Jennifer Grace L. Duldulao
05/20/2025
Foundations Of Programming: Python
Assignment 05
https://github.com/hihellojini/IntroToProg-Python-Mod05

Advanced Collections and Error Handling

Introduction

In the last module, we discussed data collections and the four data collection types in Python Programming. Then we created a program that will show the user the current data, asked for user inputs, saved to file user inputs, and exit the program using list and nested list to collect data. In this module, the assignment will be similar to module 4 assignment but instead of using list as the data collection, we will be using dictionary. In here, we will deep dive about dictionary, using JSON files and error handling. We will also be covering managing code files.

Deep Dive on Dictionary as a Data Collection

Dictionary is a powerful data collection due to its ability to store data in ***key-value*** pairs. This helps the data to be organized, flexible and readable. Although ***keys*** are unique and cannot be changed, the ***values*** are mutable, meaning they can be modified after creation. So, items can be added, removed or updated as needed, which is very helpful in data management. Dictionary also allows for efficient data retrieval with the use of the associated ***key***. In a real-world data management, ***dictionary*** is very useful as information is normally associated with a unique identifier.

Figure 1 shows a simple differentiation between ***list*** and ***dictionary***.

Example of a list

Jenny	Duldulao	Python 100
Ha	Le	Python 100
Clay	Gal	Python 100

Example of a dictionary

FirstName	*LastName*	*CourseName*
Jenny	Duldulao	Python 100
Ha	Le	Python 100
Clay	Gal	Python 100

Figure 1. List vs Dictionary

In the figure above, you will see the visual representation of the data when converted to a CSV file. Using the same information, ***list*** shows the data but it doesn't associate the data into a unique identifier whereas in ***dictionary***, the data are associated to the unique identifier. In this example, the identifier FirstName is the key and the values are Jenny, Ha and Clay. In ***dictionary***, the example in Figure 1 will be written as:

[{"FirstName": "Jenny", "LastName": "Duldulao", "CourseName": "Python 100"}, {"FirstName": "Ha", "LastName": "Le", "CourseName": "Python 100"}, {"FirstName": "Clay", "LastName": "Gal", "CourseName": "Python 100"}]

Index vs Key

Index and key are both used to access elements in the data structure. In module 4, we used index to represent the position of an item in an ordered collection. Going back to the example from the list in Figure 1 row 1, if we use ***list[0]***, the output would be ***Jenny***.

Now, ***key*** is the identifier used to access the value in the dictionary. From the dictionary example in Figure 1 row 1, if we use ***dict["FirstName"]***, the output would be ***Jenny***.

Table below shows the key differences between ***index*** and ***key***.

Feature	Index	Key
Data Structure	List, strings, tuples	Dictionaries
Type	Integer	String, number, tuple data type that immutable
Ordering	Ordered	Unordered
Purpose	Access element by position	Access element by unique identifier

Adding and removing data

The figures below show an example of adding, updating and removing elements from a dictionary.

Figure 2 shows the code and Figure 3 shows the output when the code is run.

```
# Data variables
dict_test = {'a': 12, 'b': 2}
print("Initial Data")
print(dict_test)
print()

# Adding elements
dict_test['c'] = 10
dict_test['d'] = 4
print("Adding elements key-value pair 'c': 10")
print(dict_test)
print()

# Updates an element
dict_test['a'] = 1
print("Updating an element 'a': 1")
print(dict_test)
print()

# Updates an element using update() method
print("Updating an element 'c': 3 using update() function")
dict_test.update({'c': 3})
print(dict_test)
print()

# Removes an element using del statement
print("Removing element 'd': 4 using del statement")
del dict_test['d']
print(dict_test)
print()
```

Figure 2. Code Example of Adding, Updating, and Removing Element in Dictionary

```
Initial Data
{'a': 12, 'b': 2}

Adding elements key-value pair 'c': 10
{'a': 12, 'b': 2, 'c': 10, 'd': 4}

Updating an element 'a': 1
{'a': 1, 'b': 2, 'c': 10, 'd': 4}

Updating an element 'c': 3 using update() function
{'a': 1, 'b': 2, 'c': 3, 'd': 4}

Removing element 'd': 4 using del statement
{'a': 1, 'b': 2, 'c': 3}
```

Figure 3. Output Example When Code in Figure 2 is Run

Note that just like ***list***, ***dictionary*** can also be used with data files. Assignment 5 will show more how data files a written, appended and saved.

