Cristina Larrea

3/4/2025

Foundations of Programming: Python

Assignment 06

cri-larr/IntroToProg-Python-Mod06

Assignment 06 – Functions

Introduction

This week's lesson included two main new topics, functions and classes, intended to help with organizing scripts and more scalable. The two focuses of the function lesson were to support modularity and reusability, ensuring down the road you can better scale and reduce bugs in your code. Additionally, Global and local variables are also introduced to work along these topics, providing additional categorization. Parameters and return values are also introduced as they are heavily used along with functions, although difficult to understand at first, it makes more sense as you integrate them into the concept of functions. Finally, the labs within the course heavily supported the reading material, providing additional opportunities to practice especially as script has gotten very complex, requiring additional practice to avoid being overwhelmed.

Classes and Functions

Classes and functions are introduced in these lessons to support code organization and scalability. Functions are used to support organizing code and ensuring there is modularity and reusability, meaning you can break down program into smaller pieces also supporting quick re-use. Functions help group statements and have a specific set up, statement is called and then a main body makes up the logic to run when the function is called.

Classes are used to group created functions, variables and constants, making it a key concept in programming. This is very important to leverage large projects, in the case of this week, there was beginning to be a lot of repetitive content, making it easier to relate the need for classes. For this course, we will be using static classes, which remain unchanged, indicated by using @staticmethod decorator and avoiding need to create an object first. It is good practice to include document strings for your classes, providing explainability into what the functions in this class do along with dedicated change logs.

Script Details

The assignment required running the same menu and actions as past classes, but the main task was to incorporate a group of many functions and two classes to achieve this logic while using descriptive document strings. The classes would break up the code into a section with functions intending to process the file like writing and reading and a section for dealing with inputs and outputs from the user. Image 1

shows both classes in my script, and how the functions are being identified below the class, for this example, you can see that the processing class specifically has code that interacts with the file.

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



    """
A collection of processing functions that work with Json files

ChangeLog:
Cristina, 3/4/20205, created first class
Cristina, 3/4/20205, added two functions to interact with JSON data
Cristina, 3/4/20205, included exception handling to the functions

    """

# When the program starts, read the file data into a list of lists (table)
# Extract the data from the file

    @staticmethod  1 usage
    def read_data_from_file(file_name: str, student_data: list):
        try:
            file = open(FILE_NAME, "r")
            student_data = json.load(file)
            file.close()
        except Exception as e:
            print("Error: There was a problem with reading the file.")
            print("Please check that the file exists and that it is in a json format.")
            print("-- Technical Error Message -- ")
            print(e.__doc__)
            print(e.__str__())
        finally:
            if file.closed == False:
                file.close()
        return student_data

    @staticmethod  1 usage
    def write_data_to_file(file_name: str, student_data: list):
        # global file
        # global students

        try:
            file = open(file_name, "w")
            json.dump(student_data, file)
            file.close()
        except TypeError as e:
            IO.output_error_messages( message: "Please check that the data is a valid JSON format", e)
        except Exception as e:
            IO.output_error_messages( message: "There was a non-specific error!", e)
        finally:
            if file.closed == False:
                file.close()

# Presentation ------------------------------------- #
class IO:  6 usages
    """
    A collection of presentation functions that manage user input and output
```

Image 1 – Classes

The code below in Image2 shows what calling the functions you create within the classes looks like and how much more simplified and readable it is, especially when comparing to the length of the section creating the functions, which would otherwise be difficult to debug.

```python
# Beginning of the main body of this script
students = FileProcessor.read_data_from_file(file_name=FILE_NAME, student_data=students)

# Repeat the follow tasks
while True:
    IO.output_menu(menu=MENU)

    menu_choice = IO.input_menu_choice()

    if menu_choice == "1":  # Display current data
        output_student_courses(student_data=students)
        continue

    elif menu_choice == "2":  # Get new data (and display the change)
        IO.input_student_data(student_data=students)
        output_student_courses(student_data=students)
        continue

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

    elif menu_choice == "4":  # End the program
        print('the program has ended, thank you')
        break  # out of the while loop
```

Image 2- Using Functions

Additional requirements involved only using the '@staticmethod' decorator and calling a specific function for handling error messages. The image 3 below shows this being used, along with the documentation and some exception handling.

```python
class IO:  6 usages
    """
    A collection of presentation functions that manage user input and output

    ChangeLog:
    Cristina,3.4.2025,Created Class
    Cristina,3.4.2025, Added a function to display menu details
    Cristina,3.4.2025, Added a function to request inputs from user
    Cristina,3.4.2025,Added a function to display custom error messages and course details to user
    """
    pass

    @staticmethod  3 usages
    def output_error_messages(message: str, error: Exception = None):
        """ This function displays a custom error messages to the user
        ChangeLog:
        Cristina,3.4.2025,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')
```

Image 3 – Error handling static method function

Summary

This week's lesson was the most complex so far, introducing functions which can be very handy in simplifying your code. The concept took a bit of time to grasp but once it made sense the logic was clear. The use of paraments was also covered which were supported by a lab and reading material on them. The labs and examples throughout the notes really supplemented the theory as it provided simple versions of using a function as an introduction. The assignment included a lot more complexity, but it was helpful to see how everything we have learned so far can be made more efficient and readable using functions. The final code can be easily understood when reading in comparison to not using functions, blending all the information together, with no clear order outside of top bottom.