Nick Greco
Nov 13, 2024
Foundations of Programming: Python
Assignment05
https://github.com/NBGreco/IntroToProg-Python-Mod05/tree/main

Advanced Data Collections

Introduction

This assignment explains how to create a script for a basic student enrollment system using PyCharm IDE (Integrated Development Environment). The script builds on previous assignments by incorporating data collections to store and recall enrollment data. In Python, collections of data refer to built-in data structures that allow you to store, organize, and manipulate multiple items.

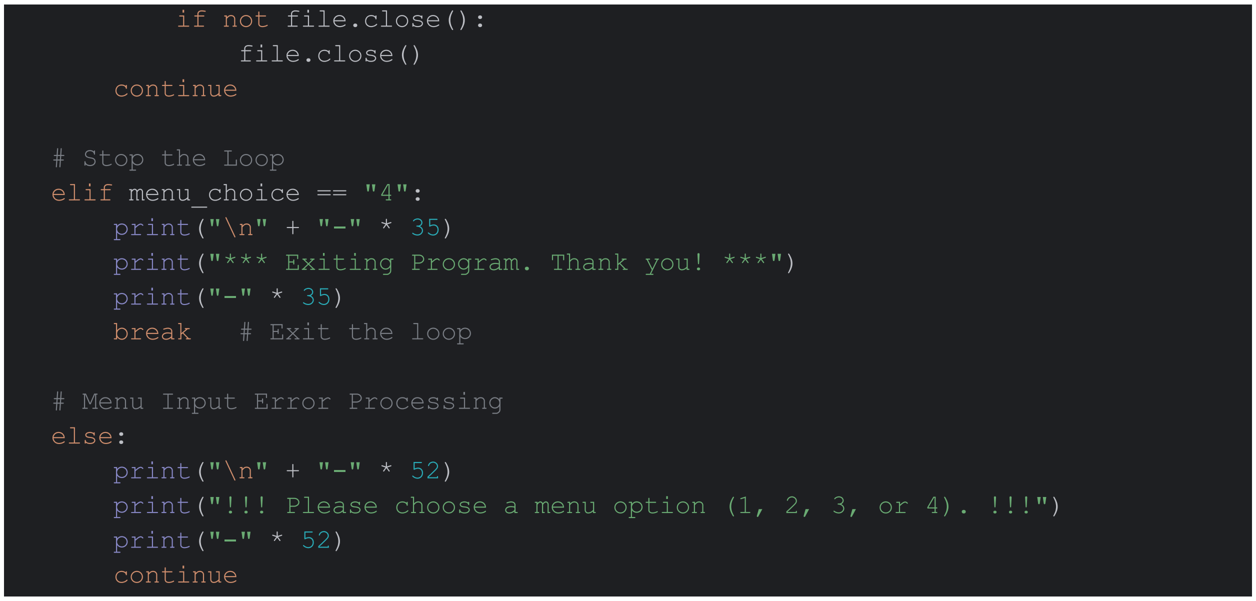
Creating the Script

Overview

This script is a course registration program that allows users to register students, view current data, and read & write to a JSON file. Figure 3 below shows the completed script for this assignment.

```
# --------------------------------------------------------------------------- #
# Title: Assignment05
# Desc: This assignment demonstrates using dictionaries, files, and exception
#       handling.
# Change Log: (Who, When, What)
#   N.Greco, 11/10/2024, Created Script
#   N.Greco, 11/12/2024, Added Exception Handling
# --------------------------------------------------------------------------- #

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
```

```python
  3. Save Data to a File
  4. Exit the Program
----------------------------------------
"""
FILE_NAME: str = "Enrollments.json"

# Define the Data Variables
course_name: str = ""               # Holds the input course name
file = None                         # Holds a reference to an opened file
json_data: str = ""                 # Holds the JSON file name
menu_choice: str = ""               # Holds the choice made by the user
student_data: dict = {}             # Holds the student data
student_first_name: str = ""        # Holds the input first name
student_last_name: str = ""         # Holds the input last name
students: list = []                 # Holds all combined student data

# Read Data from the JSON File
try:
    file = open(FILE_NAME, "r")
    students = json.load(file)
    file.close()
except FileNotFoundError as e:  # Handles file not found error
    print("\nJSON file must exist before running this script!")
    print("\n" + "-" * 20 + " Technical Error Message " + "-" * 20)
    print(e, e.__doc__, type(e), sep = '\n')
    print("-" * 65 + "\n")
except Exception as e:  # Handles all other errors
    print("\nThere was a non-specific error!")
    print("\n" + "-" * 20 + " Technical Error Message " + "-" * 20)
    print(e, e.__doc__, type(e), sep = '\n')
    print("-" * 65 + "\n")
finally:    # Closes the file if not already done
    if not file.close():
        file.close()

# Present and Process the Data
while (True):

    # Present the Menu
    print(MENU)
    menu_choice = input("Menu Selection: ")

    # Input User Data
    if menu_choice == "1":
        try:
            student_first_name = input("\nStudent's First Name: ")
            if not student_first_name.isalpha(): # Checks for letters only
