Alberto Arriola
February 17, 2025
IT FDN 100 A
Assignment 05
GitHub URL: https://github.com/albertoarriola/IntroToProg-Python-Mod05

Dictionaries, JSON Files and GitHub

Intro

This week I practiced using Dictionaries to store data. Dictionaries contain key-value pairs that make it easier to access data than using Lists with indexes. The key is similar to a column title that describes the value data stored. The assignment for this week required reading a List of Dictionary records from an external JSON file, adding to the data, and then uploading the modified List of Dictionary records back to the external JSON file.

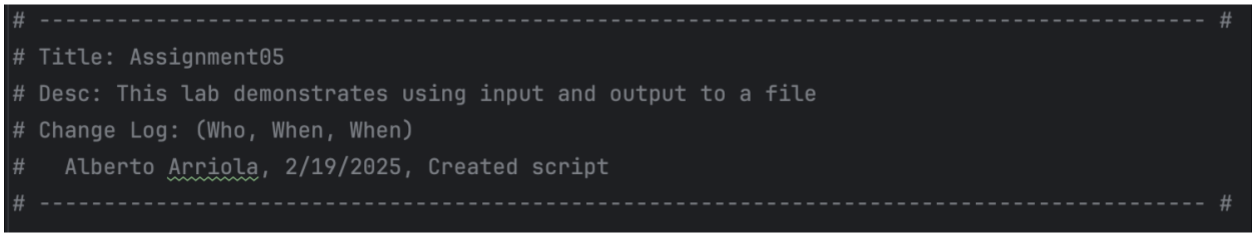
I used an external .csv file for Assignment 3 and 4 to store the student data. For Assignment 5, I used an external JSON file. A .csv file is useful for storing large amounts of data in a flat dataset, since it is basically a List of information separated by commas. A JSON file allows for a more complex data hierarchy using key-value pairs, much like Dictionaries in Python.

I also learned about structured error handling using the Try-Except block to manage any unexpected errors that might occur when I or someone else executes my code. I learned how to handle common errors that are already recognized by Python as well as customized errors that are more specific to the program that I am trying to create. By handling these errors, I am able to ensure that my code continues to run rather than allowing the program to crash or behave unexpectedly.

Finally, I signed up for a GitHub account and uploaded my assignment to the repository I created. GitHub is a platform that allows users to collaborate on coding projects by making files accessible online and tracking changes using version control.

Creating the Program

I started the program with a header section that includes the Title, Description and Change Log for the program.

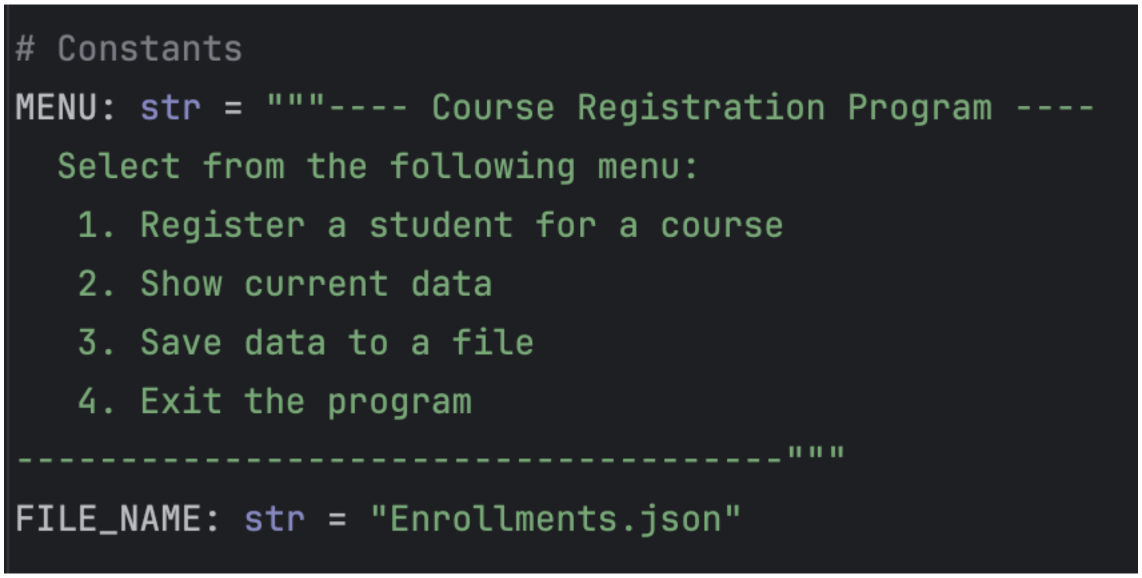


```
# ------------------------------------------------------------------------------------------ #
# Title: Assignment05
# Desc: This lab demonstrates using input and output to a file
# Change Log: (Who, When, When)
#   Alberto Arriola, 2/19/2025, Created script
# ------------------------------------------------------------------------------------------ #
```