Understanding JSON Files

"JSON (JavaScript Object Notation, pronounced /ˈdʒeɪsən/ or /ˈdʒeɪˌsɒn/) is an open standard file format and data interchange format that uses human-readable text to store and transmit data objects consisting of name–value pairs and arrays (or other serializable values). It is a commonly used data format with diverse uses in electronic data interchange, including that of web applications with servers.

JSON is a language-independent data format. It was derived from JavaScript, but many modern programming languages include code to generate and parse JSON-format data. JSON filenames use the extension .json." (https://en.wikipedia.org/wiki/JSON#:~:text=It%20is%20a%20commonly%20used,.json, 2025) (External file)

Because of its simplicity, flexibility and compatibility with popular programming languages, JSON is widely used by programmers for building web and mobile applications. It is text-based and lightweight and requires no additional code to understand and interpret the data provided. JSON also support several data types including objects, arrays, strings, Boolean, null and numbers.

JSON vs CSV

While JSON and CSV are both data formats that are simple to read, write and interpret, they differ significantly in their structure and uses.

CSV is structured in tabular format and values are separated by comma or other delimiter and each new record is separated by a newline. It is ideal for spreadsheet and exporting data from databases. However, it is limited in data complexity and cannot represent hierarchical structures.

JSON uses the key-value pairing format that allows nested objects and arrays, hence can represent more complex data structures than CVS. This makes it suitable for much more complex applications and databases that programmers would use.

In assignment 5, we are using JSON as data file to hold the values of the dictionary. A Python dictionary can be written to a JSON file manually by converting it to a string or using the JSON module that has been built-in in Python. As the data gets more complicated, it is best to use the JSON module. Assignment 5 also uses the JSON module for coding.

Structured Error Handling

“Python Exception Handling handles errors that occur during the execution of a program. Exception handling allows to respond to the error, instead of crashing the running program. It enables you to catch and manage errors, making your code more robust and user-friendly.” (https://www.geeksforgeeks.org/python-exception-handling/, 2025) (External site)

Try-Except Blocks

The ***try-except*** block is the fundamental approach for error handling in Python. The ***try*** block test a block of code for errors and the ***except*** block handles the error without stopping or crashing the program.

Some of the common ***Exception*** errors are:

- ***IOError:*** *if the file can't be opened*
- ***ValueError:*** *when the built-in function receives a wrong argument*
- ***EOFError:*** *if End-Of-File is hit without reading any data*
- ***ImportError:*** *if it is unable to find the module*

In assignment 5, error handling will be used in the code.

Assignment Discussion

Assignment 5 program calls for the user to choose to register the student information, show the current data on a JSON file, save the new information provided to the data file or exit the program. The assignment is similar to assignment 4 except that in here, the student information has to be written in dictionary and that error handling exceptions are present.

Figure 4 shows the first part of the code for assignment 5. In here you will see the header that describe what the program is about and the program log. You will also see the data constants and variables defined. One important thing that you can see in here is that after the header, the JSON module and import io as _io are imported to make the modules available. JSON module provides tools for working with JSON file and import io as _io provides the important tools for working with various types of I/O (input/output) streams. Line 37 to 51 reads the file data into a list (students) and adds exception if the JSON file does not exist.

Figure 5 shows the second part of the program. The while loop repeats the task unless the user chooses to exit the program. Note that, in the menu_choice 1, error handling is also use to tell the user that first and last names should not contain numbers. If a number is used for first and last name, a message show to tell the user that the name should not contain numbers.

