Danielle Lilleston
Python 100

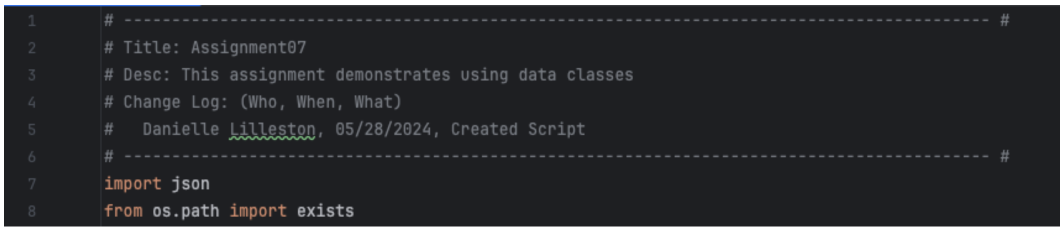
Assignment 07

Introduction

This assignment, just as the prior assignments, is a script written as a course registration system in python. However, unlike the previous assignment, this script demonstrates the use of classes.

Step One: Beginning the Script

As in all scripts, we always start with a script header. This header will always be important, and it lists the title of the script, the description, and the change log. This should always be at the beginning. Following is the import of the json file. We need to import this file as it will be called out in the script and later used to save the data too.

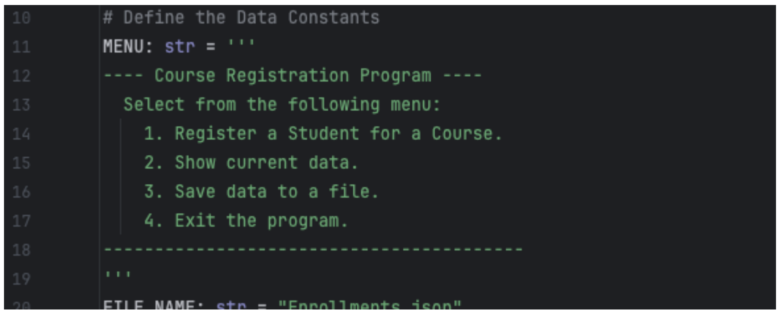


```python
# ------------------------------------------------------------------------------------------ #
# Title: Assignment07
# Desc: This assignment demonstrates using data classes
# Change Log: (Who, When, What)
#   Danielle Lilleston, 05/28/2024, Created Script
# ------------------------------------------------------------------------------------------ #
import json
from os.path import exists
```

Figure 1. This shows the script header used. It also shows the callout to import the json file on

Step Two: Constants and Variables

In this script we will be using the same constants and variables as previously found in assignment 06. The constants will be the MENU and FILE_NAME, which are both strings. The variables are the students, which is a list, and the menu choices, which is a string.

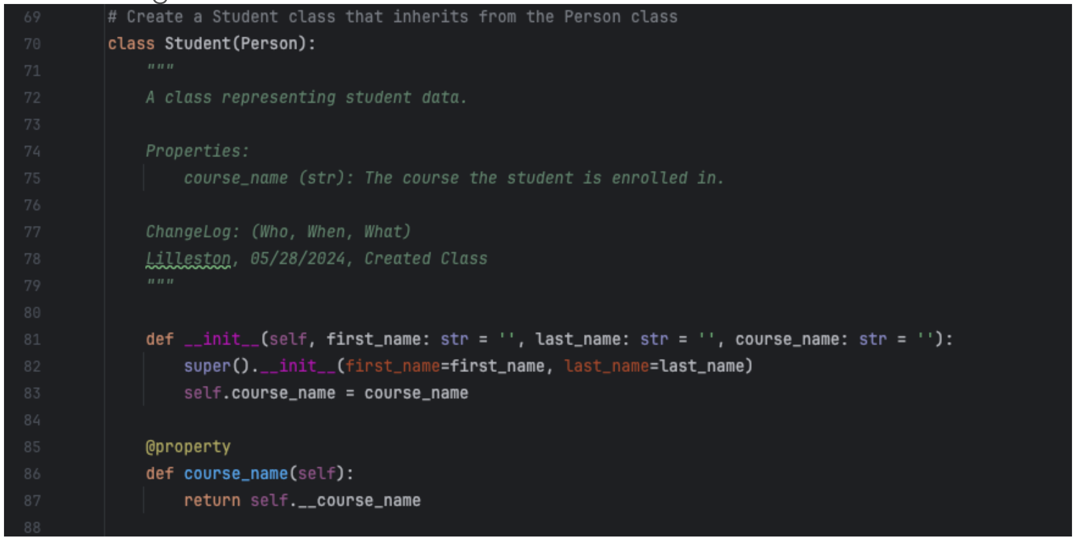


```python
# Define the Data Constants
MENU: str = '''
---- Course Registration Program ----
  Select from the following menu:
    1. Register a Student for a Course.
    2. Show current data.
    3. Save data to a file.
    4. Exit the program.
-----------------------------------------
'''
FILE_NAME: str = "Enrollments.json"
```