                raise ValueError("The first name must only contain letters.")
            student_last_name = input("Student's Last Name: ")
            if not student_last_name.isalpha(): # Checks for letters only
                raise ValueError("The last name must only contain letters.")
            course_name = input("Course Name: ")
            student_data = ({"FirstName": student_first_name, "LastName":\
                student_last_name, "CourseName": course_name})
            students.append(student_data)
            print(f"\n{student_first_name} {student_last_name} is now "\
                  f"registered for {course_name}.\n*** Please Save data to "\
                  f"file to confirm. ***")
        except ValueError as e: # Handles incorrect character error
            print("\n" + "-" * 17 + " Error Message " + "-" * 17)
            print("\t" + e.__str__())
            print("-" * 49)
        except Exception as e: # Handles all other errors
            print("\nThere was a non-specific error!")
            print("\n" + "-" * 20 + " Technical Error Message " + "-" * 20)
            print(e, e.__doc__, type(e), sep='\n')
            print("-" * 65 + "\n")
        continue

    # Present the Current Data
    elif menu_choice == "2":
        print("\nCurrent List of Students:")
        for data in students:  # Displays all registered students
            print(f"\t{data["FirstName"]}, {data["LastName"]}, "\
                f"{data["CourseName"]}")
        continue

    # Save the data to a file
    elif menu_choice == "3":
        try:
            file = open(FILE_NAME, "w")
            json.dump(students, file)
            file.close()
            print("\nThe following data was saved to file:")
            for student in students:    # Displays all enrolled students
                print(f"\t{student["FirstName"]} {student["LastName"]} is " \
                    f"enrolled in {student["CourseName"]}.")
        except TypeError as e:  # Handles incorrect formatting of JSON file
            print("\nData must be in valid JSON format!")
            print("\n" + "-" * 20 + " Technical Error Message " + "-" * 20)
            print(e, e.__doc__, type(e), sep='\n')
            print("-" * 65 + "\n")
        except Exception as e:  # Handles all other errors
            print("\nThere was a non-specific error!")
            print("\n" + "-" * 20 + " Technical Error Message " + "-" * 20)
            print(e, e.__doc__, type(e), sep='\n')
            print("-" * 65 + "\n")
        finally:    # Closes the file if not already done
```



```
            if not file.close():
                file.close()
        continue

    # Stop the Loop
    elif menu_choice == "4":
        print("\n" + "-" * 35)
        print("*** Exiting Program. Thank you! ***")
        print("-" * 35)
        break   # Exit the loop

    # Menu Input Error Processing
    else:
        print("\n" + "-" * 52)
        print("!!! Please choose a menu option (1, 2, 3, or 4). !!!")
        print("-" * 52)
        continue
```

