GJohnson

8/11/2024

Foundations Of Programming: Python

Assignment07

Create a Program

Introduction

The goal of this document will be to demonstrate the use of python to create a program using set of data classes all while using PyCharm as your IDE.

Creating the script

Acceptance Criteria

Understanding the acceptance criteria to ensure successful development is crucial to the success of any python project. The acceptance criteria for this script requires the following:

1) The file is named Assignment07.py
2) Script Header: “Name” and “Current Date”
3) Constants:
    a. **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
-----------------------------------------
```

    b. **FILE_NAME: str** is set to the value "Enrollments.json"
    c. Constants values do not change throughout the program.
4) Variables:
    a. **menu_choice: str** is set to empty string.
    b. **students: list** : list is set to and empty list
5) Classes:
    a. The program includes a class named FileProcessor.
    b. The program includes a class named IO.
    c. The program includes a class named Person.

   d. The program includes a class named <u>Student</u>.
   e. All classes include descriptive document strings

6) Class Properties:
   a. The program includes properties for student_first_name: str and defaults to an empty string
   b. The program includes properties for student_last_name: str and defaults to an empty string
   c. The program includes properties for course_name: str and defaults to an empty string
   d. The program's properties must include simple validation code.
7) Class Methods:
   a. The program includes a method to extract comma separately data from each data class
8) Functions:
   a. All functions include descriptive document strings.
   b. All functions with except blocks include calls to the function handling error messages.
   c. All functions use the @staticmethod decorator.
   d. The program includes functions with the following names and parameters:
      i. output_error_messages(message: str, error: Exception = None)
      ii. output_menu(menu: str)
      iii. input_menu_choice()
      iv. output_student_courses(student_data: list)
      v. input_student_data(student_data: list)
      vi. read_data_from_file(file_name: str, student_data: list):
      vii. write_data_to_file(file_name: str, student_data: list):
9) Input & Output
   a. **Choice 1**: Prompts user to enter students first and last name and course_name using input() and stores the inputs in the respective variables
   b. **Choice 2**: Outputs string using print() function
   c. Data collected from menu choice 1 is added to a two-dimensional list table (list of dictionaries)
   d. All data from the list is displayed when menu choice 2 is selected
10) File Processing
   a. When the program starts, the two-dimensional list is displayed
   b. **Choice 3**: program opens a file names “Enrollments.json” in write mode using the open() function. It writes the content of the students variables to the file using the dump() function then closes file using the close() method. It displays what was stored in the file.
   c. **Choice 4**: ends program
11) Error Handling

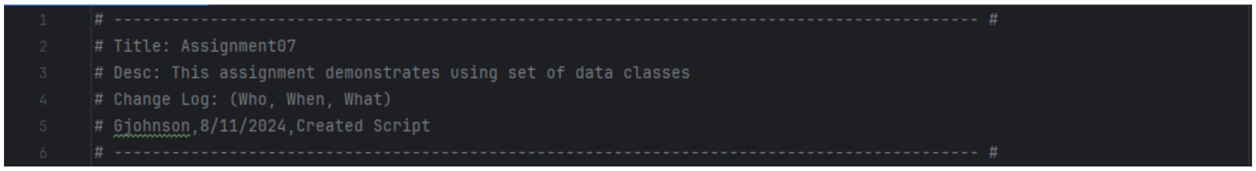
a. The program provides structured error handling when the file is read into the list of dictionary rows.
b. The program provides structured error handling when the user enters a first name.
c. The program provides structured error handling when the user enters a last name.
d. The program provides structured error handling when the dictionary rows are written to the file.

12) Output expectation
   a. Accepts user input for student's first, last and course name
   b. Displays user input for student's first, last and course name
   c. Saves user input for student's first, last and course name as a comma-separated string file
   d. Program allows for user to input multiple registrations
   e. Program allows users to display multiple registrations
   f. Program allows users to save multiple registrations
   g. Program runs correctly in PyCharm and from console or terminal

13) Source Control
   a. The script file and the knowledge document are hosted on a GitHub repository.
   b. A link to the repository is included in the knowledge document.
   c. A link to the repository is included in the GitHub links forum.

File Name and Header

In the first steps of creating this script, a header was completed, detailing the title of the assignment, name and date (Figure 1). The file was then saved as "Assignment07_Name".

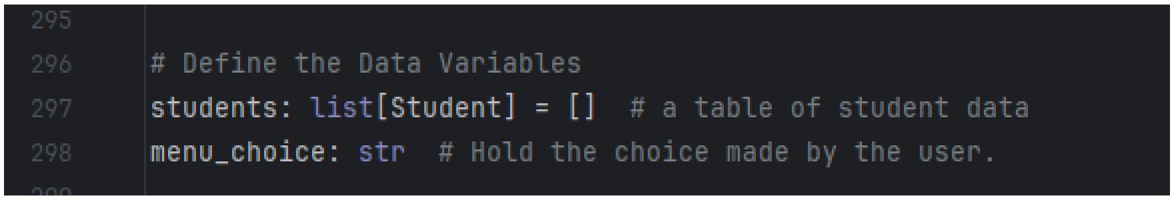