Fig 1.1: Script Header

Constants and Variables

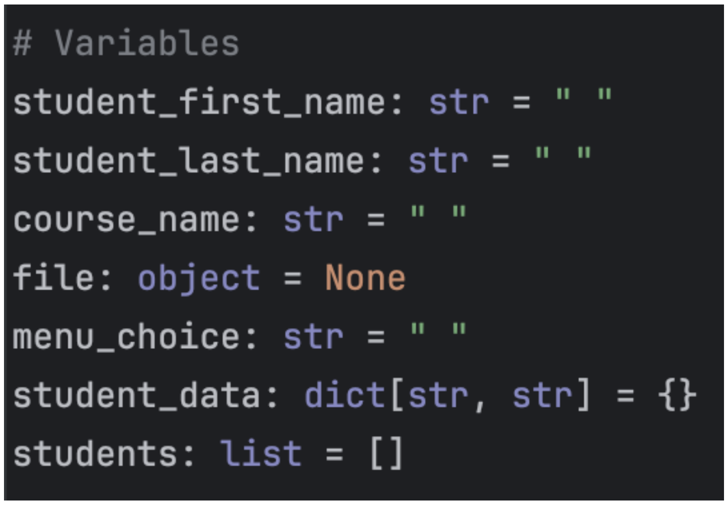
I created two constants as specified in the acceptance criteria. The first constant, MENU, is a string with multiple lines so I used triple quotes to contain the text. The second constant, FILE_NAME, is a string that contains the name of the external JSON file that the program reads from and later writes to.



```
# 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"
```

Fig. 1.2 Constants used in Assignment05.py.

I created five variables of type String, one of type Object, one of type List, and one of type Dictionary.

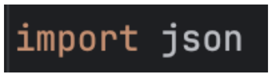


```python
# Variables
student_first_name: str = " "
student_last_name: str = " "
course_name: str = " "
file: object = None
menu_choice: str = " "
student_data: dict[str, str] = {}
students: list = []
```

Fig. 1.3 Variables used in Assignment05.py.

Import

I used an Import statement to access the JSON modules that would allow my program to use the functions necessary to read to and write from the JSON file.



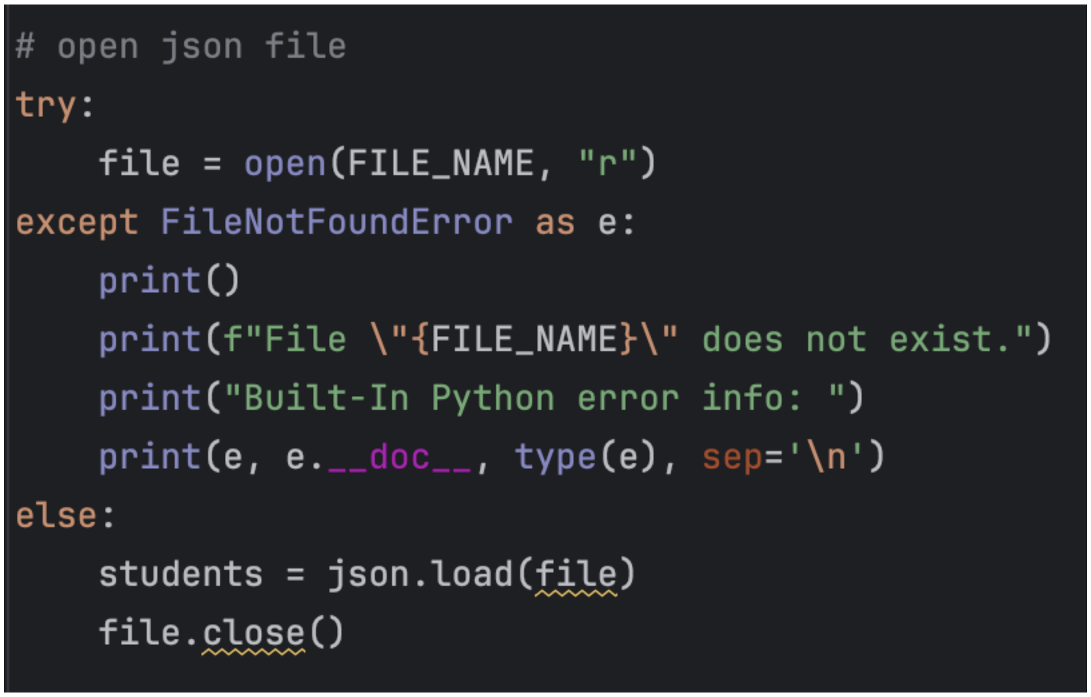
```python
import json
```

Fig. 1.4 Import JSON statement.

Open External JSON File

I start the program by attempting to open the external JSON file, Enrollments.json. I added exception handling using a Try-Except block that looks for a FileNotFoundError that occurs if the Enrollments.json file does not exist. If the file does not exist, the program prints out error messages and then skips down to the While Loop.

If the external file does exist, the error messaging is skipped and the code following the Else statement is executed. The data from the external JSON file is loaded into the students List variable, the external file is closed.



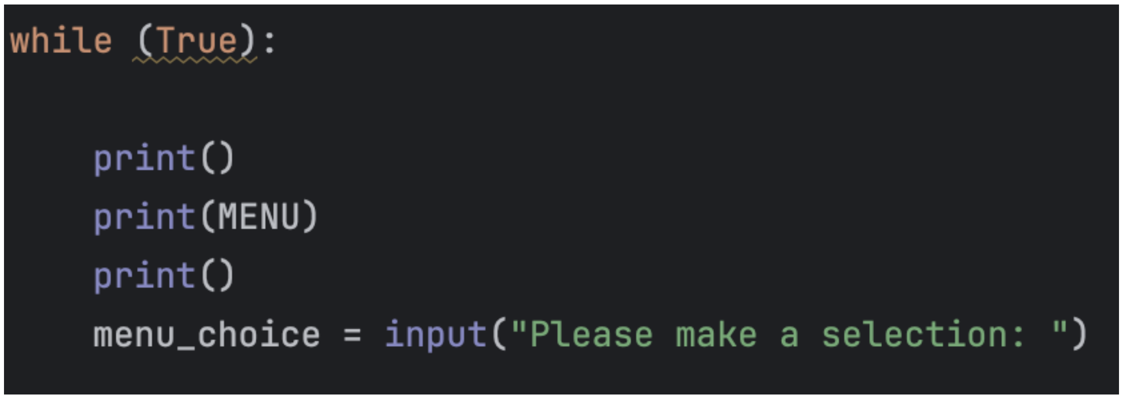
```
# open json file
try:
    file = open(FILE_NAME, "r")
except FileNotFoundError as e:
    print()
    print(f"File \"{FILE_NAME}\" does not exist.")
    print("Built-In Python error info: ")
    print(e, e.__doc__, type(e), sep='\n')
else:
    students = json.load(file)
    file.close()
```

Fig. 1.5 Open external JSON file.

