Renato Felicio
November 11, 2024
IT FDN 110 A
Assignment 05

Creating Python Scripts – Advanced Collections and Error Handling

Introduction

This Assignment 05 consists in creating a Python program that uses constants, variables, and print statements to display a message about a student's registration for a Python course. Building on the knowledge from Assignment 04, this assignment introduces new concepts, such as use of data processing using dictionaries, exception handling and a new data file format.

Preparation for this assignment

To prepare for this assignment, I reviewed the "Module 05 Notes" (Reference 1), completed the three lab examples, and watched both the Module 05 videos available on Washington University's Canvas platform (Figure 1) and the recommended external videos in References 2 through 5. Through these materials, I learned how to work with collections like dictionaries, and lists (tables) containing dictionaries. I also learnt how to work with a new data file format called JSON file (JavaScript Object Notation), a flexible and hierarchical format consisting of key-value pairs that is very suitable for using it in combination with dictionaries in Python.

I also learnt structure error handling, the Try-Except construct, which prevents user errors from causing the program to stop running by printing user friendly error messages, and redirecting the user to provide valid inputs.

I learnt about a cloud-based platform for code hosting and collaboration called GitHub that is used by developers for file and code repository hosting, that offers robust version control, collaboration features, access control, allows developers to clone repositories and provides web-based access to code repositories.

Mod05 Videos

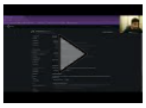
Module	Name	Link
5	Demo01–Using Dictionaries with files	https://www.youtube.com/watch?v=Kgh2TZ0tf28 
5	Demo02-Using JSON Files	https://www.youtube.com/watch?v=XDDyhzRFDuc 
5	Demo03-Try/Except	https://www.youtube.com/watch?v=O46YoSo477Y 
5	Demo04-Using The Github Website	https://www.youtube.com/watch?v=xmIAW_wJIB4 

Figure 1 – Mod05-Videos

Python Scripting

I began by using the provided Assignment05-Starter.py file as the starting point for my script. The objective was to implement the use of dictionaries and to read and save data in a JSON file format. The script should allow the user to register multiple students, display their inputs, and save the data back into a '.json' file containing all registered students. Additionally, structured error handling should be implemented when reading the file into dictionary rows, when the user enters first and last names, and when writing the dictionary rows back into the file.

The script is too long to be displayed in a single figure, so for improved readability, it has been split across Figure 2 to Figure 4.

```python
# ------------------------------------------------------------------------------------------ #
# Title: Assignment05
# Desc: This assignment demonstrates using dictionaries, files, and exception handling
# Change Log: (Who, When, What)
#   Renato Felicio,11/10/2024,Created Script
#   <Your Name Here>,<Date>, <Activity>
# ------------------------------------------------------------------------------------------ #

# 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.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.
json_data: str = ''  # This variables will hold data read from file
student_data: dict = {}  # one row of student data
students: list = []  # a table of student data
file = None  # Holds a reference to an opened file.
menu_choice: str = '' # Hold the choice made by the user.

# When the program starts, read the file data into a list of lists (table)
# Extract the data from the file
import json # Imports code from Python's json module into script
try:
    file = open(FILE_NAME, "r") # Open the JSON file for reading
    students = json.load(file) # File data is loaded into a table
    # Now 'students' contains the parsed JSON data as a Python list of dictionaries
    file.close()
except FileNotFoundError as e:  # Handles error in case there is no initial file
    print("Data file must exist before running this script!\n")
    print(e, e.__doc__, type(e), sep='\n') # prints error message
    file = open(FILE_NAME, "w") # Creates an empty initial file, in case of file not found
    print("Empty file was created")
except Exception as e:
    print("There was a non-specific error!\n")
    print("-- Technical Error Message -- ")
    print(e, e.__doc__, type(e), sep='\n')
finally: # It closes the file
        file.close()
```