Figure 1: 'Assignment05' Python Script

Key Functionalities

This Python script is designed to manage a simple course registration system using dictionaries, file handling, and exception handling. Here's an overview of its functionality:

1. **Initialization and Constants:** The script begins by defining constants for the menu and the filename ("Enrollments.json") where student registration data will be stored. It also initializes several variables to hold student information.
2. **Reading Existing Data:** It attempts to read student enrollment data from the specified JSON file at the start of the program. If the file does not exist, it catches a 'FileNotFoundError' and prints an appropriate error message. It also handles any other exceptions that might occur during this file reading process.
3. **User Interaction via Menu:** The script enters an infinite loop where it displays a menu with four options:
    - **Register a Student for a Course:** This option prompts the user for a student's first and last name and the course name. It validates that the names contain only alphabetical characters. If valid, the student data is stored in a dictionary and added to a list of students. If the input is invalid, it raises a 'ValueError' and displays an error message.
    - **Show Current Data:** This option displays the current list of registered students, showing their first names, last names, and course names.
    - **Save Data to a File:** This option saves the current list of students to the specified JSON file. It uses json.dump() to write the list to the file, handling any potential TypeError or other exceptions during this process.
    - **Exit the Program:** This option allows the user to exit the loop and end the program with a farewell message.

4. **Error Handling:** Throughout the script, various try-except blocks are employed to manage exceptions that may arise from file operations or user input, ensuring the program can gracefully handle errors without crashing.
5. **Loop Control:** The script continues to present the menu until the user selects the option to exit. If the user provides an invalid menu choice, it prompts them to select a valid option.

Concepts Demonstrated

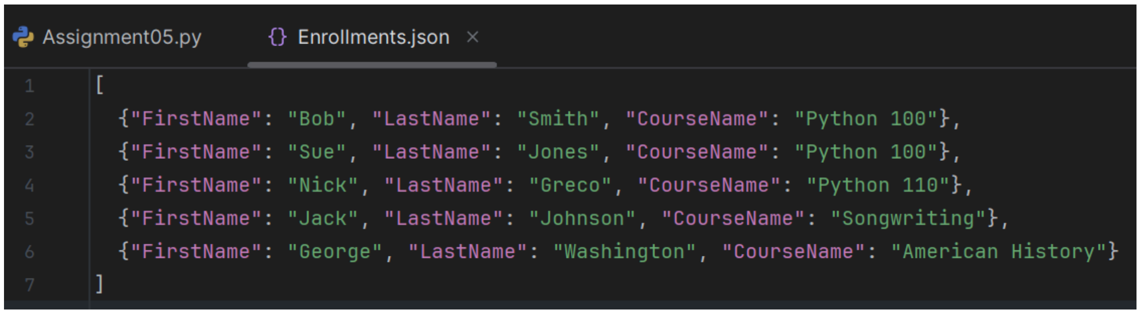
The script demonstrates several concepts from previous assignments, along with the following additions:

1. **Dictionaries:** The script uses dictionaries to store student data, where each student's information (first name, last name, course name) is stored as a key-value pair. This allows for easy access and modification of student records.
2. **Lists:** A list is used to hold multiple student dictionaries, allowing for the management of multiple student registrations within a single data structure.
3. **File Handling:** The script demonstrates how to read data from a JSON file using json.load(), allowing the program to load existing student data at startup.
4. **Writing to Files:** It also shows how to save data back to a file using json.dump(), ensuring that the user's input persists between program executions.
5. **Try-Except Blocks:** The script employs try-except blocks to handle potential errors gracefully, such as file not found errors or invalid user inputs. This is crucial for preventing the program from crashing and providing user-friendly error messages.
6. **User Input Validation:** The script includes validation checks to ensure that user inputs for names contain only alphabetic characters. This helps maintain data integrity and provides feedback to the user when inputs are invalid.