While Loop

I start the program with a While Loop so that it will continue to run until the user enters “4” to exit the program.

Each time the loop executes, the Menu is displayed and the user is asked to input their choice. The user’s input is stored in the menu_choice variable. The menu_choice variable is then compared to different If-Elif-Else statement conditions and specific code is executed based on the user’s choice.



```
while (True):

    print()
    print(MENU)
    print()
    menu_choice = input("Please make a selection: ")
```

Fig. 1.6 Running the program in a While Loop.

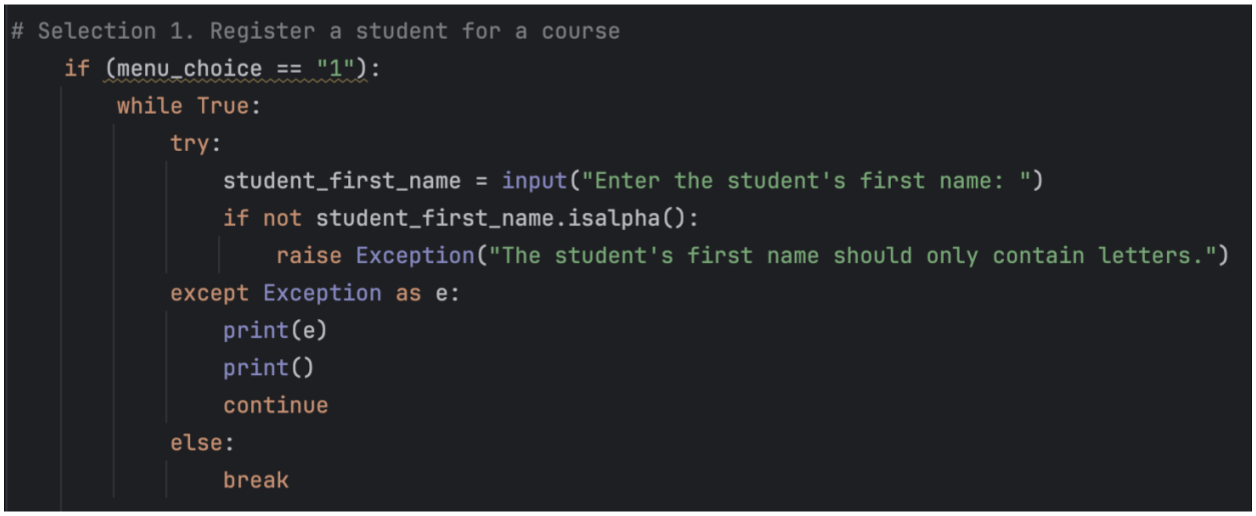
The If-Elif-Else Conditions

I created an If-Elif-Else statement to execute code based on the user’s choice.

When the user makes a choice from the menu, the input is stored in the menu_choice variable. A comparison operator is used to determine which section of code is executed.

Register a student for a course

If the user enters "1", then the program determines that they want to enter student information. The If statement is triggered by the condition "menu_choice == "1". The "==" operator is used to make the comparison because I want to make sure the code is only executed for this specific situation.

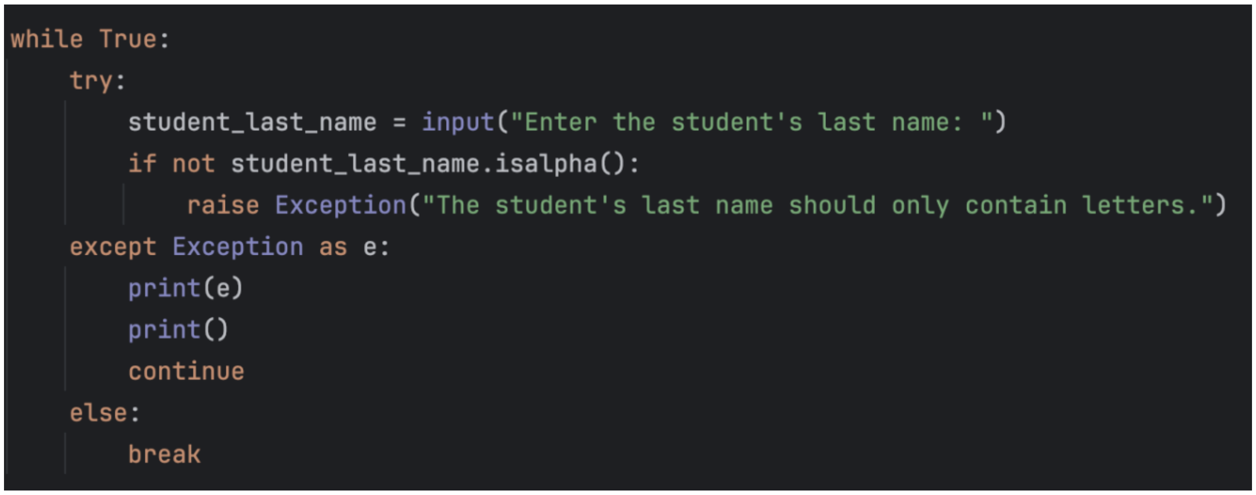


```
# Selection 1. Register a student for a course
    if (menu_choice == "1"):
        while True:
            try:
                student_first_name = input("Enter the student's first name: ")
                if not student_first_name.isalpha():
                    raise Exception("The student's first name should only contain letters.")
            except Exception as e:
                print(e)
                print()
                continue
            else:
                break
```

Fig. 1.7 Code section handling when the user selects "1 - Register a student for a course" and enters student's first name.