```
# ------------------------------------------------------------------------------------------ #
# Title: Assignment07
# Desc: This assignment demonstrates using set of data classes
# Change Log: (Who, When, What)
# Gjohnson,8/11/2024,Created Script
# ------------------------------------------------------------------------------------------ #
```

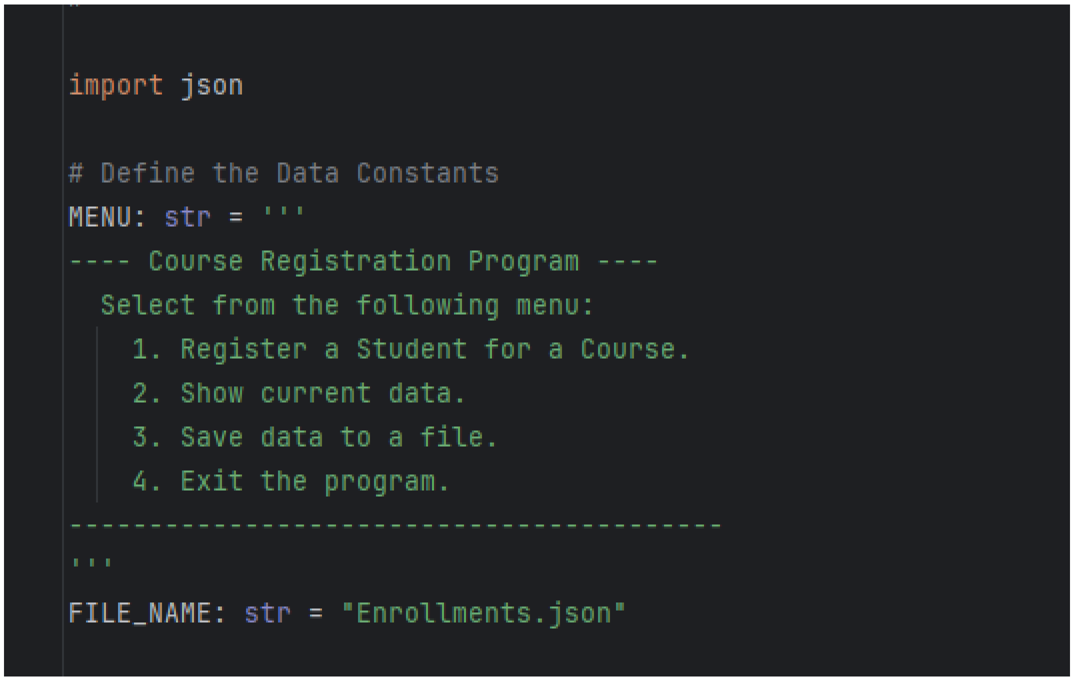
Figure 1: Example of Python Header

Constants and Variables

To meet 3 and 4 of the acceptance criteria, constants and variables were written into the code with their correlating data types- to meet acceptance criteria, including the use of two dimensional list table, however, to meet code standard, the variables were placed lower in the code (Figure 2).