Running the Script

Using PyCharm IDE

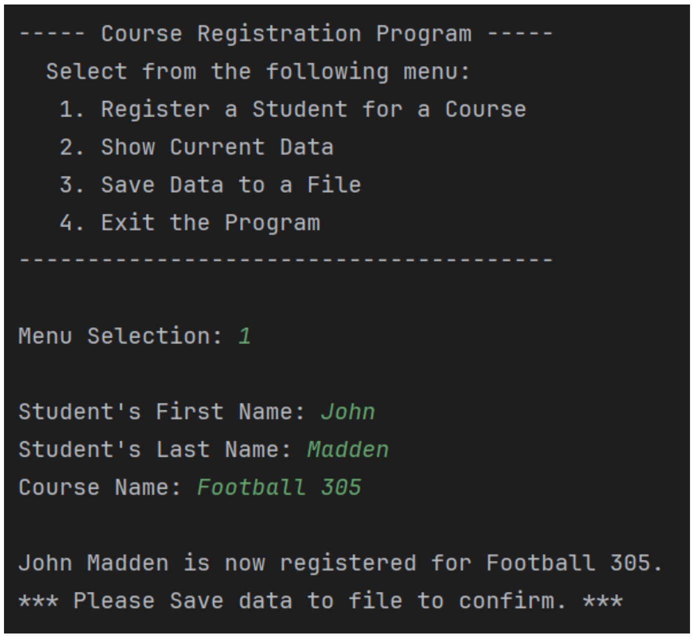
The script was run using PyCharm IDE. Figure 2 shows the contents of 'Enrollments.json' prior to running the 'Assignment05' script.



```
[
  {"FirstName": "Bob", "LastName": "Smith", "CourseName": "Python 100"},
  {"FirstName": "Sue", "LastName": "Jones", "CourseName": "Python 100"},
  {"FirstName": "Nick", "LastName": "Greco", "CourseName": "Python 110"},
  {"FirstName": "Jack", "LastName": "Johnson", "CourseName": "Songwriting"},
  {"FirstName": "George", "LastName": "Washington", "CourseName": "American History"}
]
```

Figure 2: 'Enrollments.json' Data Contents before running Script

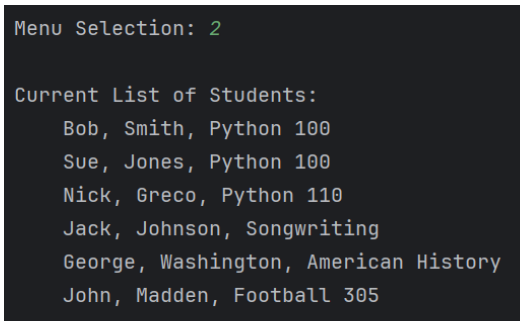
Figure 3 shows the menu presented to the user and displayed prompts when selection "1" is chosen. Example user input is shown in green text.



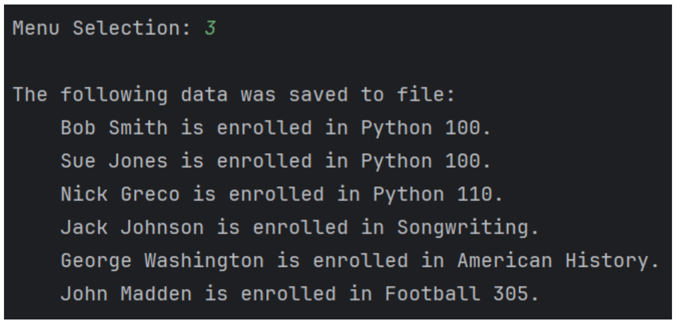
```
----- Course Registration Program -----
  Select from the following menu:
   1. Register a Student for a Course
   2. Show Current Data
   3. Save Data to a File
   4. Exit the Program
---------------------------------------

Menu Selection: 1

Student's First Name: John
Student's Last Name: Madden
Course Name: Football 305

John Madden is now registered for Football 305.
*** Please Save data to file to confirm. ***
```

Figure 3: Menu and Option "1" Prompts

Figure 4 shows the program outputs when menu options 2 (left) and 3 (right) are chosen.



