Justin Price

May 22, 2024

IT FDN 100 A

Assignment 06

GitHub: https://github.com/jwprice3 (external)

The Classless Hallows

Intro:

There were once three brothers who attended Hogwarts University and were inseparable. Professor Exception, their professor and a triplet himself, hated his brothers and would simply be brought to rage every time he saw the triples together. One of the triplets was named name Def. His sole purpose was to dictate what was needed in his brother's lives, but he could not obtain it himself. Another triplet was named Param, as he knew how to get his brothers but was too indecisive to pick a way to start. The last brother was Static as he could only pick the right way for his brother Param to get what the brothers needed. One day, after years of planning, Professor Exception snuck into the admin department of Hogwarts University and changed the brother's course registration by changing Professor Justin's Python script. He laughed manically as he imagined each brother's look as they would be separated at the start of the next semester. A few months passed, and the spring semester was to begin. Professor Exception roamed the hallways of HU to find the boys heartbroken and devasted, but that is not what he saw. He saw the boys smiling and galivanting around. Perplexed and enraged at the sight, Professor Exception ran up to them and screamed, "How did you do it!". The hoys exchanged a grin and pulled out their Personal Computers (PC). You see, they were also trained in the. Python Arts and together the brothers Def, Param, and Static formed the Classless Hallows, the most powerful scripting at Hogwarts University.

Body:

To mimic multiple entries to a script, I will implement the change log as one continuous script.

```
# ------------------------------------------------------------------------------------------ #
# Title: Assignment06
# Desc: This assignment demonstrates using dictionaries, files, and exception handling
# Change Log: (Who, When, What)
#   JP,13MAT24,Created Script
#   JP,16MAY24,Added Exception
#   JP,22MAY24,Added Classes and Functions
# ------------------------------------------------------------------------------------------ #
```

Figure 1.1 Description of File Admin

This version has a lot fewer variables since we are embedding them in the function calls.

```
# Define the Data Constants and variables
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.
-----------------------------------------
'''
# Define the Data Constants
FILE_NAME: str = "Enrollments.json"
#Define the variables
students: list = []  # a table of student data NEEDS TO BE USED
menu_choice: str = ''# Hold the choice made by the user.
```

Figure 2.1 Constants and Variables

Separation of concerns is like a table of contents for the user's script; it should help prepare the reader for processing the information.

```
# --------SEPARATION OF CONCERNS-------------
#
#   DATA LAYER
#       Class: FileProcessor
#
#   PRESENTATION LAYER
#       Class: IO
#
#   PROCESSING LAYER
#       Create/Read JSON file
#       Call functions
#       Terminate program
# ----------------------------------------

# ---------DATA LAYER-----------------------
# Define the Data Variables


# Processing ---------------------------------------- #
```

Figure 3.1 Separation of Concerns

First, I created the write_data_to_file function in the FileProcessor class.

```python
class FileProcessor:
    '''
        A collection of processing layer functions that work with json files

        ChangeLog: (Who, When, What)
        JP,22MAY24,Created Class
    '''
    @staticmethod
    def write_data_to_file(fileName: str, student_data: list):
        """ This function writes data from a list  to a json file

        Notes:
        - Data sent to the student_table parameter will be overwritten.

        ChangeLog: (Who, When, What)
        JP,22MAY24,Created function

        :param fileName: string with the name of the file we are reading
        :param student_table: list of dictionary rows we are adding data to
        :return: list of dictionary rows filled newly added data
        """
        try:
            file = open(FILE_NAME, mode: "w")
            json.dump(student_table, file)
        except FileNotFoundError as e:
            print("Text file must exist before running this script!\n")
            print("-- Technical Error Message -- ")
            print(e, e.__doc__, type(e), sep='\n')
        except Exception as e:
            print("There was a non-specific error!\n")
            print("-- Technical Error Message -- ")
            print(e, e.__doc__, type(e), sep='\n')
        finally:
            file.close()
            print("The following data was saved to file!")
            return student_table
```