```

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



```
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"
```

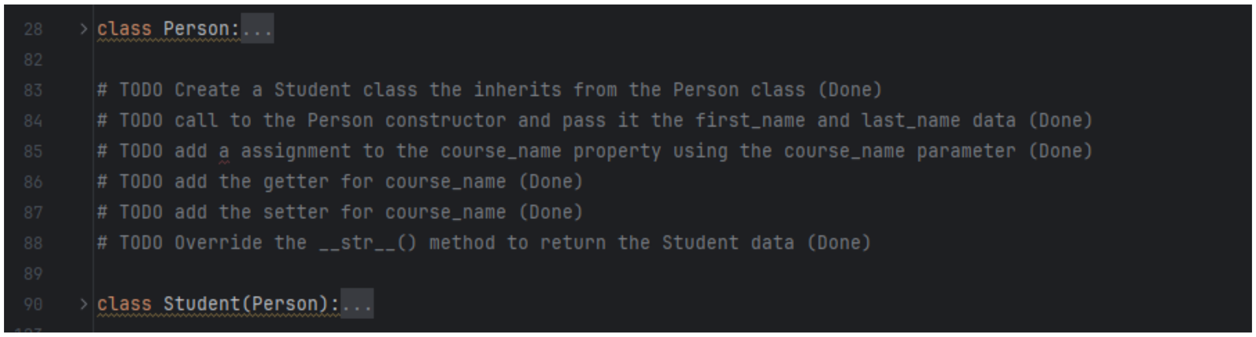
Figure 2: Constants & Variables Development

Classes and Functions

After completing the basic setup for the code, classes- which helps bundle data and functionality together, while detangling complex code- were added. FileProcessor was added in to handle all parts of the program dealing with read and write privileges to the program. Under FileProcessor the following functions were added: write_data_to_file and read_data_to_file.

For IOProcessor, all code relating to print statements, including error handling was added to this class. This included the following functions: output_error, output_menu, input_menu_choice, output_student_data and input_student_data.

Class for Person and Student were also created to break up the code and produce a method to capture more than just students, while streamlining the code needed. Properties and setters were also used to produce simple validations(Figure 3).



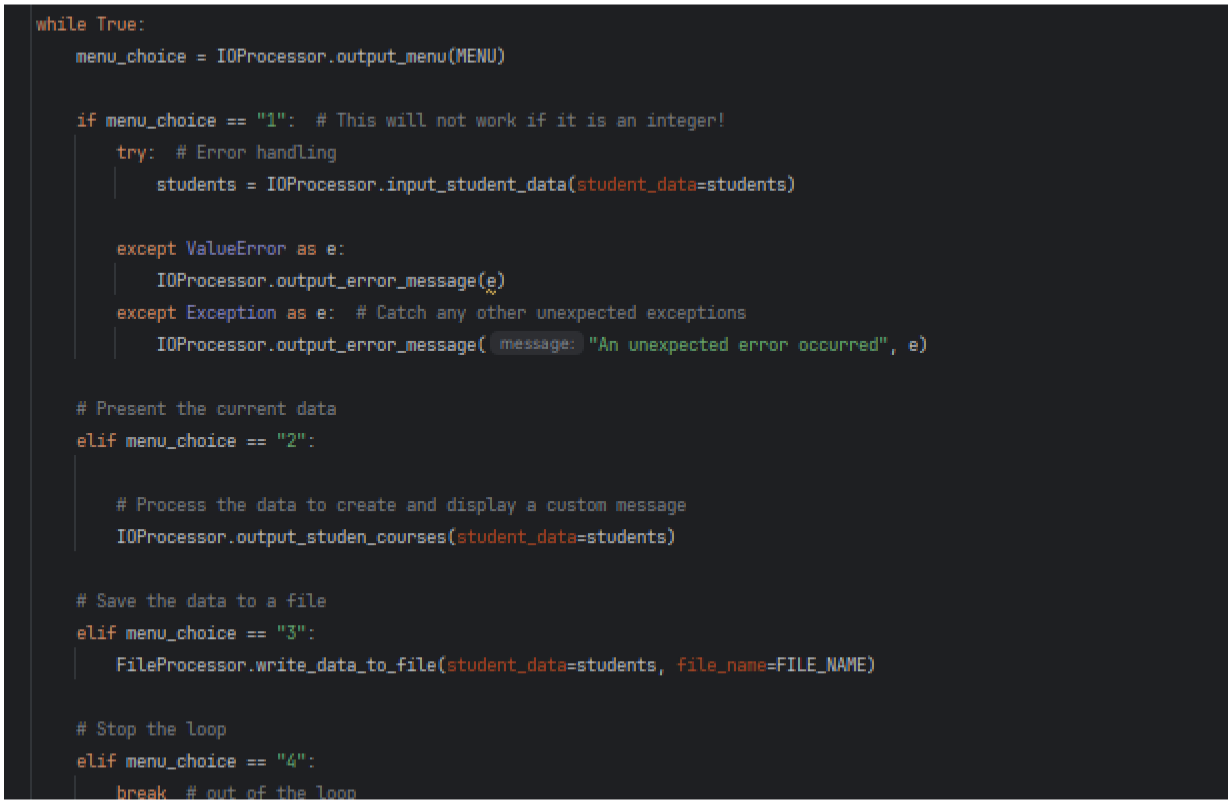
```
> class Person:...

  # TODO Create a Student class the inherits from the Person class (Done)
  # TODO call to the Person constructor and pass it the first_name and last_name data (Done)
  # TODO add a assignment to the course_name property using the course_name parameter (Done)
  # TODO add the getter for course_name (Done)
  # TODO add the setter for course_name (Done)
  # TODO Override the __str__() method to return the Student data (Done)

> class Student(Person):...
```

