Hyrum Magnusson

2025MAR05

IT FDN 110 A Wi 25: Foundations of Programming: Python

Assignment 06

https://github.com/HMag-PFun/IntroToProg-Python-Mod6

Organization of Code

Introduction

Programming is built upon fundamental structures that allow for efficient, organized, and maintainable code. Among these, functions and classes are crucial for structuring a program effectively. Functions encapsulate logic into reusable blocks, while classes organize data and methods, promoting modularity and re-usability.

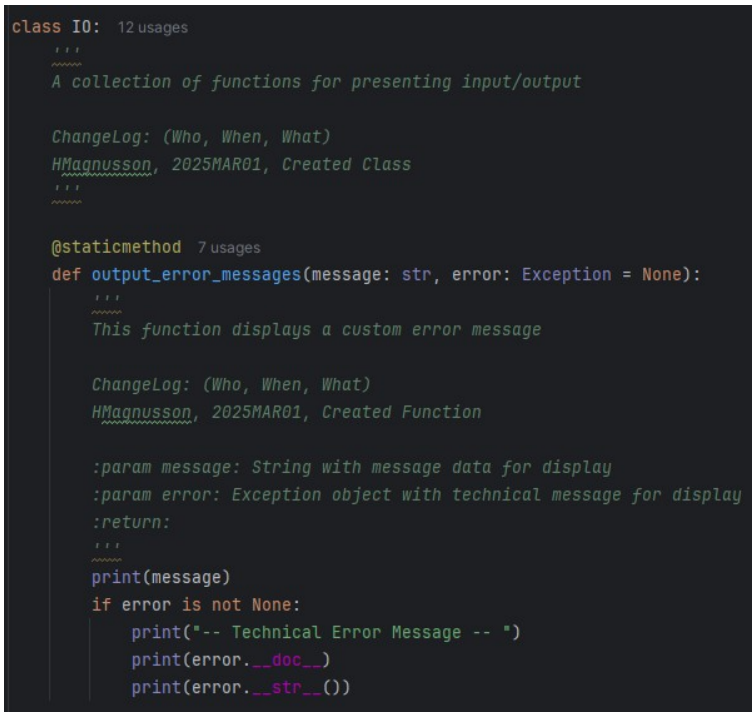
Separation of Concerns

The principle of Separation of Concerns (SoC) further enhances maintainability, scalability, and readability by dividing code into district layers, each with a specific responsibility:

- Data Layer: Handles data storage and retrieval.
- Processing Layer: Contains logic for manipulation of data.
- Presentation Layer: Manages user input/output.

By structuring code in this way, code modifications can be made in one layer without affecting others, ensuring scalability. Python promotes SoC through this layered architecture as well as modularization and design patterns achieved through the use of classes, functions and decorators (Figure 1).

Figure 1: *Example of Code structure showing a Class, Function and Decorator.*



```
class IO:  12 usages
    '''
    A collection of functions for presenting input/output

    ChangeLog: (Who, When, What)
    HMagnusson, 2025MAR01, Created Class
    '''

    @staticmethod  7 usages
    def output_error_messages(message: str, error: Exception = None):
        '''
        This function displays a custom error message

        ChangeLog: (Who, When, What)
        HMagnusson, 2025MAR01, Created Function

        :param message: String with message data for display
        :param error: Exception object with technical message for display
        :return:
        '''
        print(message)
        if error is not None:
            print("-- Technical Error Message -- ")
            print(error.__doc__)
            print(error.__str__())
```

Classes and Functions

Functions in Python are blocks of reusable code used to performs a specific task. Python has numerous built in functions that require no additional coding, such as the print() function, though in this module user defined functions are introduced. This fundamental programming concept has many benefits:

- Re-usability: A defined function can be recalled and reused multiple times
- Modularity: Allows the breakdown of large complex blocks of code into smaller, independent units. Making them easier to follow, simplifying troubleshooting and debugging
- Encapsulation: A function can operate independently, improving readability and reducing unintended interactions.
- Scalability: using functions allows for easier expansion and modification of code without disrupting the entire program.

Classes are used to organize groups of related data and methods together. This organization follows the principles of Object-Oriented Programming (OOP) that are used to create modular, reusable and easy to read code. Key principles of OOP that classes support include:

- Encapsulation: Building data and methods within a class restricts direct access to internal states, enhancing data security and integrity.
- Inheritance: New classes can derive properties and behaviors from existing ones, reducing code duplication and promoting scalability.

Summary

Understanding and implementing functions and classes effectively is key to writing efficient Python programs. Functions promote modularity and re-usability, while classes provide as structured way to manage related data and methods. By following best practices such as limiting global variables and applying the Separation of Concerns principle, programmers can write code that is both clean and maintainable.