Figure 2 – Python Script

```python

# Present and Process the data
while True:

    # Present the menu of choices
    print(MENU)
    menu_choice = input("What would you like to do: ")

    # Input user data
    if menu_choice == "1":  # This will not work if it is an integer!

        try: # It handles user entering a non-alphabetic character for first name
            student_first_name = input("Enter the student's first name: ") # Asks user for student's first name
            if not student_first_name.isalpha(): # checks if student's first name is all alphabetic characters
                raise ValueError("Student's first name should contain only alphabetic characters.") # Custom error

        except ValueError as e: # Prints error message and restarts loop
            print(e, e.__doc__, type(e), sep='\n')  # Prints the custom message
            continue

        try:# It handles user entering a non-alphabetic character for last name
            student_last_name = input("Enter the student's last name: ")  # Asks user for student's last name
            if not student_last_name.isalpha(): # checks if student's last name is all alphabetic characters
                raise ValueError("Student's last name should contain only alphabetic characters.") # Custom error
        except ValueError as e:# Prints error message and restarts loop
            print(e, e.__doc__, type(e), sep='\n')  # Prints the custom message
            continue

        course_name = input("Please enter the name of the course: ")# Asks user for student's course name

        #User input data is loaded into a dictionary below:
        student_data = {"FirstName": student_first_name, "LastName": student_last_name, "CourseName":course_name}
        students.append(student_data) # Dictionary with user input data is appended to a table with all students\
                                      # information
    # Present the current data
    elif menu_choice == "2":

        # Process the data to create and display a custom message
        print("-"*50)
        # Loops through dictionaries in the list and prints as a formated string
        for student in students:
            print(f"Student {student["FirstName"]} {student["LastName"]} is enrolled in {student["CourseName"]}")
        print("-"*50)
        continue
```

Figure 3 – Python Script Continuation 1

```python

    # Save the data to a file
    elif menu_choice == "3":
        try:
            file = open(FILE_NAME, "w")
            json.dump(students, file) # It writes the list of dictionaries into a json file
        except Exception as e: # It handles any exception that could happen when writing the file
            print("There was an error writing into the data file")
            print(e, e.__doc__, type(e), sep='\n')
        finally:
            print("Data File Closed")
            file.close()
        print("The following data was saved to file!\n")
        for student in students:
            json_data +=f"Student {student["FirstName"]} {student["LastName"]} is enrolled in {student["CourseName"]}\n"
        print(json_data)  # print formated string with all the data saved into json file
        continue

    # Stop the loop
    elif menu_choice == "4":
        break  # out of the loop
    else:
        print("Please only choose option 1, 2, or 3")

print("Program Ended")

```
Figure 4 – Python Script Continuation 2

I started by defining the data variables and their initial values. I changed the student_data variable from a list to a dictionary, renamed the csv_data string to json_data, and included a statement to import the json module into my script.

After setting up variables, I modified the data presentation and processing parts of the script. In menu choice 1, I changed the student_data list into a dictionary, which stores user input collected through input functions in the student_first_name, student_last_name, and course_name string variables and appends student_data to a table (students). In menu option 2, I updated the loop to iterate through the students table, which contains dictionary rows. This loop collects information from each dictionary and presents it to the user in a formatted string via the print() function.

Next, I updated the part of the script that reads student data from Enrollments.json at the start of the script and writes data back to the same file in menu option 3, using json module functions json.load() and json.dumps().

Finally, I added try-except structured error handling. This includes error handling for invalid first and last names that will prompt the user to re-enter valid input (see Figure 5), error handling when reading the data file at the start of the script (notifying the user if the file doesn't exist and creating a new one), and error handling in menu choice 3 to notify of any issues saving the data (see Figure 6 and Figure 7).

```python
    # Input user data
    if menu_choice == "1":  # This will not work if it is an integer!

        try: # It handles user entering a non-alphabetic character for first name
            student_first_name = input("Enter the student's first name: ") # Asks user for student's first name
            if not student_first_name.isalpha(): # checks if student's first name is all alphabetic characters
                raise ValueError("Student's first name should contain only alphabetic characters.") # Custom error

        except ValueError as e: # Prints error message and restarts loop
            print(e, e.__doc__, type(e), sep='\n')  # Prints the custom message
            continue

        try:# It handles user entering a non-alphabetic character for last name
            student_last_name = input("Enter the student's last name: ")  # Asks user for student's last name
            if not student_last_name.isalpha(): # checks if student's last name is all alphabetic characters
                raise ValueError("Student's last name should contain only alphabetic characters.") # Custom error
        except ValueError as e:# Prints error message and restarts loop
            print(e, e.__doc__, type(e), sep='\n')  # Prints the custom message
            continue
```

