*

# Building the Main Module

The main module imported each of the following Modules:

1. IOClasses
2. DataClasses
3. ProcessingClasses

The main program was then edited to include error handling, a while loop to return to the main menu unless the user chooses to exit and calls to each of the modules. The code was also edited to add the user’s choice and to run an if/then/else loop depending on which option was chosen.

Once this problem was corrected, the program ran smoothly in PyCharm as shown in Figure 3.

A screenshot of a social media post

Description automatically generated

A screenshot of a social media post

Description automatically generated

Figure - Final code results run in PyCharm

Figure 4 shows the file running in the CMD window.

A screenshot of a cell phone

Description automatically generated.

Figure - Code results run in CMD Window

Summary

This script introduced the concept of using modules for “separation of concerns.” It is a concept we have been building upon in previous scripts, first by writing the code without any functions, then by taking out redundant operations and putting them into functions. Then we created objects so we could customize the data by adding formatting and error handling. This made it fairly straightforward to take the classes and functions out and put them into modules and call them using a main function. As programs get bigger, however, I can see the benefit of using UML to diagram which functions and classes are in which modules and how they tree up to one another. I am looking forward to the next assignment.