Denesse Citlaly Gomez

June 4, 2024

IT FDN 110A

Assignment 07

Creating Python Script Using Programming Tools and Techniques: Classes and Objects

Introduction

For assignment 7, the task was like that of assignment 6 where I had to write a Python Script that would allow the user to choose menu options and have different outputs based on the menu choice chosen by the user. Like assignment 6, input and output functions, classes, error handling, and while loops were used; however, for this assignment, a set of data classes was used. These various programming techniques/tools gave structure to my Python Script overall.

Creating Python Script for Task

Define Constants

The Python script for this assignment began similar to previous assignments. For this portion, the JSON file needed to have existing data for the code to fully run properly. At first, I was running across issues because my JSON file was originally blank, and Python could not retrieve any data from it. Thus, for this Python Script, it is important to ensure that there is existing data in the JSON file for Python to retrieve and load for the menu 2 option, if not an error will appear. The structured error handling statements imbedded within the Python Script help prevent the code from crashing.

Image 1: Define Data Constants

```
import json #Note: There must be an existing data for the code to run

# 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" #Note: Constant values do not change throughout the program
```

Structured Error Handling

Structured error handling is a technique that was previously utilized in previous assignments. An example of how it was used in this assignment is seen in Images 2 and 3. As seen in Image 2, I input a number other than 1-4, and the code output was “Please only choose option 1, 2, or 3”.

Image 2: Output for Incorrect Value Input

```
---- 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: 34
Please, choose only 1, 2, 3, or 4

Please only choose option 1, 2, or 3
```

As you can see for this portion of code (Image 3), the output is within the “try” block. Since the choice was not within the strings listed in line 236, the code moved to the “except” block where the “Please, choose only 1, 2, 3, or 4” message is written. This exception is the code’s way of registering an unexpected error.

Image 3: Example of Structure Error Handling Statements

```
224     @staticmethod
225     def input_menu_choice():
226         """ This function gets a menu choice from the user
227
228         ChangeLog: (Who, When, What)
229         DGomez, 6/3/24,Created function
230
231         :return: string with the users choice
232         """
233         choice = "0"
234         try:
235             choice = input("Enter your menu choice number: ")
236             if choice not in ("1","2","3","4"):  # Note these are strings
237                 raise Exception("Please, choose only 1, 2, 3, or 4")
238         except Exception as e:
239             IO.output_error_messages(e.__str__())  # Not passing e to avoid the technical message
240
241         return choice
```

Set Of Data Classes

A set of classes helps when you are trying to reuse certain attributes and methods throughout your code and helps with the overall program structure. In this case, a set of classes was utilized in this Python script as seen in Images 4 and 5. The class named “Person” is the superclass; whereas the class named “Student” is considered to be the “subclass”. This means that the “Person” class is the base class, and the “Student” class will be obtaining attributes and methods from the “Person” class. In the code, the “@property” was used to make methods act like attributes. By using a “setter”, the code will essentially

be validating input data before it assigns it to a specific attribute and in this way, it is able to provide control within the program.

Image 4: Class Named Person

```
class Person:
    def __init__(self,first_name:str,last_name:str):
        self.first_name=first_name
        self.last_name=last_name #Note: "_" makes them protected properties

    3 usages (2 dynamic)
    @property
    def first_name(self)->str:
        """
        Returns the first_name as a title
        :return: The first name, properly formatted
        """
        return self._first_name.title()

    3 usages (2 dynamic)
    @first_name.setter
    def first_name(self,value:str) -> None:
        """
        Sets the first name while doing validations
        :param value: The value to set
        """
        if value.isalpha():
            self._first_name=value
        else:
            raise ValueError("First name must be alphabetic")

    @property
    def last_name(self)->str:
        """
        Returns the last_name as a title
        :return: The last name, properly formatted
        """
        return self._last_name.title()

    2 usages (1 dynamic)
    @last_name.setter
    def last_name(self, value:str) -> None:
        """
        Sets the first name while doing validations
        :param value: The value to set
        """
        if value.isalpha():
            self._last_name = value
        else:
            raise ValueError("Last name must be alphabetic")

    def __str__(self):
        """
        The string function for Person
        :return: The string as a csv value
        """
        return f'{self.first_name},{self.last_name}'
```

In addition, the class named “Student” is considered a subclass since it obtains the attributes and function methods from the class named “Person”. This is denoted by the “class Student(Person)” seen in Image 5. Within this code, the “self” is used to refer to attributes and methods that belong to the class it is being called in.

Image 5: Class Named “Student”

```
class Student(Person):
    def __init__(self,first_name:str,last_name:str,course_name:str):
        super().__init__(first_name,last_name)
        self._course_name=course_name

    4 usages
    @property
    def course_name(self)-> str:
        """
        Returns the course_name
        :return: course_name
        """
        return self._course_name
    @course_name.setter
    def course_name(self,value)-> None:
        self._course_name=value

    def __str__(self)-> None:
        """
        The string function for Person
        :return: The string as a csv value
        """
        return f'{super().__str__()},{self.course_name}'
```

Validation Code

Furthermore, validation codes were also used in this code to ensure that a technical message is displayed with the reasoning behind the error. Validation codes are used to check that data or inputted data is correct before proceeding with the rest of the program. For instance, in the script, I purposely include incorrect values to test out the functionality of these validation code lines. As seen in image 6, line 52 is an example of one of the validation codes used throughout the Python code. If there is an incorrect user input being entered for the requested information, then the program will provide reasoning. For instance, in this case, I tested the validation code by inputting numbers for the first and last names as shown in Image 7. The program did not let me continue until I had started the first and last name input with an alphabetical letter.

Image 6: Validation Code in the Python Script

```
    3 usages (2 dynamic)
43      @first_name.setter
44      def first_name(self,value:str) -> None:
45          """
46          Sets the first name while doing validations
47          :param value: The value to set
48          """
49          if value.isalpha():
50              self._first_name=value
51          else:
52              raise ValueError("First name must be alphabetic")
```

Image 7: Validation Code Output

```
Enter your menu choice number: 1
Enter the student's first name: 56774
Enter the student's last name: 5657895
Please enter the name of the course: 33235
One of the values was the correct type of data!

-- Technical Error Message --
First name must be alphabetic
Inappropriate argument value (of correct type).
<class 'ValueError'>
```

Python Script Verification

To verify my Python Script, I used both PyCharm and the Command Prompt. Through the process, I came across various errors, but I believe the debugging tool on PyCharm was helpful when troubleshooting. I was able to successfully run the Python script in both PyCharm and Command Prompt.

Summary

Overall, by working on this assignment, I realized how crucial running the code in pieces could be because it was easier to find my errors and adjust accordingly. One of the reasons why I had so many errors while writing my code is that Python is case-sensitive, and I would incorrectly place certain spaces. I also learned about a set of data classes and how they are used to manage data that is being inputted by the user.

Link to GitHub Repository:

Resources

1. Mod07-Notes