John D. Blake

22 May 2024

IT FDN 110 A Sp 24: Foundations Of Programming: Python

Assignment06

https://github.com/jdblake2357/IT-FDN-110-Work

HIGH-LEVEL Module Topic Title

Introduction

In Module 6 we covered the following:

- Functions
- Classes
- Separation of Concerns (SoC)

In Assignment06, we used these new tools to further refine the script that we have developed over the previous few modules.

The following report expands on these objectives.

Functions

A function is simply a blocks of code that performs a certain task, or multiple related tasks when called upon to do so.[1] Functions can pass parameters (not to be confused with arguments), and can return data to the main program. The basic structure of a function is as follows:

```
def Function_Name(Parameter1_Name: Parameter_1_Type, Parameter2_Name: Parameter2_Type, …)
     Local_Variable1 = value # Variable to be used only within the function
     .
     .
     <code>
     return Data_to_return_to_main_program
```

The function is then called, and parameter values (or arguments) are prescribed:

```
Function_Name(Parameter1_Name = Argument, Paremeter2_Name = Argument, …)
```

[1] While the tasks within a function don't necessarily HAVE to be related, it would be inefficient and confusing to have unrelated tasks within a function. Since we strive for efficiency and clarity, we simply will not entertain such an absurd notion.

Class

Related constants, variables and functions can be grouped into classes.[2] The structure of a class is as follows:

```
class Class_Name:
     Class_Variable1: Type = Initial_value # Variable used only within the class
     .
     .
     CLASS_CONSTANT1: Type = Value # CONSTANT used only within the class
     .
     .
     def Function1_Name(Parameter1_Name: Parameter_1_Type, Parameter2_Name: Parameter2_Type, …)
     .
     .
```

When calling on a function within a class, we create an instance. The basic structure is as follows

```
Instance_Name = Class_Name() # Creates instance
Instance_Name.Class_Variable = Class_Name.Function_Name(Parameter1_Value, Parameter2_Value, …)
```

Separation of Concerns (SoC)

Separation of Concerns is the practice of grouping a program's elements is related collections. It's like outlining a narrative to tell a coherent story, but for software.

[2] While the elements within a class don't necessarily HAVE to be related … just see previous footnote and substitute 'class' for 'function.'

Assignment

From a user perspective, the code that we develop in Assignment06 is functionally identical to that which we developed in Assignment05. However, once you look under the hood, the code is much different. And, while the actual number of operational (i.e. non-comment) lines is similar, the logical flow of the script is much more cohesive. (See script in Appendix 1.)

My interpretation of the Layer-Class-Function hierarchy is that it follows a basic outline form where we go from broad to specific. Thus, after the main header, and the global variable and constant definition, the 'body' of the script follows the following (outline) structure:

LAYER	**CLASS**	**FUNCTION**
Data	FileProcessor	read_data_from_file
		write_data_to_file
Presentation	IO	output_error_message
		output_menu
		input_menu_choice
		input_student_data
		output_student_data
		output_student_data_clean

In the Processing layer, we call the required functions, and thus the outline is as follows

LAYER	**CALL**	**WHEN CALLED**
Processing	FileProcessor.read_data_from_file	At script initiation
	IO.output_menu	Until termination
	IO.input_menu_choice	Until termination
	IO.input_student_data	At menu choice '1'
	IO.output_student_data	At menu choice '2'
	FileProcessor.write_data_to_file	At menu choice '3'
	IO.output_student_data_clean	At menu choice '3'

Novel Features

Most of the features in this script were 'cut-and-pasted' from the previous assignment, and structured within the Layer-Class-Function framework. added a function.

- Incorrect menu choice (lines 152 - 153 in script): This sends a message to the user directing a menu choice between 1 & 4.
- Clean data output (lines 208 - 224 in script): This returns a succinctly-formatted message informing the user of the enrollment data that was saved, and it is called immediately after the enrollment data is saved to the prescribed .json file (menu choice '3').