Figure 5 – First and Last Name Error Handling

```python
import json # Imports code from Python's json module into script
try:
    file = open(FILE_NAME, "r") # Open the JSON file for reading
    students = json.load(file) # File data is loaded into a table
    # Now 'students' contains the parsed JSON data as a Python list of dictionaries
    file.close()
except FileNotFoundError as e:  # Handles error in case there is no initial file
    print("Data file must exist before running this script!\n")
    print(e, e.__doc__, type(e), sep='\n') # prints error message
    file = open(FILE_NAME, "w") # Creates an empty initial file, in case of file not found
    print("Empty file was created")
except Exception as e:
    print("There was a non-specific error!\n")
    print("-- Technical Error Message -- ")
    print(e, e.__doc__, type(e), sep='\n')
finally: # It closes the file
        file.close()
```

Figure 6 – Enrollments Reading Error Handling

```python
    elif menu_choice == "3":
        try:
            file = open(FILE_NAME, "w")
            json.dump(students, file) # It writes the list of dictionaries into a json file
        except Exception as e: # It handles any exception that could happen when writing the file
            print("There was an error writing into the data file")
            print(e, e.__doc__, type(e), sep='\n')
        finally:
            print("Data File Closed")
            file.close()
        print("The following data was saved to file!\n")
        for student in students:
            json_data +=f"Student {student["FirstName"]} {student["LastName"]} is enrolled in {student["CourseName"]}\n"
        print(json_data)  # print formated string with all the data saved into json file
        continue
```

Figure 7 – Enrollments Writing Error Handling

Python Script Testing

I executed the Python script using PyCharm (see Figure 9) and also tested it in the Windows Command Prompt (see Figure 10), verifying that the script performed as expected in both environments. Additionally, I confirmed that the Enrollments.json file was updated correctly and contained the expected output (see Figure 11). Following the assignment instructions, I used the initial Enrollments.json file provided in the module_05.zip file, after correcting one key for the second student from 'Email' to 'LastName.' The original content of this file is shown in Figure 8.