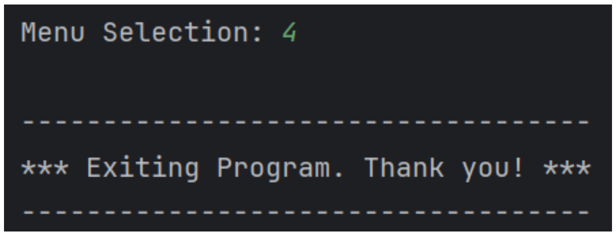
```
Menu Selection: 2

Current List of Students:
    Bob, Smith, Python 100
    Sue, Jones, Python 100
    Nick, Greco, Python 110
    Jack, Johnson, Songwriting
    George, Washington, American History
    John, Madden, Football 305
```



```
Menu Selection: 3

The following data was saved to file:
    Bob Smith is enrolled in Python 100.
    Sue Jones is enrolled in Python 100.
    Nick Greco is enrolled in Python 110.
    Jack Johnson is enrolled in Songwriting.
    George Washington is enrolled in American History.
    John Madden is enrolled in Football 305.
```

Figure 4: Menu Options "2" (left) and "3" (right) Outputs

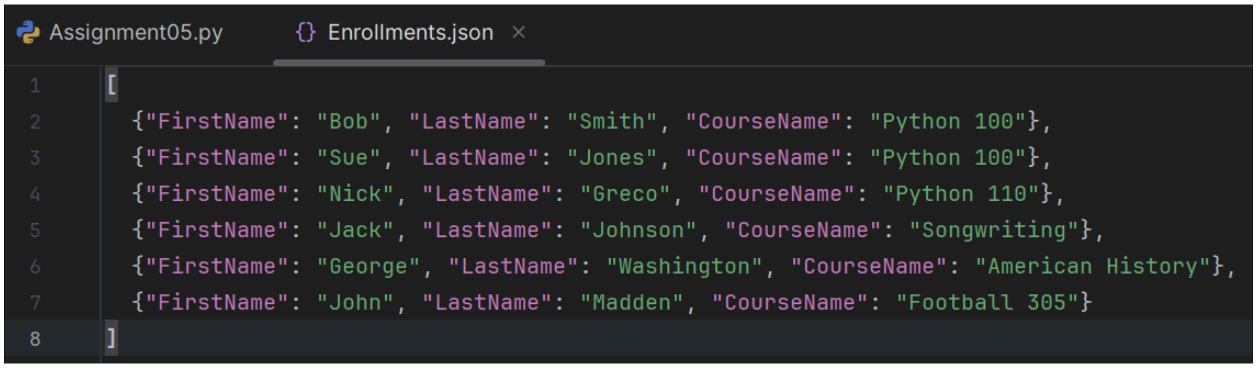
Figure 5 shows the program output when menu option 4 is chosen.



```
Menu Selection: 4


-----------------------------------
*** Exiting Program. Thank you! ***
-----------------------------------
```

Figure 5: Menu Option "4" Output

Figure 6 shows the contents of the 'Enrollments.json' file after running the 'Assignment05' script and providing user input to register a single additional student.