Assignment05.py | Enrollments.json

```python
# ------------------------------------------------------------------------------------------ #
# Title: Working With Dictionaries And JSON Files
# Desc: Shows how work with dictionaries and json files when using a table of data
# Change Log: (Who, When, What)
#   JDuldulao,05/14/2025,Created Script
# ------------------------------------------------------------------------------------------ #
import json
import io as _io  # Needed to try closing in the finally block

# Define the Data Constants
FILE_NAME: str = 'Enrollments.json'

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 data variables
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 course name of the student entered by the user.
file = _io.TextIOWrapper  # This is the actual type of the file handler
menu_choice: str = ''   # Hold the choice made by the user.
student_data: dict = {}  # one row of student data
students: list = []  # a table of student data
message: str = ''  # Holds a custom message string
file_data: str = ''  # Holds combined string data separated by a comma.

# When the program starts, read the file data into a list (students)
# Extract the data from the file
# Use of try-except error handling
try:
    file = open(FILE_NAME, "r")
    students = json.load(file)
    file.close()
except FileNotFoundError as e:
    print("\nText file must exist before running this script!\n")
    print("-- Technical Error Message -- ")
    print(e, e.__doc__, type(e), sep='\n')
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:
    if file.closed == False:
        file.close()
```

Figure 4. First Part of Assignment 5 Code

Figure 5. Second Part of Assignment 5 Code

```
    elif menu_choice == "3":
        try:
            #  Save the data to the file
            file = open(FILE_NAME, "w")
            json.dump(students, file)
            file.close()

            print("-" * 50)
            print("The following data has been saved to the file:")
            for student in students:
                message = " {},{},{}."
                print(message.format( *args: student["FirstName"], student["LastName"], student["CourseName"]))
            print("-" * 50)
            continue
        except TypeError as e:
            print("Please check that the data is a valid JSON format\n")
            print("-- Technical Error Message -- ")
            print(e, e.__doc__, type(e), sep='\n')
        except Exception as e:
            print("-- Technical Error Message -- ")
            print("Built-In Python error info: ")
            print(e, e.__doc__, type(e), sep='\n')
        finally:
            if file.closed == False:
                file.close()

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

    else:
        continue
```

Figure 6. Third Part of Assignment 5 Code

Figure 6 above is the third part of the code. This is still part of the while loop and the user has the option to choose menu 3 or 4. Note that another exception has been added in menu_choice 3. This is specific to the validity of the JSON format. Example if on the JSON file, I will remove an opening brace, an exception is called.

Figure 7, 8 and 9 are the output of the program when run in PyCharm, Command Prompt and IDLE. Note that the assumption here is that Enrollments.json file does not exist when the program was first run.

```
C:\Users\jenni\Documents\Python\PythonCourse\PythonLabs\.venv\Scripts\

Text file must exist before running this script!

-- Technical Error Message --
[Errno 2] No such file or directory: 'Enrollments.json'
File not found.
<class 'FileNotFoundError'>

---- 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: 1

Enter the student's first name? Jenny
Enter the student's last name? Duldulao1
The last name should not contain numbers.
-- Technical Error Message --
Inappropriate argument value (of correct type).
The last name should not contain numbers.

---- 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: 1

Enter the student's first name? Jenny
Enter the student's last name? Duldulao
Enter the student's course name? 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
----------------------------------------
```

```
Enter your menu choice number: 2

-------------------------------------------------
 Jenny,Duldulao,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
----------------------------------------

Enter your menu choice number: 3

-------------------------------------------------
The following data has been saved to the file:
 Jenny,Duldulao,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
----------------------------------------

Enter your menu choice number: 4


Process finished with exit code 0
```

