

Jae Barnett

03/05/2025

Foundations of Programming: Python

Assignment 06

<https://github.com/jaeuw>

# Functions and Classes

## Introduction

In this document I will be creating a program that uses constants, variables and print statements to display a message about a student's registration for a Python course. I will also be incorporating three common techniques for improving my scripts with the use of functions, classes, and using the separation of concerns programming pattern.

## Declarations in Python

### Global vs Local Variables

Variables can be categorized as either local or global depending on the location where they are declared. Local variables are declared within a function and can only be accessed/used within that same function. (figure 1) Global variables are declared outside of a function and can be accessed and modified from anywhere in the script. (figure 2)

```
@staticmethod 1 usage
def input_menu_choice():
    """
    This function gets a menu choice from the user
    ChangeLog: (Who, When, What)
    JBarnett,03/05/2025,Created Class
    :return: string with the users choice
    """
    choice = "0"
    try:
        choice = input("Enter your menu choice number: ")
        if choice not in ("1","2","3","4"):
            raise Exception("Please, choose only 1,2,3,4")
    except Exception as e:
        IO_output_error_messages(e.__str__())
    return choice
```

**Figure 1: Local variable example**

```

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

# Processing -----
class FileProcessor:
    """
    A collection of functions that work with json files

    ChangeLog: (Who, When, What)
    JBarnett, 03/05/2025, Created Class
    """

    @staticmethod
    def read_data_from_file(file_name: str, student_data: list):
        """

```

**Figure 2: Global variable example**

## Functions

Functions allow you to group a set of programming statements and later reference those blocks of code by the name you gave them. The two we focused on this week were Modularity and Reusability – per the mod06 notes, they are both defined as the following:

- **Modularity:** Functions allow you to break down a large program into smaller, more manageable pieces. Each function can focus on a specific task, making the code easier to
- **Reusability:** Once you've defined a function, you can use its code multiple times throughout your program without having to rewrite the same code. This promotes code reuse and helps prevent redundancy.

You declare a function by defining it with the “def” keyword followed by the name of the function and the function itself. For example

```

@staticmethod
def read_data_from_file(file_name: str, student_data: list):
    """
    This function reads data from file and converts into json
    :param file_name: string data with file to read from
    :param student_data: list of student data in dictionaries
    :return: list
    """

```

**Figure 3: Defined function named read\_data\_from\_file()**

Another notable keyword is the “pass” keyword which is used as a placeholder statement when Python requires a statement, but you aren’t ready to write one at the time. It is intended to do what it says, to recognize the function but to ignore/pass by and continue until you add code to the function later. An example of this would be:

```
def write_data_to_file():  
    pass
```

## Classes

In Python classes are another way to organize your code by grouping functions, variables and constants under a class name. For this assignment we used two classes, the FileProcessor and IO which house the @staticmethod decorator.

*FILE PROCESSOR:* This class can be found in the processing layer. This class houses functions like opening, reading, writing and closing the json file.

```
class FileProcessor: 2 usages  
    """  
    A collection of processing layer functions that work with Json fi  
    ChangeLog: (Who, When, What)  
    JBarnett,03/05/2025, Created Class  
    """  
  
    @staticmethod 1 usage  
    def read_data_from_file(file_name: str, student_data: list):  
        """  
        This function reads data from file and converts into json  
        ChangeLog: (Who, When, What)  
        JBarnett,03/05/2025, Created Class  
        :param file_name: string data with file to read from  
        :param student_data: list of dictionary rows  
        :return: student data  
        """  
  
        try:  
            file = open(FILE_NAME, "r")  
            student_data = json.load(file)  
            file.close()  
        except FileNotFoundError as e:  
            IO.output_error_messages(message="Text file must exist be  
        except Exception as e:  
            IO.output_error_messages(message="There was a non-specifi  
        finally:  
            if file.closed == False:  
                file.close()  
        return student_data  
  
    @staticmethod 1 usage  
    def write_data_to_file(file_name: str, student_data: list):  
        """  
        This function writes data to file and returns True if  
        """
```

**Figure 4: FileProcessor class**

*IO:* This class can be found in the presentation layer. This class contains the functions that control input and output a user interacts with. For example, error messages, menu options, user input, etc.

```

class IO: 10 usages
    """
    A collection of presentation layer functions that manage user in
    ChangeLog: (Who, When, What)
    JBarnett,03/05/2025,Created Class
    """

    @staticmethod 6 usages
    def output_error_messages(message: str, error: Exception = None):
        """
        This function displays a custom error message to the user
        ChangeLog: (Who, When, What)
        JBarnett,03/05/2025,Created Class
        """
        print(message, end="\n\n")
        if error is not None:
            print("-- Technical Error Message --")
            print(error, error.__doc__, type(error), sep='\n')

    @staticmethod 1 usage
    def output_menu(menu: str):
        """
        This function displays the menu of choices
        ChangeLog: (Who, When, What)
        JBarnett,03/05/2025,Created Class
        :return: None
        """
        print() #Adding extra space to make it look nicer
        print(menu)
        print() #Adding extra space to make it look nicer

    @staticmethod 1 usage
    def input_menu_choice():

```

Figure 5: IO class

## Summary

The classes and functions are useful in creating scripts that are easier to read. I appreciated the explanation of functions because it would prove very useful if I was an outsider reading or reflecting on the program myself. As the program becomes more complex, sectioning each of the segments would prove to be a significant improvement. I also found debugging to be easier when the functions are streamlined.