I used a nested While Loop so that the code requesting that the user enter the student's first name would continue to execute until a valid student name is entered. In the Try-Except block that checks if the inputted student's name contains non-alpha characters using the .isalpha() String function, an If statement prints out the custom Exception message, "The student's first name should only contain letters." if this is determined to be true. The continue statement returns the program to the start of the While Loop and the user is asked once again to enter the student's first name. The program does not break out of the While Loop until the student's name is determined to contain only alpha characters.

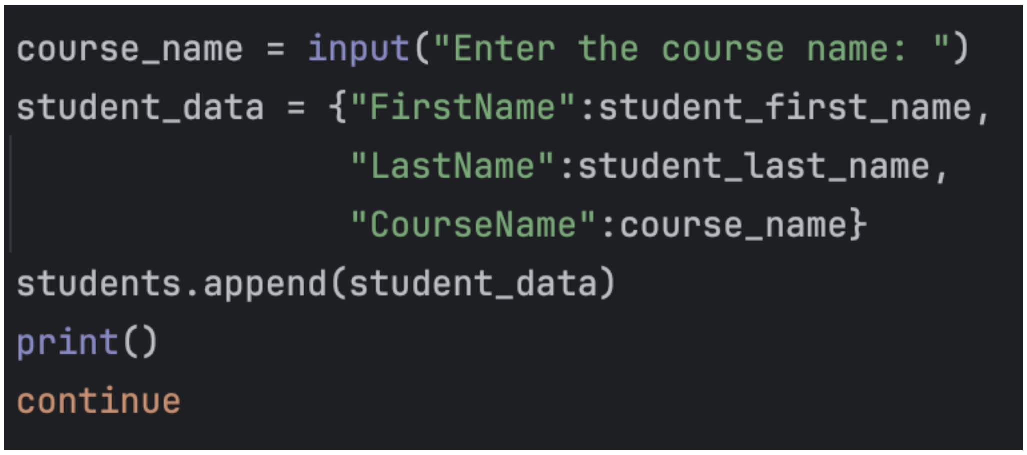
The code block asking the user to input the student's last name is similar, using a nested While Loop and custom Exception checking to make sure that the inputted student's last name only contains alpha characters. Again, the user continues to loop through the code to enter the student's last name until the requirement of containing only letters is satisfied.



```python
while True:
    try:
        student_last_name = input("Enter the student's last name: ")
        if not student_last_name.isalpha():
            raise Exception("The student's last name should only contain letters.")
    except Exception as e:
        print(e)
        print()
        continue
    else:
        break
```

Fig. 1.8 Code section handling when the user selects "1 - Register a student for a course" and enters student's last name.

The user then enters the course name and the data is saved to the student_data Dictionary variable. The student_data is appended to the students List and the continue statement returns the user to the Menu.

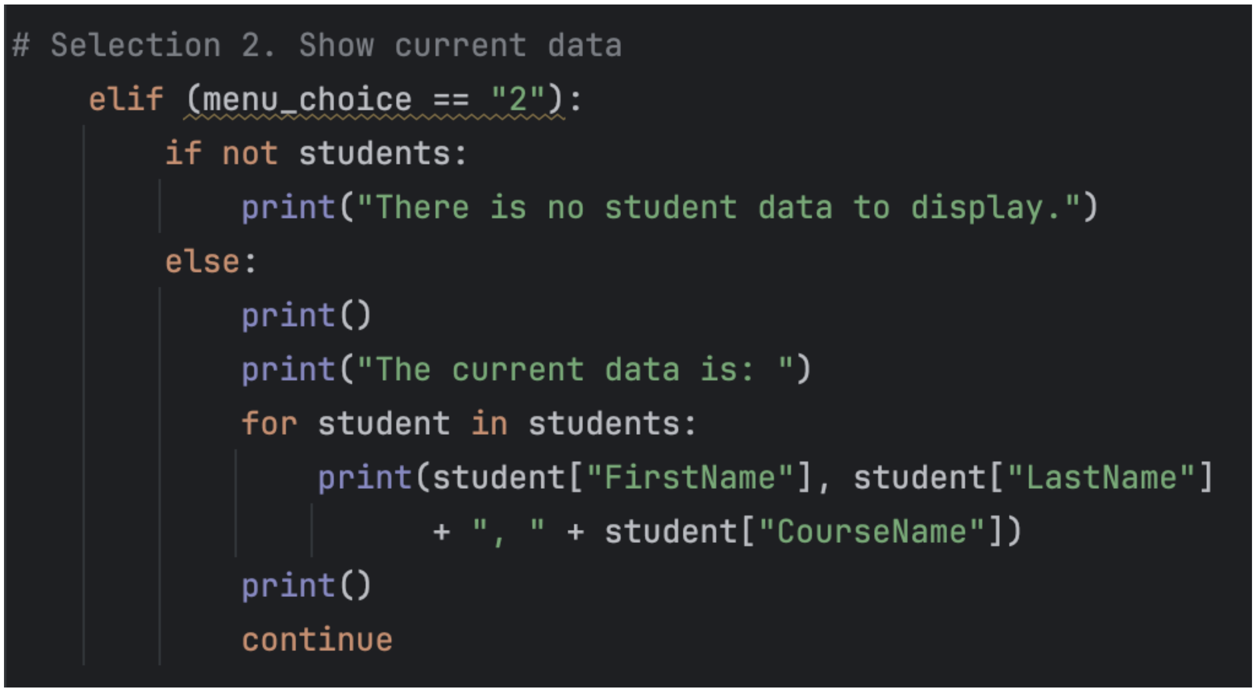


```python
course_name = input("Enter the course name: ")
student_data = {"FirstName":student_first_name,
                "LastName":student_last_name,
                "CourseName":course_name}
students.append(student_data)
print()
continue
```

