Denesse Citlaly Gomez

May 31, 2024

IT FDN 110A

Assignment 06

Creating Python Script Using Programming Tools and Techniques: Functions

Introduction

For assignment 6, I was tasked to create a Python code with similar outputs to that of assignment 5. The code will still have the four menu choices (1-4) for the user to choose from to perform several actions, such as registering a student for a course, showing current input data, saving data to a JSON file, and exiting the program once done. The new main tools applied to this assignment are both function and classes. In this assignment, I found myself testing portions of the code every step of the way to ensure that my script lines were giving the outputs that I was expecting. Testing the code piece by piece made this assignment a lot easier to troubleshoot when errors occurred.

Creating Python Script for Task

Define Constants and Variables

Similar to scripts written for past assignments, I started by defining the constants and variables as seen in Image 1. Once again, I present the menu choices for the user and establish dictionaries, lists, and constants that I will be using throughout the code to obtain the outputs desired for the functionality of the code.

Image 1: Define Constants and Variables

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


# Define the Data Variables and constants
student_first_name: str = ''  # Holds the first name of a student entered by the user.
student_last_name: str = ''  # Holds the last name of a student entered by the user.
course_name: str = ''  # Holds the name of a course entered by the user.
student_data: dict = {}  # one row of student data
students: list = []  # a table of student data
csv_data: str = ''  # Holds combined string data separated by a comma.
json_data: str = ''  # Holds combined string data in a json format.
file = None  # Holds a reference to an opened file.
menu_choice: str  # Hold the choice made by the user.
```

Functions

In addition, I utilized functions for a part of my Python script. Functions are blocks of reusable code that are used to perform a specific action within the Python script. In this script, I found it useful to use functions as they provide a means of organizing the code within logical blocks. This will inevitably help make the code easier to read and understand. Image 2 demonstrates an example of how I utilized functions within my Python Script. I started by using “def” to define the function that would be used. In this case, the function name is “read_data_from_file” and the parameters being used are the “file_name” and “student_data”. Parameters are added to a function to make the code easier to work with since these parameters allow me to pass data into a function call. I also used a return statement towards the end of this function. A return statement is used to specify which value will be returned after the function is called. The function will exit once the return statement has been performed. In this case, I used the return statement to return the list of student_data that was to be read by this function. Both parameters and return statements are optional, but in this case I used them both.

Image 2: Function within the Python Script

```
47      def read_data_from_file(file_name:str, student_data: list):
48          """ This function reads data from a json file and loads it into a list of dictionary rows
49
50                  ChangeLog: (Who, When, What)
51                  Denesse Citlaly Gomez,5/30/24,Created Class
52
53                  :return: list
54                  """
55
56          try:
57              file = open(file_name, "r")
58              student_data = json.load(file)
59              file.close()
60          except Exception as e:
61              IO.output_error_messages(message="Error: There was a problem with reading the file.", error=e)
62
63          finally:
64              if file.closed == False:
65                  file.close()
66          return student_data
```

Classes

Furthermore, I used classes within my Python Script. The functions were used within these classes that were created. A class is used to create an object in Python, specifically a class will define how an object will behave and have attributes and methods associated with it. I found classes useful to organize my code as I grouped related functions/data. For my script, I used classes to have the script manage user input and output and to work with JSON files. Image 3 shows how I used classes in this Python Script to manage the user input and output.

Image 3: Class IO (for user input and output)

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

    ChangeLog: (Who, When, What)
    Denesse Citlaly Gomez,5/30/24,Created Class
    """
    pass
```

Within this class IO, I embedded functions to display custom error messages to the user, to display menu choices to the user, to receive menu choices from the user, to display the student's name and course, and to obtain input from the user. Image 4 shows the function written to obtain a menu choice from the user. This function included error handling just in case the user inputs anything other than the "1,2,3,4" from the provided menu options. A return statement was added to this function to return the menu choice that was chosen by the user.

Image 4: Function in Class IO

```
1 usage
@staticmethod
def input_menu_choice():
    """ This function gets a menu choice from the user

    :return: string with the users choice
    """
    choice = "0"
    try:
        choice = input("Enter your menu choice number: ")
        if choice not in ("1","2","3","4"):  # Note these are strings
            raise Exception("Please, choose only 1, 2, 3, or 4")
    except Exception as e:
        IO.output_error_messages(e.__str__())  # Not passing e to avoid the technical message

    return choice
```

In this Python Script, I also created a class for working with JSON files. The functions added to this class served the purpose of reading data from the JSON file, loading it into the student_data list, and writing the data from a list of dictionary rows to the JSON file.

Image 5: Class FileProcessor (to work with JSON files)

```
# Processing ------------------------------------- #
2 usages
class FileProcessor:
    """
    A collection of processing layer functions that work with Json files

    ChangeLog: (Who, When, What)
    Denesse Citlaly Gomez,5/30/24,Created Class
    """
```

Using Data from Classes into While Loop

The main reason classes were used in this Python Script was to provide structure to the code and make it easier to read and make modifications if I were to want to adjust the attributes/behavior of the code. Image 6 shows how the class IO was used inside a while loop. This is the main body of the script, and yet it looks simple because the functions were written within the classes towards the beginning of this code, and I just retrieved data from within the classes for this portion. For example, for the first menu choice, I retrieved data from the class IO to present the menu of choice. If statements were used to provide the various menu choices to the user.

