James Eardley

February 24, 2024

IT FDN 100 A

Assignment 06

Create a Program

Introduction

In this paper I will detail how I created a Python program that builds on the things learned in the prior modules by introducing user defined functions, parameters, classes

Creating the Program

When starting to write a program in any programming language, including Python, it's important to create a header that provides a title of the program, a description of what the program does, and a change log. This is helpful for allowing other users to more clearly understand the purpose of a script and who authored it.

```
# ------------------------------------------------------------------------------------------ #
# Title: Assignment06_Starter
# Desc: This assignment demonstrates using functions
# with structured error handling
# Change Log: (Who, When, What)
#   James Eardley, 2/21/2024, Created Script
# ------------------------------------------------------------------------------------------ #
```

Figure 1.1: Header displaying the title of the assignment, description, and change log.

The number of variables were trimmed as we shifted to the use of user defined functions and defined the variables as parameters in the input of the functions. These variables are global in that they are declared outside of a function and are accessible to the entire script. As opposed to local, which are used inside of a function and only accessible to the function and not the entire script. Return is used to push a local variable to the global environment.

```
# Define the Data Constants
# FILE_NAME: str = "Enrollments.csv"
FILE_NAME: str = "Enrollments.json"

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

Figure 1.2: A much smaller list of variables and constants as a result of the user defined function structure of the program.

The components of the program were rewritten around user defined functions, which allow for functions to be repeated without having to repeatedly script the same code (reusability) and to break it

down into smaller, more manageable pieces (modularity). Changing the instructions in the function changes the way the function is used throughout the program.

A function is begun using "def". The parentheses that end the function are meant to hold the parameters that will be used by the function. You can set default parameters and arguments (values passed to parameters) here. The class of the parameters can be set within the input of the function. The green text enclosed by the triple quotes is a descriptor text which describes the function and when it was created among other information entered by the user. This can be seen in the tool tip.

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

    ChangeLog: (Who, When, What)
    James Eardley,2.21.2024,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 1.3: User defined function uses parameters inside the parentheses. This particular function handles error messaging.

I also used classes, which are a way of grouping functions, variables, and constants by the name of the class. This makes it easier to manage and maintain the code based on a common function or project. The class is defined and then the functions are indented below it.

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

    ChangeLog: (Who, When, What)
    James Eardley,2.21.2024,Created Class
    James Eardley,2.21.2024,Added menu output and input functions
    James Eardley,2.21.2024,Added a function to display the data
    James Eardley,2.21.2024,Added a function to display custom error messages
    """
    @staticmethod
    def output_error_messages(message: str, error: Exception = None):
        """ This function displays the a custom error messages to the user

        ChangeLog: (Who, When, What)
        James Eardley,2.21.2024,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 1.4: I defined an IO (input, output) class of functions for all of the input and output based functions. The functions are indented below the class definition, which can also contain a description and audit of the changes to the function group. This information is also available via the tooltip.

The end result is a much easier to read main body of the script which uses the same while loop and conditional statements, but calls the individual functions.

```
#--------- Beginning of the main body of this script ----------------------- #

# 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):

    IO.output_menu(menu=MENU)

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

        IO.input_student_data(student_data=students)
        continue

    # Present the current data
    elif menu_choice == "2":

        IO.output_student_courses(student_data=students)
        continue

    # Save the data to a file
    elif menu_choice == "3":

        FileProcessor.write_data_to_file(file_name=FILE_NAME, student_data=students)
        continue

    # Stop the loop
    elif menu_choice == "4":
        break  # out of the loop
    else:
        print("Please only choose option 1, 2, or 3")

print("Program Ended")
```

Figure 1.5: I defined an IO (input, output) class of functions for all of the input and output based functions. The functions are indented below the class definition, which can also contain a description and audit of the changes to the function group. This information is also available via the tooltip.

Testing the Program

When done correctly, the script runs, prompting the user with a menu of items to choose from. Choosing 1 allows the user to register a student with a class. This can be run over and over to allow for the registration of multiple students. Action 2 displays the current data that has been input. Action 3 sabes the data to a csv. Finally, action 4 breaks the loop and exits the menu. Action steps 1 and 3 now utilized error handling.

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



Enter your menu choice number: 5
Please, choose only 1, 2, 3, or 4

Please only choose option 1, 2, or 3


---- 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.
----------------------------------------



Enter your menu choice number: 1

Enter the student's first name: j!
That value is not the correct type of data!

-- Technical Error Message --
The last name should not contain numbers.
Inappropriate argument value (of correct type).
<class 'ValueError'>


---- 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.
----------------------------------------



Enter your menu choice number: 2

-------------------------------------------------
Student Bob Smith is enrolled in Python 100
Student Sue Jones is enrolled in Python 100
Student James Eardley is enrolled in Python 100
Student Chloe Palmer is enrolled in Python 200
Student James Eardley is enrolled in Python 200
-------------------------------------------------


---- 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.
----------------------------------------



Enter your menu choice number: 3


---- 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.
----------------------------------------



Enter your menu choice number: 4
Program Ended
```

Figure 1.7: The output of the script as displayed in the Console of the IDE. I tested the error handling by including a number in the first name.

The script was also run in cmd prompt to ensure that the program could be run in multiple ways.

```
---- 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
What is the student's first name? james
What is the student's last name? eardley
Please enter the name of the course: python
You have registered james eardley for python.

---- 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
--------------------------------------------------
First Name: James, Last Name: Eardley, Course: Python 100
First Name: Chloe, Last Name: Palmer, Course: Python 200
First Name: james, Last Name: eardley, Course: python
--------------------------------------------------

---- 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: 3

---- 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
```

Summary

Assignment 06 built on the prior assignments by including the use of user defined functions, classes, and parameters. The enhancements take a large step toward making the scripts more organized, manageable, and readable.