Fig. 1.9 Code section for Course Name input and appending student data to the students List.

Show current data

If the user enters "2", then the program determines that they want to see the current data that has already been entered. The Elif statement is triggered by the condition "menu_choice == "2".

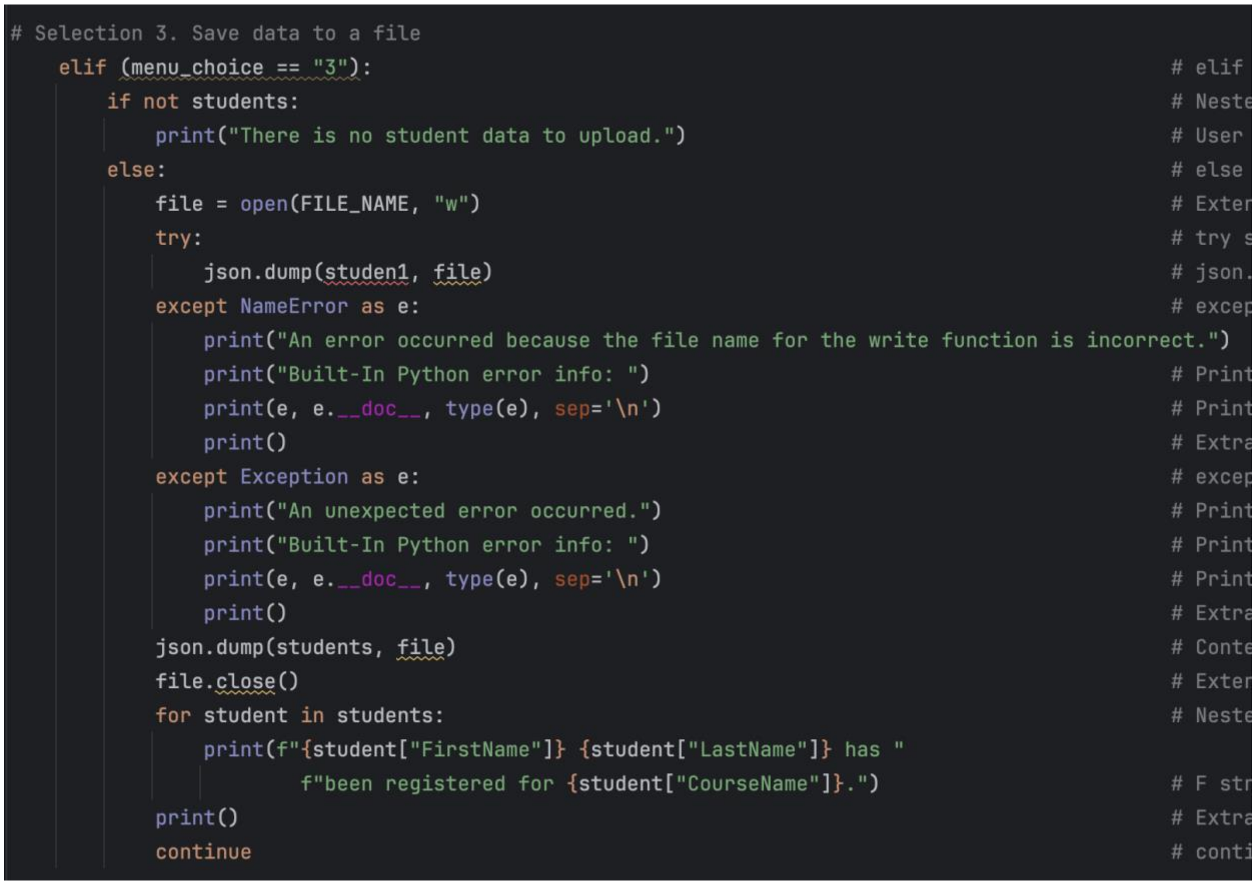

```python
# Selection 2. Show current data
    elif (menu_choice == "2"):
        if not students:
            print("There is no student data to display.")
        else:
            print()
            print("The current data is: ")
            for student in students:
                print(student["FirstName"], student["LastName"]
                      + ", " + student["CourseName"])
            print()
            continue
```
Fig. 1.10 Code section handling when the user selects "2 – Show current data".

I created a nested If statement to determine if student data has been recorded. For the if statement, the code evaluates if the students List contains any information by evaluating the "not students" statement. If student data has not been downloaded from an external file or entered by the user, then the value of "not students" is True and the text "There is no student data to display." is printed out.

If student data has been downloaded or entered, then the value of the "not students" statement is False and the Else statement is triggered. A nested For Loop is used to run through all the students in the students List variable. A print() statement is used to print out the data for each student in the List. The For Loop ends after all the students are run through. An extra line is printed out and the 'continue' statement returns the program to the main While Loop printing out the Menu again.