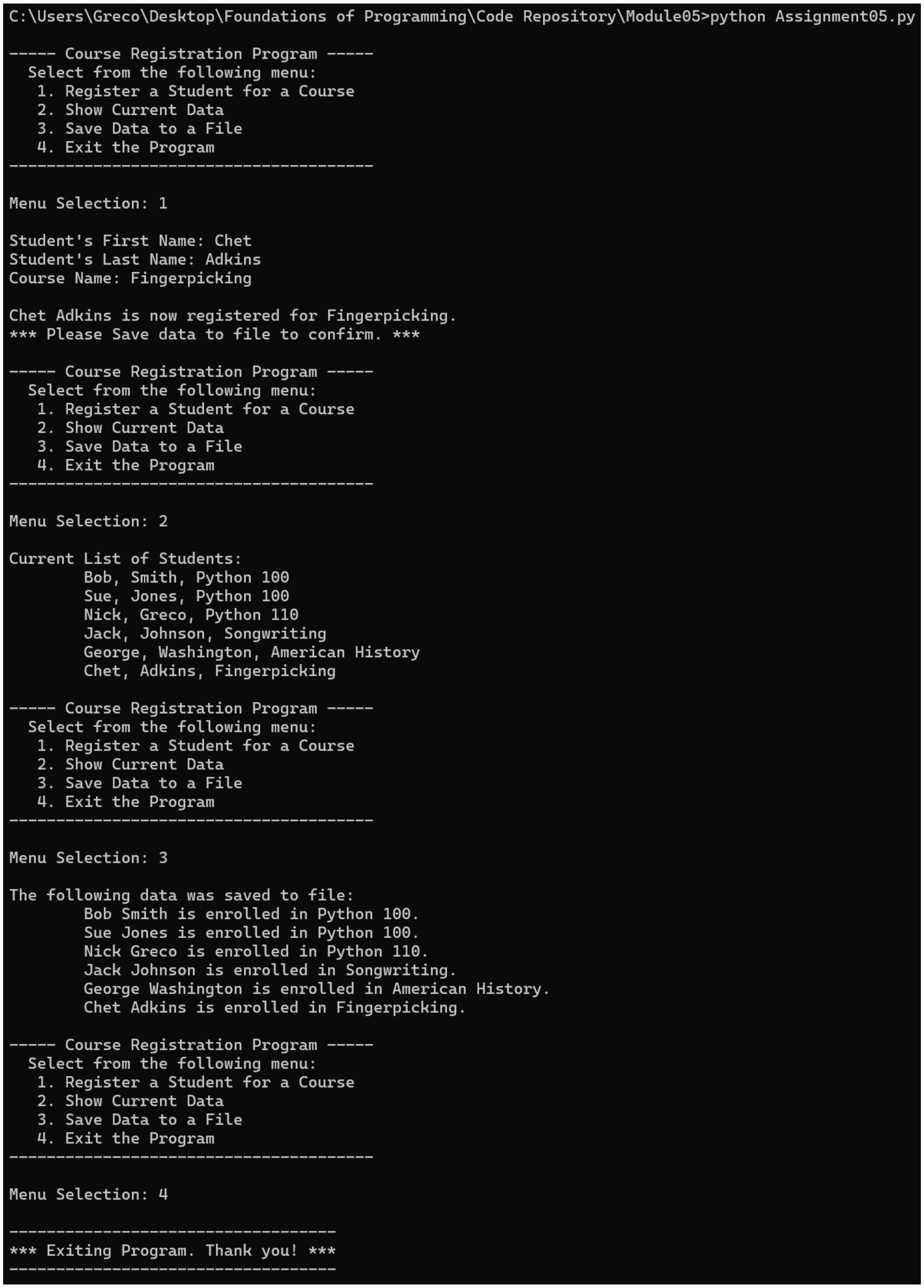


```
[
  {"FirstName": "Bob", "LastName": "Smith", "CourseName": "Python 100"},
  {"FirstName": "Sue", "LastName": "Jones", "CourseName": "Python 100"},
  {"FirstName": "Nick", "LastName": "Greco", "CourseName": "Python 110"},
  {"FirstName": "Jack", "LastName": "Johnson", "CourseName": "Songwriting"},
  {"FirstName": "George", "LastName": "Washington", "CourseName": "American History"},
  {"FirstName": "John", "LastName": "Madden", "CourseName": "Football 305"}
]
```