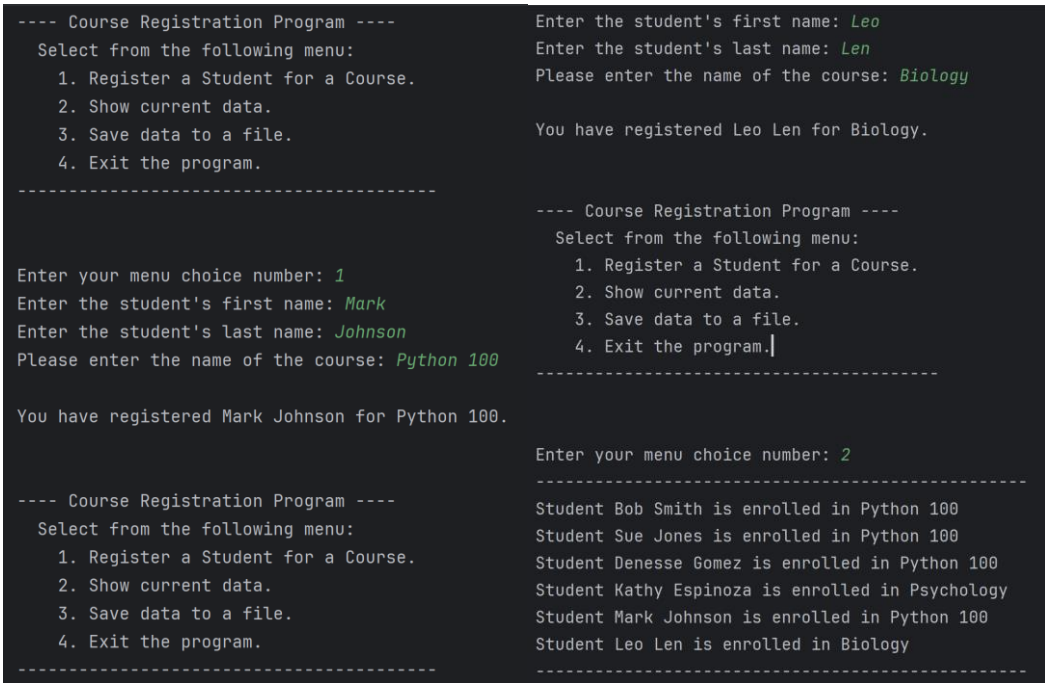
Image 6: While Loop in Python Script

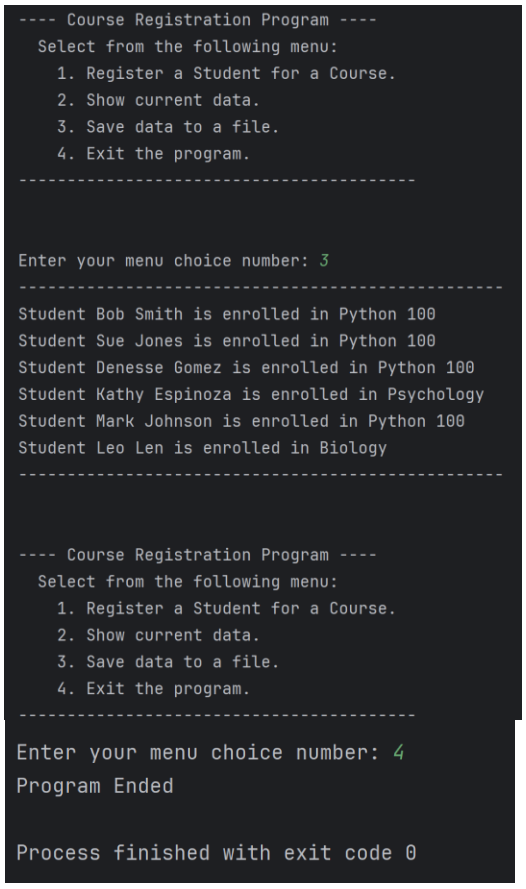
```
196  # When the program starts, read the file data into a list of lists (table)
197  # Extract the data from the file
198
199  students= FileProcessor.read_data_from_file(file_name=FILE_NAME, student_data=students)
200  # Present and Process the data
201  while (True):
202
203      # Present the menu of choices
204      IO.output_menu(menu=MENU)
205
206      menu_choice = IO.input_menu_choice()
207
208      # Input user data
209      if menu_choice == "1":  # This will not work if it is an integer!
210
211          students = IO.input_student_data(student_data=students)
212          continue
213
214      # Present the current data
215      elif menu_choice == "2":
216
217          IO.output_student_and_course_names(students)
```

Python Script Verification

To verify my Python Script, I used both PyCharm and the Command Prompt. Image 7 shows the verification using PyCharm. I was able to successfully run my Python Script since it generated all the required outputs and executed properly.

Image 7: Python Script Verification using PyCharm





Summary

One thing I found truly useful within PyCharm is the debugging tool because I was able to troubleshoot a lot faster with its help. Since I was testing the code in pieces as I progressed through writing the script, this debugging tool allowed me to locate errors within my code. For this specific code,

I found myself testing every single grouped portion using breakpoints to ensure that I could locate an error while writing the code instead of waiting until the end to troubleshoot this long code. Overall, I used functions and classes, among other techniques that I have learned throughout the course, to create a Python script that was easier to read and understand and be able to maintain if I were to want to make modifications later.

Link to GitHub Repository:

Resources

1. Mod06-Notes