Figure 4.1 FileProcessor class

```python
class FileProcessor:
    '''
        A collection of processing layer functions that work with json files

        ChangeLog: (Who, When, What)
        JP,22MAY24,Created Class
    '''
    @staticmethod
    def write_data_to_file(fileName: str, student_data: list):
        """ This function writes data from a list  to a json file

        Notes:
        - Data sent to the student_table parameter will be overwritten.

        ChangeLog: (Who, When, What)
        JP,22MAY24,Created function

        :param fileName: string with the name of the file we are reading
        :param student_table: list of dictionary rows we are adding data to
        :return: list of dictionary rows filled newly added data
        """
        try:
            file = open(FILE_NAME, mode: "w")
            json.dump(student_table, file)
        except FileNotFoundError as e:
            print("Text file must exist before running this script!\n")
            print("-- Technical Error Message -- ")
            print(e, e.__doc__, type(e), sep='\n')
        except Exception as e:
            print("There was a non-specific error!\n")
            print("-- Technical Error Message -- ")
            print(e, e.__doc__, type(e), sep='\n')
        finally:
            file.close()
            print("The following data was saved to file!")
            return student_table
```

Figure 4.2 FileProcessor class

I then created the IO class along with the functions.

```python
class IO:
    '''
        A collection of input/output (IO) layer functions that work with json files

        ChangeLog: (Who, When, What)
        JP,22MAY24,Created Class
    '''

    @staticmethod
    def output_error_message(message: str, error: Exception = None):
        """ This function displays and error message when an Exception is reached

        Notes:
        -None
        :param message and the new lines at the end of the message
        ChangeLog: (Who, When, What)
        JP,22MAY24,Created function
        """
        print(message, end="\n\n")
        if error is not None:
            print("-- Technical Error Message -- ")
            print(e, e.__doc__, type(e), sep='\n')
```

Figure 5.1 IO class

```python
    @staticmethod
    def input_menu_choice():
        """ This function will allow the user to select options 1 - 4 and will raise an exception
        for any other input

        Notes:
        - None
        ChangeLog: (Who, When, What)
        P,22MAY24,Created function
        :return: prompts the user to input accepted an accepted input
        """
        try:

            options = {"1", "2", "3", "4"}
            menu_choice = input("What would you like to do: ")
            if menu_choice not in options:
                raise Exception("Invalid choice. Please enter a number from 1 through 4.")

        except Exception as e:
                IO.output_error_message(e.__str__())
        finally:
            return menu_choice


    @staticmethod
    def output_menu(menu: str):
        """ This function will display the menu options, the MENU is a global constant

            Notes:
            - None
            ChangeLog: (Who, When, What)
            JP,22MAY24,Created function
            :param menu: str
        """
        global MENU

        print()
        print(MENU)
        print()  # Adding extra space to make it look nicer.
```

Figure 5.2 IO class

```python
@staticmethod
def input_student_data(student_data: list):
    """ This function will allow the user to input their first name, last name and course.
        Secondly this adds the newly input data in the student_table. Finally, a message with
        the new data will display to the user what was just added.

        Notes:
        - There are error exceptions if the user inputs numbers when it expects letters
        ChangeLog: (Who, When, What)
        JP,22MAY24,Created function
        :param student_data
    """
    global student_first_name
    global student_last_name
    global course_name

    try:
        # Check that the input does not include numbers
        student_first_name = input("Enter the student's first name: ")
        if not student_first_name.isalpha():
            raise ValueError("The first name should not contain numbers.")
        student_last_name = input("Enter the student's last name: ")
        if not student_last_name.isalpha():
            raise ValueError("The last name should not contain numbers.")
    except Exception as e:
        print("There was a non-specific error!\n")
        print("-- Technical Error Message -- ")
        print(e, e.__doc__, type(e), sep='\n')
    course_name = input("Please enter the name of the course: ")
    student_data = {"FirstName": student_first_name,
                    "LastName": student_last_name,
                    "Course": course_name}
    student_table.append(student_data)
    print(f"You have registered {student_first_name} {student_last_name} for {course_name}.")
```

Figure 5.3 IO class

```python
    @staticmethod
    def output_student_courses(student_data:list):
        """ This function will display the current data from student_table which is in JSON format.

            Notes:
            - None
            ChangeLog: (Who, When, What)
            JP,22MAY24,Created function
            :param student_data
        """
        print("-" * 50)
        print("\nCurrent registered students:")
        print(student_table)
        print("-" * 50)
```