Figure 3: Classes

Input and Outputs

For the input and output acceptance criteria, the input() function and output() functions were used. Each data type was labeled while also assigning values to our constants. Data collected from option 1 was stored in a two-dimensional list table. Error handling was also added to include errors for when characters other than alpha's were used, as well as adding the ability to restart the program instead of ending due to an error.

For option number 2, the code recalls all inputs and provides the list to users (Figure 4).



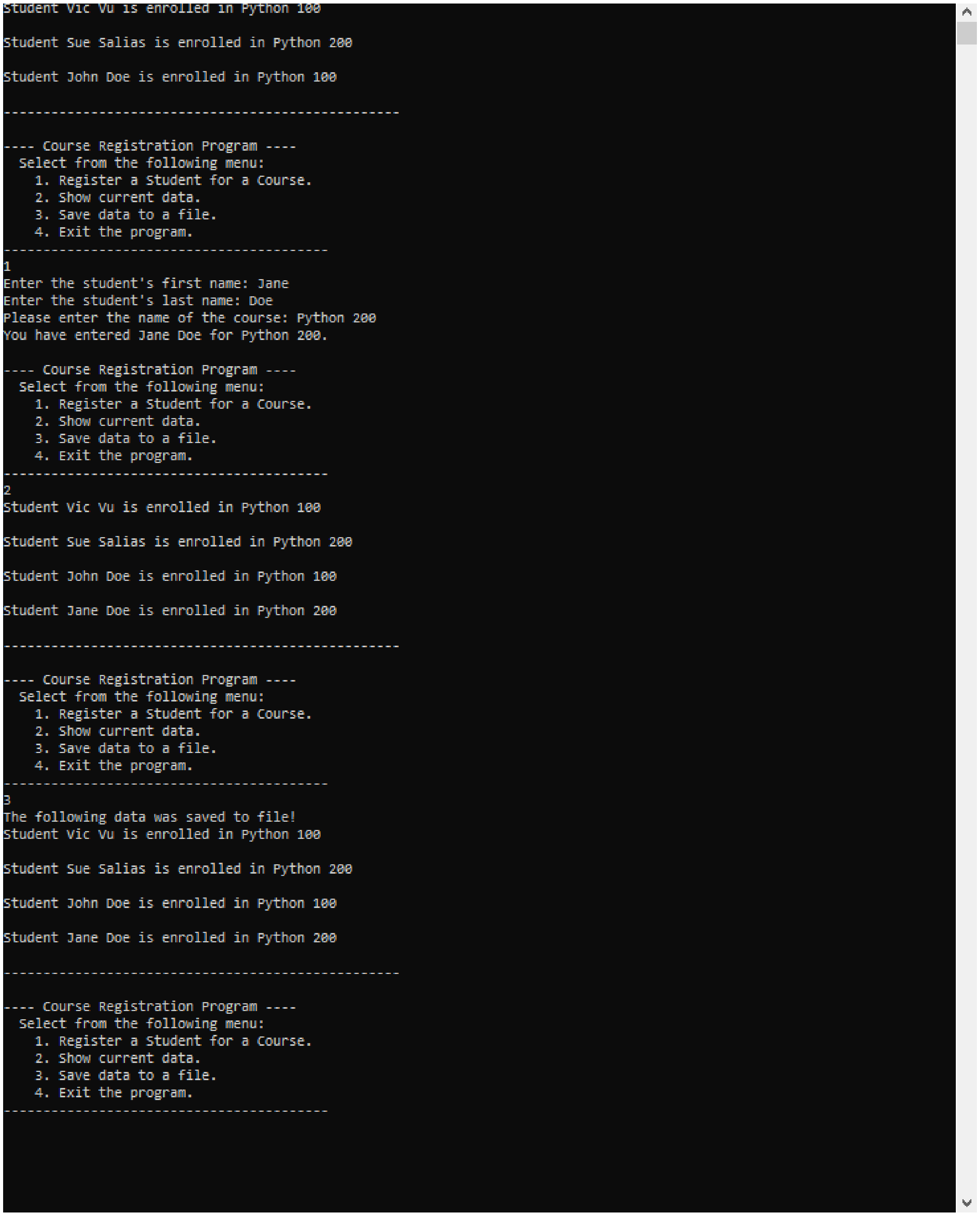
```
while True:
    menu_choice = IOProcessor.output_menu(MENU)

    if menu_choice == "1":  # This will not work if it is an integer!
        try:  # Error handling
            students = IOProcessor.input_student_data(student_data=students)

        except ValueError as e:
            IOProcessor.output_error_message(e)
        except Exception as e:  # Catch any other unexpected exceptions
            IOProcessor.output_error_message( message: "An unexpected error occurred", e)

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

        # Process the data to create and display a custom message
        IOProcessor.output_studen_courses(student_data=students)

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

    # Stop the loop
    elif menu_choice == "4":
        break  # out of the loop
```