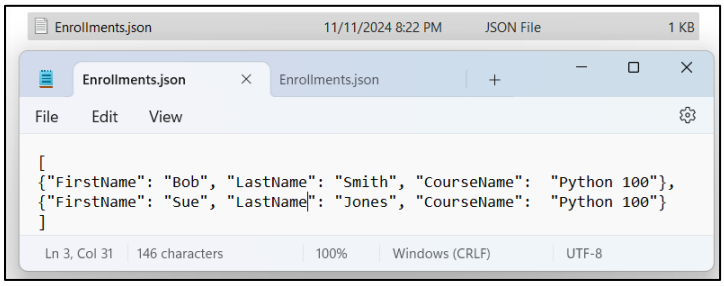


Figure 8 – Initial Enrollment File Python

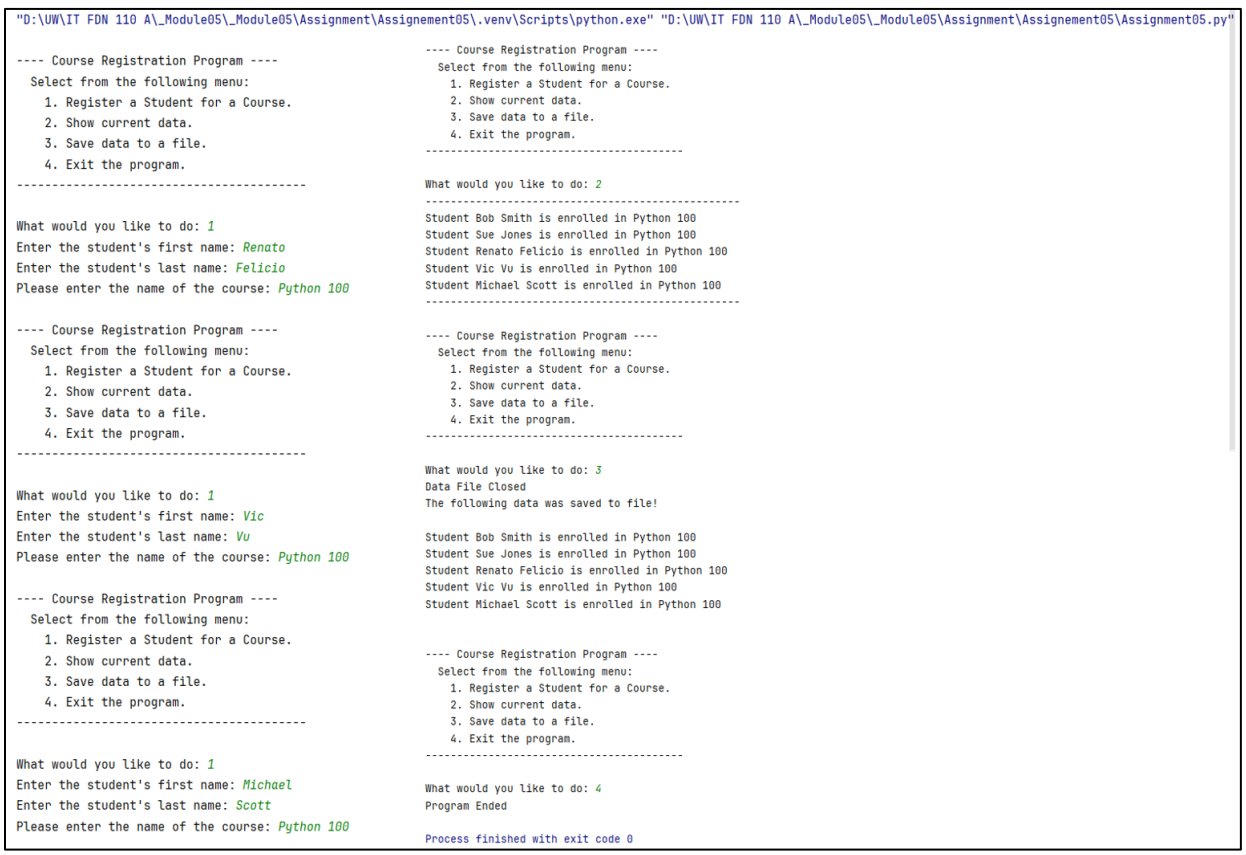


Figure 9 – Python Script PyCharm Run

```
Command Prompt

D:\UW\IT FDN 110 A\_Module05\_Module05\Assignment\Assignement05>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.
----------------------------------------

What would you like to do: 1
Enter the student's first name: Renato
Enter the student's last name: Felicio
Please enter the name of the course: 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.
----------------------------------------

What would you like to do: 1
Enter the student's first name: Vic
Enter the student's last name: Vu
Please enter the name of the course: 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.
----------------------------------------

What would you like to do: 1
Enter the student's first name: Michael
Enter the student's last name: Scott
Please enter the name of the course: 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.
----------------------------------------

What would you like to do: 2
-------------------------------------------------
Student Bob Smith is enrolled in Python 100
Student Sue Jones is enrolled in Python 100
Student Renato Felicio is enrolled in Python 100
Student Vic Vu is enrolled in Python 100
Student Michael Scott 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.
----------------------------------------

What would you like to do: 3
Data File Closed
The following data was saved to file!

Student Bob Smith is enrolled in Python 100
Student Sue Jones is enrolled in Python 100
Student Renato Felicio is enrolled in Python 100
Student Vic Vu is enrolled in Python 100
Student Michael Scott 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.
----------------------------------------

What would you like to do: 4
```

Figure 10 – Python Script Windows Command Prompt Run

Figure 11 shows that the Enrollments.json file was updated correctly with the three new student data.

