Mine Hantal Wernsing
5/17/2024
IT FDN110
A06 – Functions
[Mine Wernsing GitHub]

Creating and Testing a Python Script with Functions, Classes, and the Separation of Concerns Programming Pattern

Introduction

Functions, classes, and separation of concerns programming pattern are three common techniques for improving our Python scripts. I have created a Python program that demonstrates using constants, variables, and print statements with the use of functions, classes, and using the separation of concern pattern for a more structured and organized code which helps us create modular, maintainable code for working as a more proficient programmer.

Creating the Script

File Name:

The file is named assignment06_minewernsing.py (Fig. 1)

Script Header:

The script header includes the title, description, change log and is updated with name and the current date (Fig. 1).

Constants:

The constant **MENU: str** is set to the value:

```
---- 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
----------------------------------------
```

The constant **FILE_NAME: str** is set to the value "Enrollments.json" (Fig. 1).

Variables:

menu_choice: str is set to a string

students: list[dict[str, str]] = [] is set to and empty list (Fig. 1)

```python
# ------------------------------------------------------------------------------------------ #
# Title: Assignment06
# Desc: This assignment demonstrates using functions
# with structured error handling
# Change Log: (Who, When, What)
#   Mine Wernsing,5/17/2024,Created Script
# ------------------------------------------------------------------------------------------ #

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[dict[str, str]] = []  # a table of student data
menu_choice: str  # Holds the choice made by the user
```

Figure 1. File name, script reader, constants, and variables.

Classes

The program includes a class named **FileProcessor** for file reading and writing code (Fig. 2).

The program includes another class named **IO** for input/output code (Fig. 2).

All classes include descriptive document strings.

```python
# Processing ------------------------------------- #
2 usages
class FileProcessor:...


# Presentation ----------------------------------- #
10 usages
class IO:...
```

Figure 2. FileProcessor and IO classes.

Functions

All functions include descriptive document strings (Fig. 3a, 3b).

All functions except blocks include calls to the function handling error messages.

All functions use the @staticmethod decorator (Fig. 3a, 3b).

The program includes functions with the following names and parameters:

- output_error_messages(message: str, error: Exception = None) (Fig. 4)
- output_menu(menu: str) (Fig. 5)
- input_menu_choice() (Fig. 5)
- output_student_courses(student_data: list) (Fig. 6)
- input_student_data(student_data: list) (Fig. 7)
- read_data_from_file(file_name: str, student_data: list): (Fig. 8)
- write_data_to_file(file_name: str, student_data: list): (Fig. 9)

```
# Processing ------------------------------------- #
2 usages
class FileProcessor:
    """A collection of processing layer functions that work with json files..."""
    1 usage
    @staticmethod
    def read_data_from_file(file_name: str, student_data: list):...

    1 usage
    @staticmethod
    def write_data_to_file(file_name: str, student_data: list):...
```

Figure 3a. Defined functions with @staticmethod in FileProcessor class.

```python
# Presentation ------------------------------------- #
10 usages
class IO:
    """A collection of presentation layer functions that manage user input and output..."""

    5 usages
    @staticmethod
    def output_error_message(message: str, error: Exception = None):...

    1 usage
    @staticmethod
    def output_menu(menu: str):...

    1 usage
    @staticmethod
    def input_menu_choice():...

    2 usages
    @staticmethod
    def output_student_and_course_names(student_data: list):...

    1 usage
    @staticmethod
    def input_student_data(student_data: list):...
```

Figure 3b. Defined functions with @staticmethod in IO class.

```python
class IO:
    """
    A collection of presentation layer functions that manage user input and output

    ChangeLog: (Who, When, What)
    Mine Wernsing,5/17/2024,Created Class
    Mine Wernsing,5/17/2024,Added menu output and input functions
    Mine Wernsing,5/17/2024,Added a function to display the data
    Mine Wernsing,5/17/2024,Added a function to display custom error messages

    """

    5 usages
    @staticmethod
    def output_error_message(message: str, error: Exception = None):
        """ This function displays a custom error message to the user

        Change Log: (Who, When, What)
        Mine Wernsing,5/17/2024,Created Function

        :return: None
        """
        print(message, end="\n\n")
        if error is not None:
            print("-- Technical Error Message--")
            print(error, error.__doc__, type(error), sep='\n')
```