Save data to a file

If the user enters "3", then the program determines that they want to upload the current data to an external file. The Elif statement is triggered by the condition "menu_choice == "3".



```
# Selection 3. Save data to a file
    elif (menu_choice == "3"):                                                    # elif
        if not students:                                                          # Neste
            print("There is no student data to upload.")                          # User
        else:                                                                     # else
            file = open(FILE_NAME, "w")                                           # Exter
            try:                                                                  # try s
                json.dump(studen1, file)                                          # json.
            except NameError as e:                                                # excep
                print("An error occurred because the file name for the write function is incorrect.")
                print("Built-In Python error info: ")                            # Print
                print(e, e.__doc__, type(e), sep='\n')                            # Print
                print()                                                           # Extra
            except Exception as e:                                                # excep
                print("An unexpected error occurred.")                            # Print
                print("Built-In Python error info: ")                            # Print
                print(e, e.__doc__, type(e), sep='\n')                            # Print
                print()                                                           # Extra
            json.dump(students, file)                                             # Conte
            file.close()                                                          # Exter
            for student in students:                                              # Neste
                print(f"{student["FirstName"]} {student["LastName"]} has "
                        f"been registered for {student["CourseName"]}.")         # F str
            print()                                                               # Extra
            continue                                                              # conti
```

Fig. 1.11 Code section handling when the user selects "3 – Save data to a file".

Again, I created a nested If statement to determine if student data has been recorded. For the if statement, the code evaluates if the students List contains any information by evaluating the "not students" statement. If student data has not been downloaded from an external file or entered by the user, then the value of "not students" is True and the text "There is no student data to upload." is printed out.

If student data has been downloaded or entered, then the value of the "not students" statement is False and the Else statement is triggered. An external file with a title equal to the String value assigned to the FILE_NAME constant (Enrollments.csv) is opened using the .open() function and is assigned to the object_file Object variable. The "w" argument for the open() function overwrites any data previously written to the Enrollments.csv file with the latest data.

I used two Try-Except blocks to check for handle errors that might occur during the upload. The first one looks for an incorrect name for the external file. If the file name is incorrect, the NameError exception is sent and a custom error message is printed out along with the default Python messages.

I intentionally added a line attempting to write to a file with the wrong name so the NameError message is automatically triggered, so whenever the user select "3" to upload information to the external json file, this error situation occurs and the error messages are displayed.

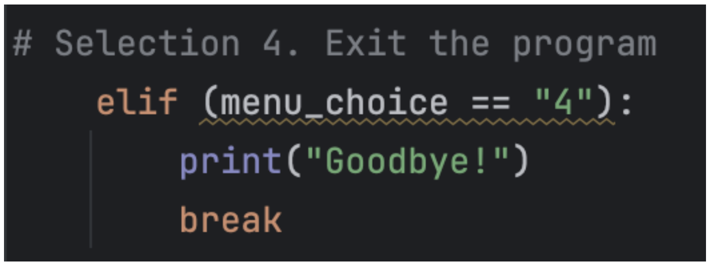
I also added another Try-Except block to catch any other errors that might occur.

After the error handling , the information from the students List is written to the external file using the json.dump() function and then the file is closed using the close() function.

A nested For Loop runs through all the students in the students List. For each student, a print statement formatted using F string is printed until all the students uploaded to the external file are listed as being registered. An extra line is printed out and the 'continue' statement returns the program to the main While Loop printing out the Menu again.

Exit the program

If the user enters "4", then the program determines that they want to exit. The Elif statement is triggered by the condition "menu_choice == "4".



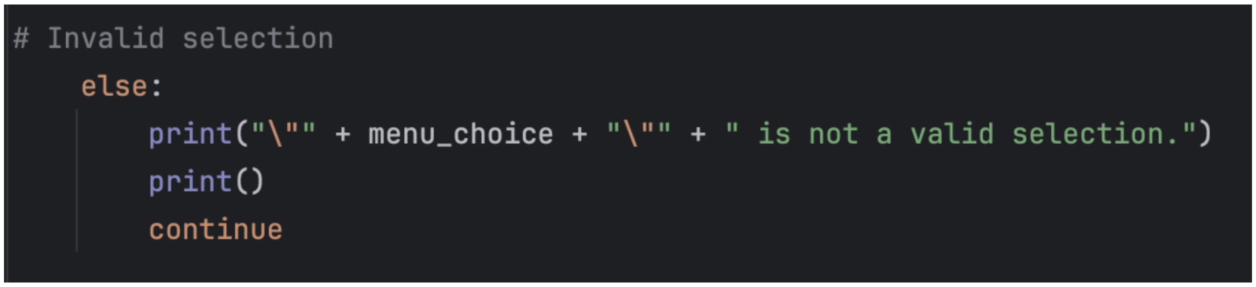
```
# Selection 4. Exit the program
    elif (menu_choice == "4"):
        print("Goodbye!")
        break
```

Fig. 1.12 Code section handling when the user selects "4 – Exit the program".

A print() statement outputs the string "Goodbye!". Then the 'break' statement exits the main While Loop and ends the program.

Invalid selection

If the user enters anything other than a 1,2, 3, or 4, then the program lets them know that they made an invalid selection. A print() statement prints out the user's entry and says that it is not a valid selection. An extra line is printed out and the 'continue' statement returns the program to the main While Loop printing out the Menu again.



```
# Invalid selection
    else:
        print("\"" + menu_choice + "\"" + " is not a valid selection.")
        print()
        continue
```

Fig. 1.13 Code section handling when the user enters an invalid selection.