Figure 2. This shows the variables and constants used in this script. I should call out that constants are

Step Three: Creating the Classes

In this script we have set four different classes, “Person”, “Students”, and “FileProcessor”, and “I0”. Classes provide a blueprint for creating objects in an organized and manageable way. How does one set up a class? First, we must define the class, let’s use “Person” as an example. Just like a script header providing information, we also start the class using a string comment to define what the class is, in this case it is the persons data, and its properties, which in this case is the students last name and first name. It also needs to include a change log as classes are flexible and can be changed over time if needed. Classes are also useful because we can create other classes that can inherit properties from another class. In this script, “Students” is a subclass of “Person”. Also, in the class “FileProcessor” there is also a function included. Within classes, we can have functions. The class “I0” is a collection of presentation layer functions that manages the users input and output. This is basically calling out to the menu choices and is telling the user to pick between options 1,2,3, or 4 and writing in error messages if this is not followed.

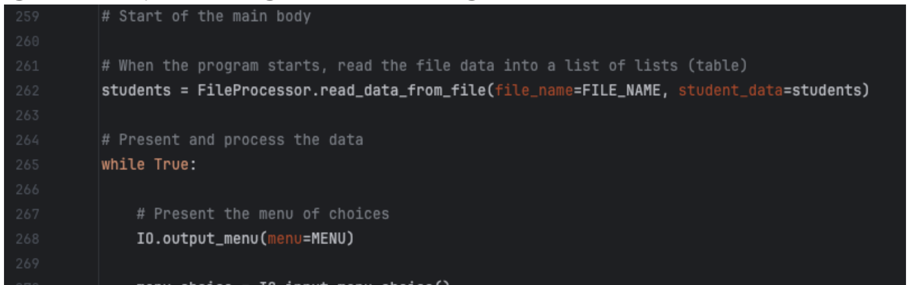


```
# Create a Student class that inherits from the Person class
class Student(Person):
    """
    A class representing student data.

    Properties:
        course_name (str): The course the student is enrolled in.

    ChangeLog: (Who, When, What)
    Lilleston, 05/28/2024, Created Class
    """

    def __init__(self, first_name: str = '', last_name: str = '', course_name: str = ''):
        super().__init__(first_name=first_name, last_name=last_name)
        self.course_name = course_name

    @property
    def course_name(self):
        return self.__course_name

```

Figure 3. This shows the “Student” Class. It is just a snippet of the entire code used to form the class but it shows how it has inherited characteristics from the “Person” class as called out from the comment. It also shows the string in the beginning explaining what the class is and

Step 4: Main Body of the Script

The main body of the script is when it starts to process all the information. Funny enough, this part of the script is very small. The bulk of the script is when setting up the classes. In the main body, we ask the user to choose between the menu options provided. As the user chooses, it will process the data and save it into a json file, the one called out in the beginning. The reason we will this the main body is because it is the interactive part of the script that the user will experience, either entering in data, presenting data, or saving data to a file.



```
# Start of the 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)

    menu_choice = IO.input_menu_choice()
```

Figure 4. This is a snippet of the main body of the script. It shows the user the different menu options and the execution

Summary

This assignment is the same as past assignments with the utilization of different options that are provided in python. Although more complex, they do almost the same thing. These complexities are useful though as they allow for more manageable code, its flexible and can be changed, provides organization, and it helps build a much more solid script. They are definitely getting more challenging as time goes on.