Figure 4. Class IO output_error_message function.

```python
class IO:
    @staticmethod
    def output_menu(menu: str):
        """ This function displays the menu of choices to the user

        Change Log: (Who, When, What)
        Mine Wernsing,5/17/2024,Created Function

        :return: None
        """
        print()  # Adding extra space to make it look nicer
        print(menu)
        print()  # Adding extra space to make it look nicer

    1 usage
    @staticmethod
    def input_menu_choice():
        """ This function gets a menu choice from the user
        Change Log: (Who, When, What)
        Mine Wernsing,5/17/2024,Created Function

        :return: string with the user's choice
        """
        choice = "0"
        try:
            choice = input("Enter your menu choice number: ")
            if choice not in ("1", "2", "3", "4"):
                raise Exception("Please choose only 1, 2, 3, or 4")
        except Exception as error_details:
            IO.output_error_message(error_details.__str__())  # Not passing error details to avoid the technical message

        return choice
```

Figure 5. IO class output_menu and input_menu_choice functions.

```python
@staticmethod
def output_student_and_course_names(student_data: list):
    """ This function displays the students and course names to the user

    Change Log: (Who, When, What)
    Mine Wernsing,5/17/2024,Created Function

    :return: None
    """
    print("-" * 50)
    for student in student_data:
        print(f'Student {student["FirstName"]} '
              f'{student["LastName"]} is enrolled in {student["CourseName"]}')
    print("-" * 50)
```

Figure 6. IO class output_students_and_course_names function.

```python
@staticmethod
def input_student_data(student_data: list):
    """This function gets the student's first name and last name, with a course name from the user..."""

    try:
        student_first_name = input("Enter the student's first name: ")
        if not student_first_name.isalpha():
            raise ValueError("Student first name must be alphabetic. ")

        student_last_name = input("Enter the student's last name: ")
        if not student_last_name.isalpha():
            raise ValueError("Student last name must be alphabetic. ")

        course_name = input("Please enter the name of the course: ")
        student = {"FirstName": student_first_name,
                   "LastName": student_last_name,
                   "CourseName": course_name}
        student_data.append(student)
        print()
        print(f"You have registered {student_first_name} {student_last_name} for {course_name}.")
    except ValueError as error_details:
        IO.output_error_message(message="One of the values was not the correct type of data", error=error_details)

    except Exception as error_details:
        IO.output_error_message(message="Error: There was a problem with your entered data.", error=error_details)

    return student_data
```

Figure 7. IO class input_student_data function.

```python
class FileProcessor:
    Mine Wernsing,5/17/2024,Created Class
    """
    1 usage
    @staticmethod
    def read_data_from_file(file_name: str, student_data: list):
        """ This function reads data from a json file and loads it into a list of dictionary rows

        Change Log: (Who, When, What)
        Mine Wernsing,5/17/2024,Created Function

        :return: list
        """

        file = None

        try:
            file = open(file_name, "r")
            student_data = json.load(file)
            file.close()
        except Exception as error_details:
            IO.output_error_message(message="Error: There was a problem with reading the file.", error=error_details)

        finally:
            if file is not None and not file.closed:
                file.close()
        return student_data
```

Figure 8. FileProcessor class read_data_from_file function.

```python
@staticmethod
def write_data_to_file(file_name: str, student_data: list):
    """ This function writes data to a json file and with data from a list of dictionary rows

    Change Log: (Who, When, What)
    Mine Wernsing,5/17/2024,Created Function

    :return: None
    """
    file = None

    try:
        file = open(file_name, "w")
        json.dump(student_data, file)
        file.close()
        IO.output_student_and_course_names(student_data=student_data)
    except Exception as error_details:
        message = "Error: There was a problem with writing to the file.\n"
        message += "Please check that the file is not open by another program."
        IO.output_error_message(message=message, error=error_details)
    finally:
        if file is not None and not file.closed:
            file.close()
```

Figure 9. FileProcessor class write_data_to_file function.

Input / Output:

On menu choice 1, the program prompts the user to enter the student's first name and last name, followed by the course name, using the input() function and stores the inputs in the respective variables.

On menu choice 2, the program presents a string by formatting the collected data using the print() function.

Data collected for menu choice 1 is added to a list of dictionaries.

All data in the list is displayed when menu choice 2 is used.

Processing

When the program starts, the contents of the "Enrollments.json" are automatically read into a list of dictionary rows.

On menu choice 3, the program opens a file named "Enrollments.json" in write mode using the open() function. It writes the content of the students variable to the file using the dump() function, then file is closed using the close() method. The program then displays what was stored in the file.