Figure 5.4 IO class

I left my json creation portion in, since I use multiple environments to complete this, creating the script and therefore a pathway is always helpful.

```
# creates a dictionary if one does not exist for the Professor utilizing the Python Arts script.
'''
This portion is to create a json file, assuming that there has not been a file that
has not been created or a defined pathway. >>>>
'''

try:
    file = open(FILE_NAME, mode: "r")
    student_table = json.load(file)
    for item in students:
        print(f"FirstName: {item['FirstName']}, LastName: {item['LastName']}, Course: {item['Course']}")
except FileNotFoundError as e:
    print("JSON file must exist before running this script!\n")
    print("-- Technical Error Message -- ")
    print(e, e.__doc__, type(e), sep='\n')
    print("Prof. Justin will create a JSON for you..")
    student_row1: dict = {"FirstName": "First Name", "LastName": "Last Name", "Course": "Course"}
    student_table: list = [student_row1]
    file = open( name: "Enrollments.json", mode: "w")
    json.dump(student_table, file)
    file.close()
except Exception as e:
    print("There was a non-specific error!\n")
    print("-- Technical Error Message -- ")
    print(e, e.__doc__, type(e), sep='\n')
    print("JSON document successfully created")

'''
<<<< This portion is to create a json file, assuming that there has not been a file that
has not been created or a defined pathway.
'''
# Begining the main body of the script
```

Figure 6.1 JSON creation

This is portion where the functions get executed.

```
# Begining the main body of the script

while True:
    IO.output_menu(menu=MENU)

    # Present the menu of choices
    menu_choice=IO.input_menu_choice()
    # Present the menu of choices

    # Input user data
    if menu_choice == "1":  # This will not work if it is an integer!
        IO.input_student_data(student_data=students)

    # Present the current data
    elif menu_choice == "2":
        IO.output_student_courses(student_data=students)
        # Process the data to create and display a custom message

    # Save the data to a file
    elif menu_choice == "3":
        try:
            FileProcessor.write_data_to_file(fileName=FILE_NAME, student_data=students)
        finally:
            FileProcessor.read_data_from_file(fileName=FILE_NAME, student_data=students)
        continue

    # Stop the loop
    elif menu_choice == "4":
        break  # out of the loop

print("Program Ended")
```

Figure 7.1 Execution of the funtions.

I also ran this in IDLE check my work.

```
Python 3.12.3 (v3.12.3:f6650f9ad7, Apr  9 2024, 08:18:47) [Clang 13.0.0 (clang-1300.0.29.30)] on
darwin
Type "help", "copyright", "credits" or "license()" for more information.

= RESTART: /Users/jp/Desktop/Python/01 - Python_Foundations/06 - Module/_Module06/Assignment/Ass
ignment06.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
Enter the student's first name: Seamus
Enter the student's last name: Finnigan
Please enter the name of the course: Python100
You have registered Seamus Finnigan for Python100.


---- 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
Enter the student's first name: Moaning
Enter the student's last name: Myrtle
Please enter the name of the course: Python100
You have registered Moaning Myrtle for Python100.


---- 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
-------------------------------------------------

Current registered students:
[{'FirstName': 'First Name', 'LastName': 'Last Name', 'Course': 'Course'}, {'FirstName': 'Seamus
', 'LastName': 'Finnigan', 'Course': 'Python100'}, {'FirstName': 'Moaning', 'LastName': 'Myrtle
', 'Course': 'Python100'}]
-------------------------------------------------


---- 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.
-----------------------------------------
```

Figure 8.1 IDLE Execution

```
What would you like to do: 3
The following data was saved to file!
[{'FirstName': 'First Name', 'LastName': 'Last Name', 'Course': 'Course'}, {'FirstName': 'Seamus
', 'LastName': 'Finnigan', 'Course': 'Python100'}, {'FirstName': 'Moaning', 'LastName': 'Myrtle'
, 'Course': 'Python100'}]


---- 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
```

Figure 8.2 IDLE Execution

Summary:
It’s a bit of hogwash, innit? I mean, who really believes in the Classless Hallows? It is just a story my mum told me before I fell asleep. But what if they did exist? How powerful could someone trained in the Python Arts become…to be continued.