Verification

I first ran the script in the PyCharm shell (see Appendix 2), and then ran the script in the macOS shell. In both instances, the script ran per the Assignment 6 'Acceptance Criteria.'

Conclusion

In this module, we added additional logical structure to our program through the use of the Layer-Class-Function. I find this extremely valuable as it allows me to think of script development as following a coherent narrative form.

Appendix 1: Assignment06 Script

```
# ------------------------------------------------------------------------------------------ #
# Title: Assignment06
# Desc: This assignment demonstrates using functions
# with structured error handling
# Change Log: (Who, When, What)
#   RRoot,1/1/2030,Created Script
#   JD Blake, 5/19/24, Working script initiated, written and debugged.
#   JD Blake, 5/20/24, Revision with full comments
#   JD Blake, 5/21/24, Added 'clean' registration message function in IO class
#
# ------------------------------------------------------------------------------------------ #
import json

# Define the Data Constants
MENU: str = '''
---- Course Registration Program ----
  Select from the following menu:
    1. Register a Student for a Course.
    2. Show current data.
    3. Save data to a file.
    4. Exit the program.
-----------------------------------------
'''

FILE_NAME: str = "Enrollments.json"

# Define the Data Variables
students: list = []  # a table of student data
menu_choice: str  # Hold the choice made by the user.

# --------SEPARATION OF CONCERNS-------------
#
#   DATA LAYER
#       Class: FileProcessor
#
#   PRESENTATION LAYER
#       Class: IO
#
#   PROCESSING LAYER
#       Read in existing json data file
#       Call functions
#       Terminate program
# -------------------------------------------

# ---------DATA LAYER-----------------------


class FileProcessor:
    """
    File processing functions:
        read_data_from_file(file_name, student_data)
        write_data_to_file(tile_name, student_data)
    ChangeLog:
        JD Blake, 5/19/24, Created Class and populated with functions
    """

    @staticmethod
    def read_data_from_file(file_name: str, student_data: list):
        """
        Function to open and read json file data into list
        ChangeLog:
            JD Blake, 5/19/24, Created Function
        :param file_name:
        :param student_data:
        :return: student_data
        """
        try:
            file = open(file_name, "r")
            student_data = json.load(file)
            file.close()
        except FileNotFoundError as e:
            IO.output_error_messages("Text file must exist before running this script!", e)
        except Exception as e:
            IO.output_error_messages("There was a non-specific error!", e)
        finally:
            if file.closed == False:
                file.close()
        return student_data

    @staticmethod
    def write_data_to_file(file_name: str, student_data: list):
        """
        Function to write data stored in 'students' list to json file
        ChangeLog:
            JD Blake, 5/19/24, Created Function
        :param file_name:
        :param student_data:
        :return: None
        """
        try:
            file = open(file_name, "w")
            json.dump(student_data, file)
            file.close()
        except TypeError as e:
            IO.output_error_messages("Please check that the data is a valid JSON format", e)
        except Exception as e:
            IO.output_error_messages("There was a non-specific error!", e)
        finally:
            if file.closed == False:
                file.close()


# ---------PRESENTATION LAYER-----------------------

class IO:
    """
    Functions to control input/output (IO):
        output_error_message(message, Exception)
        output_menu(menu)
        input_menu_choice()
        input_student_data(student_data)
        output_student_data(student_data)
    ChangeLog:
        JD Blake, 5/19/24, Created Class and populated with functions
    """
    @staticmethod
    def output_error_messages(message: str, error: Exception = None):
        """
        Function to display a custom error messages to the user
        ChangeLog:
            JD Blake, 5/19/24, Created Function
        Returns: None
        """
        print(message, end="\n\n")
        if error is not None:
            print("-- Technical Error Message -- ")
            print(error, error.__doc__, type(error), sep='\n')

    @staticmethod
    def output_menu(menu: str):
        """
        Function displays the menu to the user
        ChangeLog:
            JD Blake, 5/19/24, Created Function
        :param menu:
        :return:
        """
        print()
        print(menu, '\n')  # Added new line to make it look nicer

    @staticmethod
    def input_menu_choice():
        """
        Function prompts user for menu choice
            ChangeLog
        JD Blake, 5/19/24, Created Function
        :return: choice
        """
        choice = "0"
        try:
            choice = input("What would you like to do? ")
            if choice not in ("1","2","3","4"):  # Note these are strings
                raise Exception("Please, choose only 1, 2, 3, or 4")
        except Exception as e:
            IO.output_error_messages(e.__str__())  # Not passing the exception object to avoid the technical
message
        return choice
    @staticmethod
    def input_student_data(student_data: list):
        """
        Function gets the first name, last name, and course name from the user
        and append it to list
        ChangeLog:
            JD Blake, 5/19/24, Created Function
        :param student_data:
        :return:
        """
        try:
            # Input the data
            student_first_name = input("What is the student's first name? ")
            if not student_first_name.isalpha():
                raise ValueError("The first name should not contain numbers.")

            student_last_name = input("What is the student's last name? ")
            if not student_last_name.isalpha():
                raise ValueError("The last name should not contain numbers.")

            course_name = input("What is the course name? ")

            student = {"FirstName": student_first_name,
                       "LastName": student_last_name,
                       "CourseName": course_name}
            student_data.append(student)

        except ValueError as e:
            IO.output_error_messages("That value is not the correct type of data!", e)
        except Exception as e:
            IO.output_error_messages("There was a non-specific error!", e)
        return student_data

    @staticmethod
    def output_student_data(student_data: list):
        """
        Function displays current student registration data stored in 'students' list formatted
        in to a friendly sentence.
        ChangeLog:
            JD Blake, 5/19/24, Created Function
        :param student_data:
        :return:
        """
        # Process the data to create and display a custom message
        print("-" * 50)
        for student in students:
            print(f'Student {student["FirstName"]} '
                  f'{student["LastName"]} is enrolled in {student["CourseName"]}')
        print("-" * 50)

    @staticmethod
    def output_student_data_clean(student_data: list):
        """
        Function displays current student registration data stored in 'students' list
        formatted to simply display first & last name and course name
        ChangeLog:
            JD Blake, 5/21/24, Created Function
        :param student_data:
        :return:
        """
        # Process the data to create and display a custom message
        print("-" * 50)
        for student in students:
            print(f'{student["FirstName"]} '
                  f'{student["LastName"]}, '
                  f'{student["CourseName"]}')
        print("-" * 50)


# ---------PROCESSING LAYER------------------

# Read In file data to 'students' list
students = FileProcessor.read_data_from_file(file_name=FILE_NAME, student_data=students)

# These operations repeat until the user opts to quit the program
while True:  # Loop will run until it sees 'break'
    # Call menu display function
    IO.output_menu(menu=MENU)

    # Call menu choice input function
    menu_choice = IO.input_menu_choice()

    if menu_choice == '1':  # Register student for course
        # Call student registration function
        IO.input_student_data(student_data=students)
        continue

    elif menu_choice == '2':  # Show current data
        # Call data display function
        IO.output_student_data(student_data=students)
        continue

    elif menu_choice == '3':  # Save data to a file
        # Call write-to-file function
        FileProcessor.write_data_to_file(file_name=FILE_NAME, student_data=students)
        print("\nData saved in:", FILE_NAME, '\n')  # Confirmation message
        print("\nFile Contents")
        print(students)
        print("\n Or More Clearly")
        IO.output_student_data_clean(student_data=students)
        continue

    elif menu_choice == '4':  # Exit the program
        break

print("Program Ended")  # Termination Message
```