Figure 7: Output in PyCharm

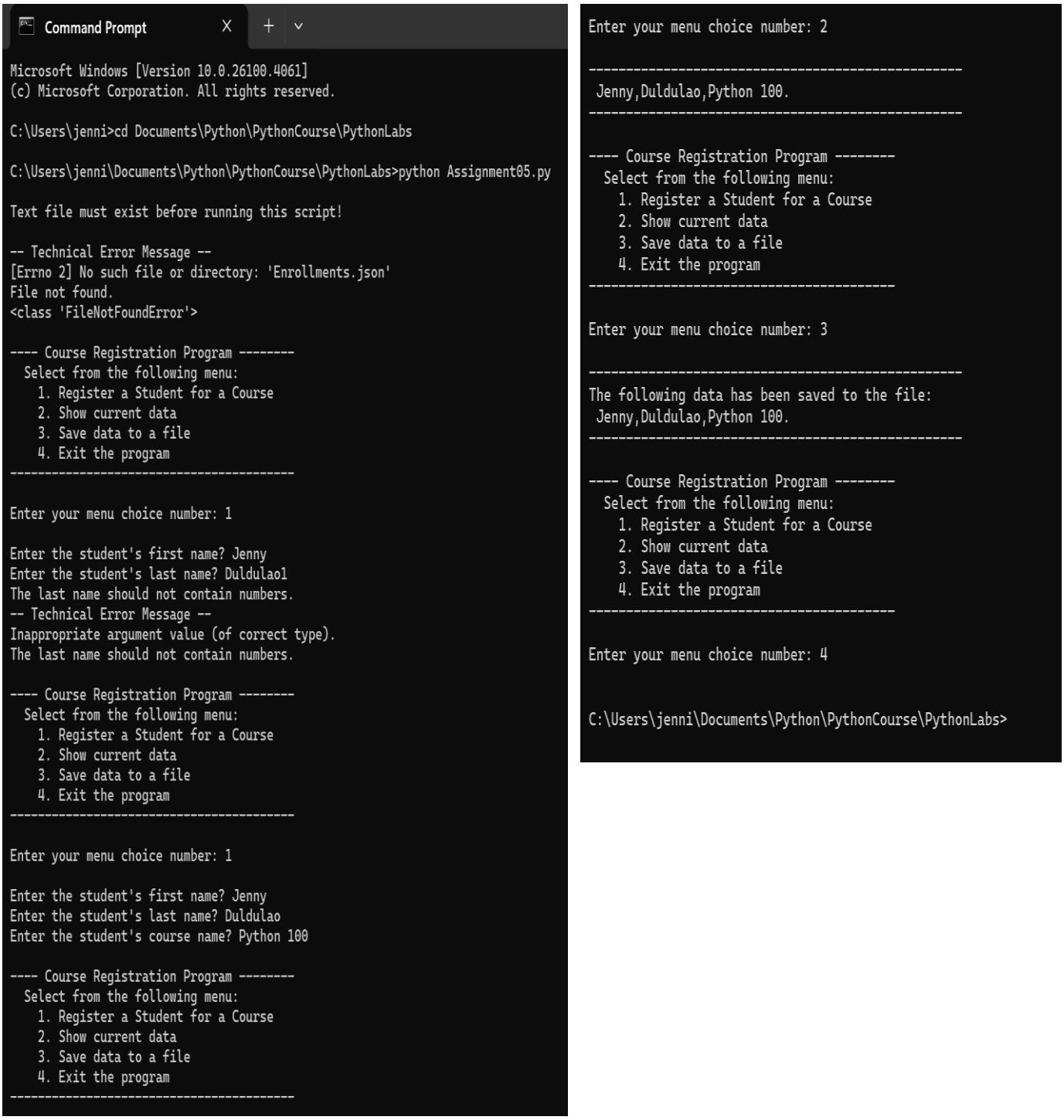

Figure 8. Output in Console

Figure 9. Output in IDLE

Summary

In this module, I get a deeper understanding of how dictionary works and JSON as the data file. At the beginning, I was having a hard time understanding the concept and coding them because of its complexity. However, as I keep going, it gets easier to understand the concept and in fact, I think, that using dictionary is more versatile as compared to list in assignment 4 as dictionary and JSON can handle the complexity of the program. I also learned the importance of error handling. When error handling is in place, the program can continue to run without crashing when exceptions are found.