Comments

Finally, I added comments to each line of code using the '#'. This will make it easier for other developers to figure out what I was trying to do with each line.

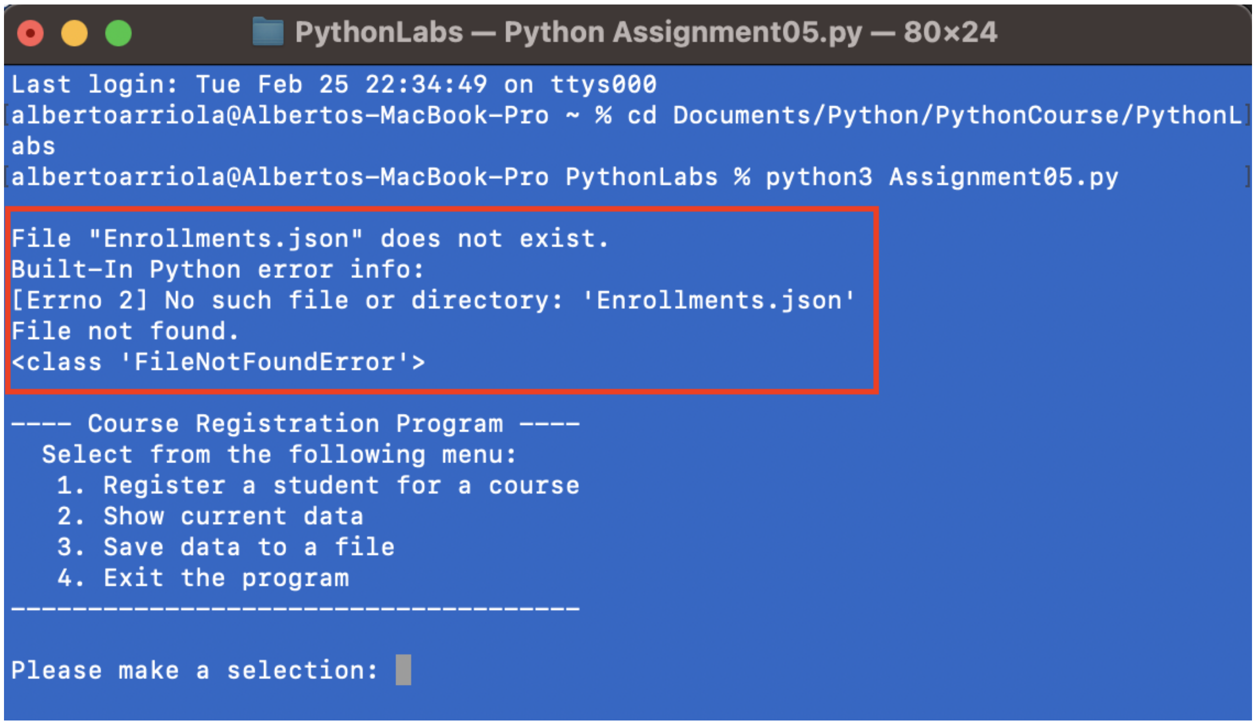
Testing the Program

Now that the scripting is complete, it is time to test the program. I ran it first using Terminal on my mac. I opened the Terminal app and then used the 'cd' command to point to the directory in which my script was saved.

I typed 'python3 Assignment05.py' to run the script.

Before running my script, I deleted the Enrollments.json file from the current directory so that the Try-Exception block surrounding the open() function would print the exception. I used the existing FileNotFoundError. A custom message was printed out as well as the default message for the error. This allowed the script to continue to execute even if the external file did not exist. If the external json file did exist, the else statement would read the file into the students List.

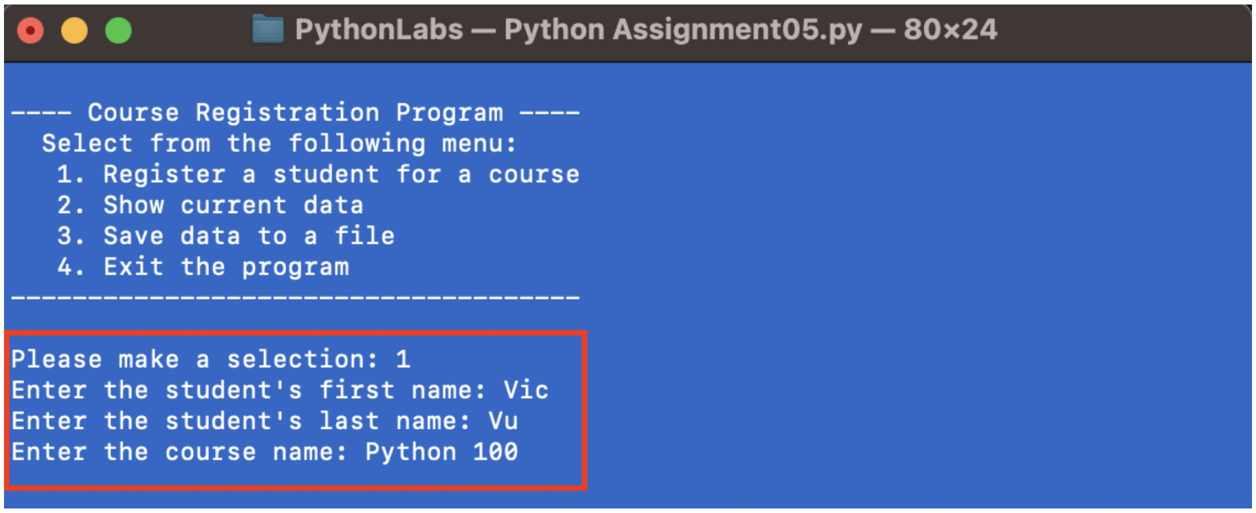
The program then presented the Menu and asked the user to make a selection.



```
Last login: Tue Feb 25 22:34:49 on ttys000
albertoarriola@Albertos-MacBook-Pro ~ % cd Documents/Python/PythonCourse/PythonL
abs
albertoarriola@Albertos-MacBook-Pro PythonLabs % python3 Assignment05.py

File "Enrollments.json" does not exist.
Built-In Python error info:
[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
-------------------------------------

Please make a selection: 
```

Fig. 1.14 Execution of the script in Terminal including the error message for FileNotFound.