Figure 4: Input, Output & Values

File Processing

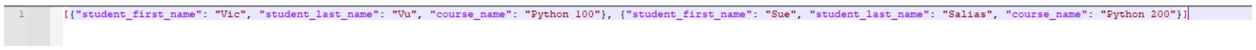
Lastly, to complete the script and meet all acceptance criteria, methods to open, write and close were created in the script- using the open(), dump() and close() functions. Error handling was added to ensure if there were specific JSON and or other exceptions errors- this was added as a function in IOProcessor- that helpful information to users would be provided, while also preventing the program from crashing. Additional error help was added when starting the program. Code was added to ensure if the file was empty, the program was not ended. Details were added in case the file name was not found, and was built to provide clear feedback.

With all elements present (Figure 5) use of command prompt was used to ensure that our code meets the output expectations (Figure 6) and the file was created with the correct values (Figure 7).



```
Student Vic Vu is enrolled in Python 100

Student Sue Salias is enrolled in Python 200

Student John Doe is enrolled in Python 100

--------------------------------------------------

---- 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.
-----------------------------------------
1
Enter the student's first name: Jane
Enter the student's last name: Doe
Please enter the name of the course: Python 200
You have entered Jane Doe for 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.
-----------------------------------------
2
Student Vic Vu is enrolled in Python 100

Student Sue Salias is enrolled in Python 200

Student John Doe is enrolled in Python 100

Student Jane Doe 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.
-----------------------------------------
3
The following data was saved to file!
Student Vic Vu is enrolled in Python 100

Student Sue Salias is enrolled in Python 200

Student John Doe is enrolled in Python 100

Student Jane Doe 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.
-----------------------------------------
```

Figure 5: Example of Command Prompt Use



```
[{"student_first_name": "Vic", "student_last_name": "Vu", "course_name": "Python 100"}, {"student_first_name": "Sue", "student_last_name": "Salias", "course_name": "Python 200"}]
```

Figure 6: Enrollments JSON

Source Control

Source control for code is an important part of development and a way to ensure revisions are tracked while protecting in case a roll back to previous code is needed. For this project, GitHub was used, storing the repository here.

Summary

The use of python to create programs with loops and conditional logic is an example of how python can serve in any business or individual use case. By adding in classes, your code will become more organized, more streamlined and allow for more application flexibility.

Appendix

What is a "Header"

A python header can be used in various ways but serves the main purpose of providing details to a user related to the code. In above use (Figure 1), the header is used to document the title of the project, a description of what the code does and a log to track the changes to the project and code.

What is a "Constant"

A constant is a method to store data or information in python. A constant is documented in all upper-case and is generally meant to be unchanged throughout a program's lifecycle.

What is a "Variable"

A variable is a method to store data or information in python. A variable is documented in all lower-case characters and is generally okay to update throughout a program's lifecycle.