Rob Armstrong

August 6, 2024

IT FDN 110 A

Assignment06

https://github.com/robstrong517/IntroToProg-Python-Mod06

Intro to Programming with Python

Module 06 - Functions

Introduction

Since last week allowed for a more personalized touch to code with the use of error handling, this week took a step back in the direction of automation with the intention of using the code for reiterative use. While the interactive response of error handling is still in place, this week required many more operational definitions (and the respective shorthand) to recall these definitions in later steps within the code.

Writing the Code

As it has been each week, the first step has been to define the constants, variables, and determine the output file. This week, the constants were identical, and the variables used were greatly reduced. Many definitions throughout used more local variables so only two global variables were required.

```
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"

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

FIGURE 1: IMPORT JSON, CONSTANTS, AND GLOBAL VARIABLES USED WITHIN.

Functions

As mentioned in the introduction, using these definitions, or *functions*, significantly simplifies the main body of script. While the earlier chapters of the script become much more complex, it allows for each chapter to be better compartmentalized. Prior to creating functions for the inputs and outputs, it was required to build functions for reading and writing to the JSON. Following that was the presentation of the data, to include the inputs (menu choices) and their respective outputs. Each of these subcategories also included a portion for error handling.

```
# Processing ------------------------------------ #
class FileProcessor:
    """
    A collection of processing layer functions that work with JSON files.

    ChangeLog: (Who, When, What)
    RArmstrong,08.05.2024,Created Class
    """
    @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

        ChangeLog: (Who, When, What)
        RArmstrong,08.05.2024,Created function

        :return: list
        """
        try:
            with open(file_name, "r") as file:
                student_data = json.load(file)
        except Exception as e:
            IO.output_error_messages(message="Error: There was a problem with reading the file.", error=e)
        return student_data
```

FIGURE 2: THE FIRST FUNCTION USED, DEFINING READ_DATA_FROM_FILE.

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

    ChangeLog: (Who, When, What)
    RArmstrong,08.05.2024,Created function

    :return: None
    """
    try:
        with open(file_name, "w") as file:
            json.dump(student_data, file)
        IO.output_student_and_course_names(student_data=student_data)
    except Exception as e:
        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_messages(message=message, error=e)
```

FIGURE 3: FUNCTION TWO, DEFINITION FOR WRITE_DATA_TO_FILE.

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

    ChangeLog: (Who, When, What)
    RArmstrong,08.05.2024,Created Class
    """

    @staticmethod
    def output_error_messages(message: str, error: Exception = None):
        """ This function displays custom error messages to the user.

        ChangeLog: (Who, When, What)
        RArmstrong,08.05.2024,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: PRESENTATION FUNCTION (FUNCTION THREE), DEFINING OUTPUT_ERROR_MESSAGES.

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

    ChangeLog: (Who, When, What)
    RArmstrong,08.05.2024,Created function

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

FIGURE 5: FUNCTION FOUR, DEFINING OUTPUT_MENU.

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

    ChangeLog: (Who, When, What)
    RArmstrong,08.05.2024,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: FUNCTION FIVE, DEFINING OUTPUT_STUDENT_AND_COURSE_NAMES.

```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.

    ChangeLog: (Who, When, What)
    RArmstrong,08.05.2024,Created function

    :return: list with new student data
    """
    try:
        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.")
        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(f"You have registered {student_first_name} {student_last_name} for {course_name}.")
    except ValueError as e:
        IO.output_error_messages(message="One of the values was an incorrect type of data!", error=e)
    except Exception as e:
        IO.output_error_messages(message="Error: There was a problem with your entered data.", error=e)
    return student_data
```

FIGURE 7: FUNCTION SIX, DEFINITION FOR INPUT_STUDENT_DATA.

```python
# Start of main body ------------------------------------- #

# When the program starts, read the file data into a list of lists (table)
students = FileProcessor.read_data_from_file(file_name=FILE_NAME, student_data=students)

# Present and Process the data
while True:
    # Present the menu of choices
    IO.output_menu(menu=MENU)

    # Get menu choice
    menu_choice = IO.input_menu_choice()

    # Input user data
    if menu_choice == "1":
        students = IO.input_student_data(student_data=students)

    # Present the current data
    elif menu_choice == "2":
        IO.output_student_and_course_names(student_data=students)

    # Save the data to a file
    elif menu_choice == "3":
        FileProcessor.write_data_to_file(file_name=FILE_NAME, student_data=students)

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

    else:
        print("Please only choose option 1, 2, 3, or 4")

print("Program Ended")
```

FIGURE 8: MAIN BODY OF SCRIPT, USING SHORTHAND DEFINITIONS.

Main Body

Lastly, after all functions had been defined and worked as they should, it was time to build the main body of the script. This main body was able to be concise and clean, making it easy to read exactly what the purpose of the script is and what actions are or should be taken.

Summary

This was another difficult assignment, yet being able to construct code in this manner repeatedly would be a good test of proficiency. In early assignments, I compared coding to using algebraic equations, subbing in *x* and *y* values into a formula. The foundations of what is happening in my code this week are similar, yet I'm using more complex derivatives to arrive at a similar result.