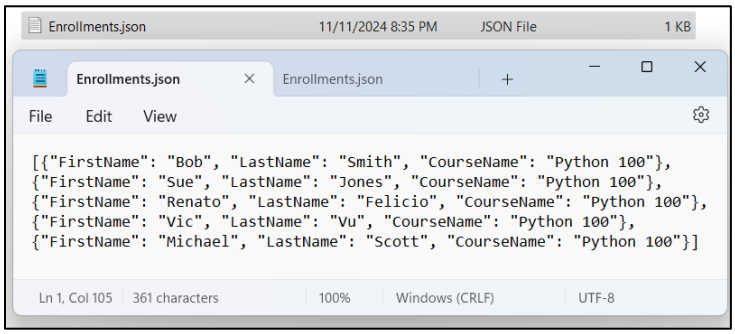


Figure 11 – Output JSON File Content

Python Script Error Handling Test

After testing the script for the valid user inputs with a pre-existing Enrollments.json file, I tested its error handling for scenarios such as a 'file not found' exception and invalid first and last name inputs. These tests were conducted using PyCharm.

The Figure 12 below shows the FileNotFound handling and the Enrolments.json being updated correctly after menu choice 3 is selected.

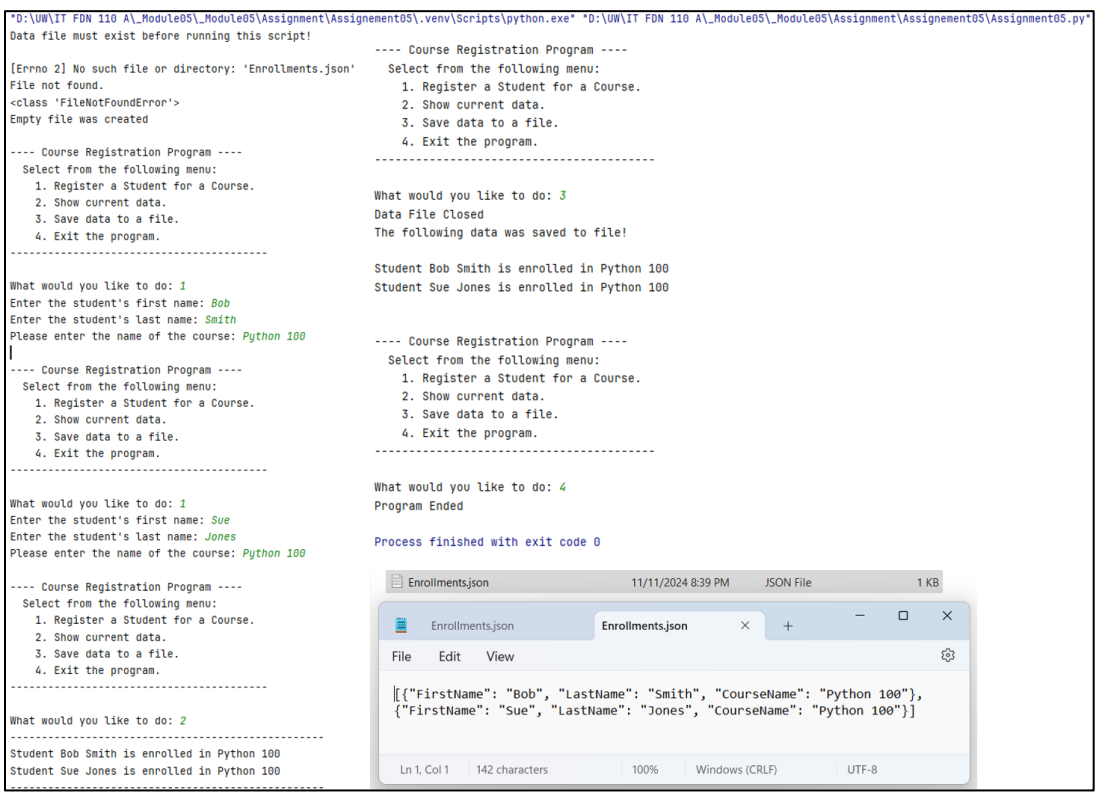


Figure 12 – FileNotFoundError Handling

The second error handling test involved checking the condition where the user enters non-alphabetic characters for the first or last names (see Figure 13). The Enrollments.json file was also correctly updated

```
"D:\UW\IT FDN 110 A\_Module05\_Module05\Assignment\Assignement05\.venv\Scripts\python.exe" "D:\UW\IT FDN 110 A\_Module05\_Module05\Assignment\Assignement05\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.
----------------------------------------

What would you like to do: 1
Enter the student's first name: 1
Student's first name should contain only alphabetic characters.
Inappropriate argument value (of correct type).
<class 'ValueError'>

---- 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.
----------------------------------------

What would you like to do: 1
Enter the student's first name: Renato
Enter the student's last name: Felicio
Please enter the name of the course: 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.
----------------------------------------

What would you like to do: 1
Enter the student's first name: Vic
Enter the student's last name: 2
Student's last name should contain only alphabetic characters.
Inappropriate argument value (of correct type).
<class 'ValueError'>

---- 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.
----------------------------------------

What would you like to do: 1
Enter the student's first name: Vic
Enter the student's last name: Vu
Please enter the name of the course: 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.
----------------------------------------

What would you like to do: 2
-------------------------------------------------
Student Bob Smith is enrolled in Python 100
Student Sue Jones is enrolled in Python 100