Figure 6: 'Enrollments.json' Data Contents after running Script

Using Windows Command Prompt

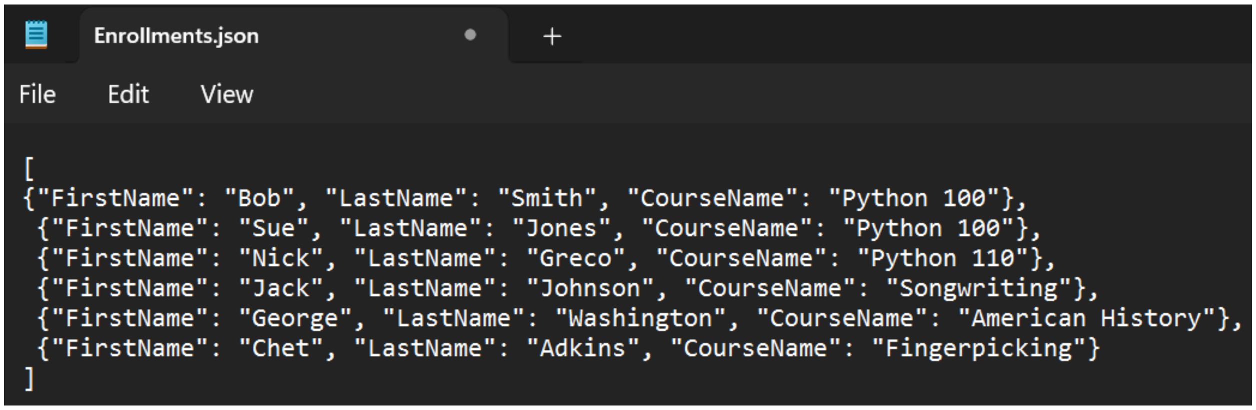
Figure 7 shows running the same script using Windows Command Prompt.



```
C:\Users\Greco\Desktop\Foundations of Programming\Code Repository\Module05>python Assignment05.py

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

Menu Selection: 1

Student's First Name: Chet
Student's Last Name: Adkins
Course Name: Fingerpicking

Chet Adkins is now registered for Fingerpicking.
*** Please Save data to file to confirm. ***

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

Menu Selection: 2

Current List of Students:
        Bob, Smith, Python 100
        Sue, Jones, Python 100
        Nick, Greco, Python 110
        Jack, Johnson, Songwriting
        George, Washington, American History
        Chet, Adkins, Fingerpicking

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

Menu Selection: 3

The following data was saved to file:
        Bob Smith is enrolled in Python 100.
        Sue Jones is enrolled in Python 100.
        Nick Greco is enrolled in Python 110.
        Jack Johnson is enrolled in Songwriting.
        George Washington is enrolled in American History.
        Chet Adkins is enrolled in Fingerpicking.

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

Menu Selection: 4

-----------------------------------
*** Exiting Program. Thank you! ***
-----------------------------------
```

Figure 7: Using Windows Command Prompt to run 'Assignment05.py'

Figure 8 shows the contents of the JSON file after executing the script using Command Prompt.



```
[
{"FirstName": "Bob", "LastName": "Smith", "CourseName": "Python 100"},
 {"FirstName": "Sue", "LastName": "Jones", "CourseName": "Python 100"},
 {"FirstName": "Nick", "LastName": "Greco", "CourseName": "Python 110"},
 {"FirstName": "Jack", "LastName": "Johnson", "CourseName": "Songwriting"},
 {"FirstName": "George", "LastName": "Washington", "CourseName": "American History"},
 {"FirstName": "Chet", "LastName": "Adkins", "CourseName": "Fingerpicking"}
]
```

Figure 8: JSON File Contents after running Script

Summary

This document outlines the process of creating a basic student enrollment system using Python and the PyCharm IDE, detailing the script's functionality, which includes student registration, data viewing, and file operations. The script utilizes key programming concepts such as dictionaries, lists, file handling, and exception handling, ensuring robust user input validation and error management. Additionally, the document provides examples of the user interface and outputs when executing the script, demonstrating how the system operates both in PyCharm and via the Windows Command Prompt.

Citations

OpenAI ChatGPT. (November 2024). https://chatgpt.com/: Aspects of this assignment were informed and created by queries I submitted to ChatGPT.