Appendix 2: Validation Test

```
/usr/local/bin/python3.12 /Users/johnblake/Desktop/Python_Class/work/A06/Assignment06/Assignment06.py



---- Course Registration Program ----
  Select from the following menu:
    1. Register a Student for a Course.
    2. Show current data.
    3. Save data to a file.
    4. Exit the program.
-----------------------------------------



What would you like to do? 1
What is the student's first name? Jimi
What is the student's last name? Hendrix
What is the course name? GOAT 101



---- Course Registration Program ----
  Select from the following menu:
    1. Register a Student for a Course.
    2. Show current data.
    3. Save data to a file.
    4. Exit the program.
-----------------------------------------



What would you like to do? 2
--------------------------------------------------
Student Bob Smith is enrolled in Python 100
Student Sue Jones is enrolled in Python 100
Student Jimi Hendrix is enrolled in GOAT 101
--------------------------------------------------
```

```
---- Course Registration Program ----
  Select from the following menu:
    1. Register a Student for a Course.
    2. Show current data.
    3. Save data to a file.
    4. Exit the program.
-----------------------------------------



What would you like to do? 3


Data saved in: Enrollments.json
```

File Contents

```
[{'FirstName': 'Bob', 'LastName': 'Smith', 'CourseName': 'Python 100'}, {'FirstName': 'Sue', 'LastName': 'Jones',
'CourseName': 'Python 100'}, {'FirstName': 'Jimi', 'LastName': 'Hendrix', 'CourseName': 'GOAT 101'}]
```

Or More Clearly

```
--------------------------------------------------
Bob Smith, Python 100
Sue Jones, Python 100
Jimi Hendrix, GOAT 101
--------------------------------------------------



---- Course Registration Program ----
  Select from the following menu:
    1. Register a Student for a Course.
    2. Show current data.
    3. Save data to a file.
    4. Exit the program.
-----------------------------------------
```

```
What would you like to do? 1
What is the student's first name? Vince
What is the student's last name? Lombardi
What is the course name? GOAT 101


---- Course Registration Program ----
  Select from the following menu:
    1. Register a Student for a Course.
    2. Show current data.
    3. Save data to a file.
    4. Exit the program.
-----------------------------------------


What would you like to do? 1
What is the student's first name? Hank
What is the student's last name? Aaron
What is the course name? GOAT 101


---- Course Registration Program ----
  Select from the following menu:
    1. Register a Student for a Course.
    2. Show current data.
    3. Save data to a file.
    4. Exit the program.
-----------------------------------------


What would you like to do? 2
--------------------------------------------------
Student Bob Smith is enrolled in Python 100
Student Sue Jones is enrolled in Python 100
Student Jimi Hendrix is enrolled in GOAT 101
Student Vince Lombardi is enrolled in GOAT 101
Student Hank Aaron is enrolled in GOAT 101
--------------------------------------------------



---- Course Registration Program ----
  Select from the following menu:
    1. Register a Student for a Course.
    2. Show current data.
    3. Save data to a file.
    4. Exit the program.
-----------------------------------------



What would you like to do? 3


Data saved in: Enrollments.json
```

File Contents

```
[{'FirstName': 'Bob', 'LastName': 'Smith', 'CourseName': 'Python 100'}, {'FirstName': 'Sue', 'LastName': 'Jones',
'CourseName': 'Python 100'}, {'FirstName': 'Jimi', 'LastName': 'Hendrix', 'CourseName': 'GOAT 101'}, {'FirstName':
'Vince', 'LastName': 'Lombardi', 'CourseName': 'GOAT 101'}, {'FirstName': 'Hank', 'LastName': 'Aaron',
'CourseName': 'GOAT 101'}]
```

Or More Clearly

```
--------------------------------------------------
Bob Smith, Python 100
Sue Jones, Python 100
Jimi Hendrix, GOAT 101
Vince Lombardi, GOAT 101
Hank Aaron, GOAT 101
--------------------------------------------------



---- Course Registration Program ----
  Select from the following menu:
   1. Register a Student for a Course.
   2. Show current data.
   3. Save data to a file.
   4. Exit the program.
-----------------------------------------


What would you like to do? 5
Please, choose only 1, 2, 3, or 4



---- Course Registration Program ----
  Select from the following menu:
   1. Register a Student for a Course.
   2. Show current data.
   3. Save data to a file.
   4. Exit the program.
-----------------------------------------


What would you like to do? 4
Program Ended

Process finished with exit code 0
```