Student Renato Felicio is enrolled in Python 100
Student Vic Vu 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.
----------------------------------------

What would you like to do: 3
Data File Closed
The following data was saved to file!

Student Bob Smith is enrolled in Python 100
Student Sue Jones is enrolled in Python 100
Student Renato Felicio is enrolled in Python 100
Student Vic Vu 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.
----------------------------------------

What would you like to do: 4
Program Ended

Process finished with exit code 0
```

```
Enrollments.json    11/11/2024 8:43 PM    JSON File    1 KB

Enrollments.json    Enrollments.json
File    Edit    View

[{"FirstName": "Bob", "LastName": "Smith", "CourseName": "Python 100"},
{"FirstName": "Sue", "LastName": "Jones", "CourseName": "Python 100"},
{"FirstName": "Renato", "LastName": "Felicio", "CourseName": "Python 100"},
{"FirstName": "Vic", "LastName": "Vu", "CourseName": "Python 100"}]

Ln 1, Col 1    286 characters    100%    Windows (CRLF)    UTF-8
```

Figure 13 – Non-Alphabetic Characters Error Handling

GitHub

Script and documentation for this assignment is available in my GitHub site: https://github.com/rfnaval/IntroToProg-Python-Mod05.git.

Summary

The fifth assignment built on the previous one helped me learn and practice Python data collection concepts such as dictionaries, tables (list of lists) containing rows of dictionaries, a new data file format called JSON file. JSON file consists of key-value pairs, ideal to use in combination with Python dictionaries. I also learned about structured error handling using the try-except construct, which captures and handles errors, preventing the introduction of new bugs during user interaction with the program. Additionally, it was very interesting to learn about GitHub and to create an account and a repository."

References

1. Module 05 - Advanced Collections and Error Handling, Randal Root, January 02, 2024.
2. External site: Python Tutorial for Beginners 5: Dictionaries - Working with Key-Value Pairs, Corey Schafer.
3. External site: What Is JSON | Explained, Hostinger Academy.
4. External site: Exceptions in Python - Python Tutorial , Socratica.
5. External site: GitHub Tutorial - Beginner's Training Guide, Anson Alexander.