On menu choice 4, the program ends.

Error Handling

The program provides structured error handling when the file is read into the list of dictionary rows.

The program provides structured error handling when the user enters a first name (Fig. 10).

The program provides structured error handling when the user enters a last name.

The program provides structured error handling when the dictionary rows are written to the file.

```
Enter your menu choice number: 1
Enter the student's first name: Dave4
One of the values was not the correct type of data

-- Technical Error Message--
Student first name must be alphabetic.
Inappropriate argument value (of correct type).
<class 'ValueError'>
```

Figure 10. Input student data function error handling.

Testing the Script

Test

The program takes the user's input for a student's first, last name, and course name.

The program displays the user's input for a student's first, last name, and course name (Figs. 11, 13).

The program allows users to enter multiple registrations (first name, last name, course name).

The program allows users to display multiple registrations (first name, last name, course name) (Figs. 11, 14).

The program allows users to save multiple registrations to a file (first name, last name, course name) (Fig. 12).

The program runs correctly in both **PyCharm** and **Command Prompt**.

```
Enter your menu choice number: 1
Enter the student's first name: Sue
Enter the student's last name: Salias
Please enter the name of the course: Python

You have registered Sue Salias for Python.


---- 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.
----------------------------------------


Enter your menu choice number: 2
-----------------------------------------------
Student David Johnson is enrolled in Python
Student Joe Preston is enrolled in Python
Student Sue Salias is enrolled in Python
-----------------------------------------------
```

Figure 11. The program takes multiple user data and displays it in PyCharm.

```
assignment06_minewernsing.py        Enrollments.json
[{"FirstName": "David", "LastName": "Johnson", "CourseName": "Python"},
  {"FirstName": "Joe", "LastName": "Preston", "CourseName": "Python"},
  {"FirstName": "Sue", "LastName": "Salias", "CourseName": "Python"}]
```

```
Run    assignment06_minewernsing

Enter your menu choice number: 3
-----------------------------------------------
Student David Johnson is enrolled in Python
Student Joe Preston is enrolled in Python
Student Sue Salias is enrolled in Python
-----------------------------------------------


---- 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.
----------------------------------------


Enter your menu choice number: 5
Please choose only 1, 2, 3, or 4

Please only choose option 1, 2, 3 or 4
```

Figure 12. The program allows users to save multiple registrations to a file.

```
Command Prompt - python a

Microsoft Windows [Version 10.0.22631.3593]
(c) Microsoft Corporation. All rights reserved.

C:\Users\mineh>cd Documents

C:\Users\mineh\Documents>cd PycharmProjects

C:\Users\mineh\Documents\PycharmProjects>cd assignment06_mine

C:\Users\mineh\Documents\PycharmProjects\assignment06_mine>python assignment06_minewernsing.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.
----------------------------------------


Enter your menu choice number: 1
Enter the student's first name: Bob
Enter the student's last name: Baker
Please enter the name of the course: Python

You have registered Bob Baker for Python.


---- 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.
----------------------------------------


Enter your menu choice number:
```

Figure 13. The program runs smoothly in Command Prompt.

```
Command Prompt

Enter your menu choice number: 2
-------------------------------------------------
Student David Johnson is enrolled in Python
Student Joe Preston is enrolled in Python
Student Sue Salias is enrolled in Python
Student Bob Baker is enrolled in Python
-------------------------------------------------


---- 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.
----------------------------------------


Enter your menu choice number: 3
-------------------------------------------------
Student David Johnson is enrolled in Python
Student Joe Preston is enrolled in Python
Student Sue Salias is enrolled in Python
Student Bob Baker is enrolled in Python
-------------------------------------------------


---- 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.
----------------------------------------


Enter your menu choice number: 4
Program Ended
```

Figure 14. Program testing results in Command Prompt.

Source Control

The script file and the knowledge document are hosted on a GitHub repository.

A link to the repository is below:

[MineWernsing/IntroToProg-Python-Mod06 (github.com)]

A link to the repository is in the GitHub links forum below:

[Topic: Assignment 06 Documents for Review! (uw.edu)]

Summary

I have created a Python program that demonstrates using constants, variables, and print statements to display a message about students' registration for courses. I have added the use of functions, classes and using the separation of concerns pattern. I organized the code into two classes, defined and called the functions with arguments. I used structured error handling to make sure the code doesn't break. I tested the Python code in both PyCharm and Command Prompt for a successful run.