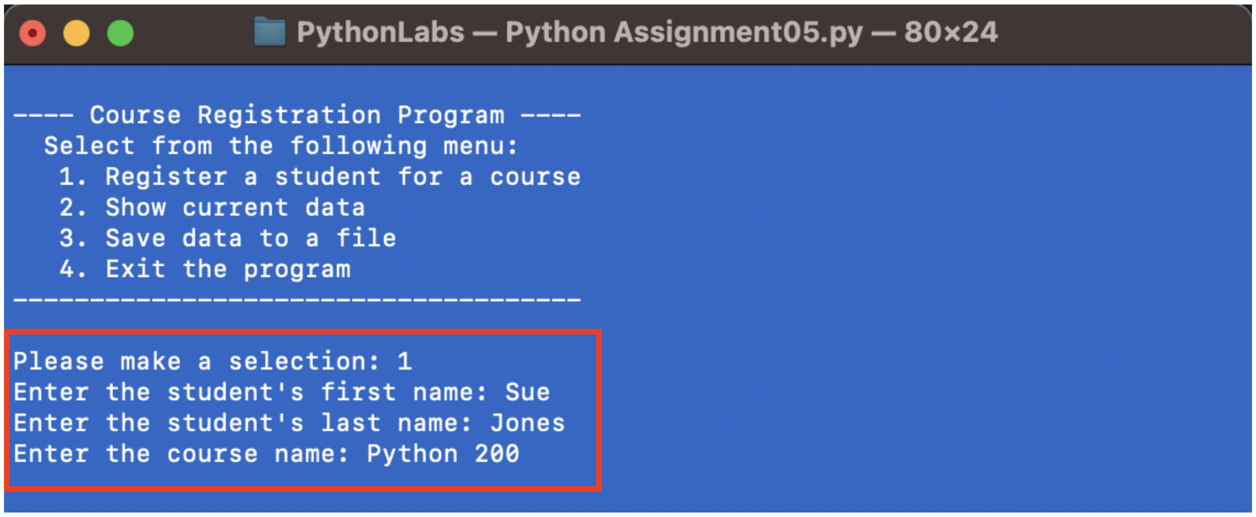
I entered “1” to select “Register a student for a course”. I was prompted for the student’s first name and entered “Vic”. Then I was prompted for the student’s last name and entered “Vu”. Finally, I was prompted for the course name and entered “Python 100”. After entering the data, I was returned to the Menu.



```

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

Please make a selection: 1
Enter the student's first name: Vic
Enter the student's last name: Vu
Enter the course name: Python 100
```

Fig. 1.15 Selecting “1 – Register a student for a course”.

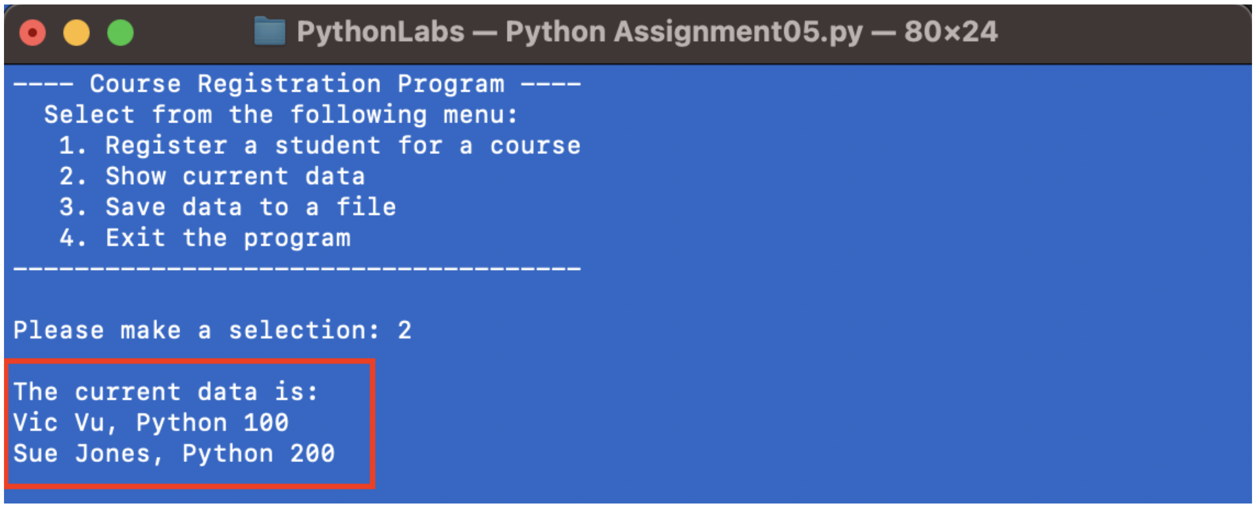
I entered “1” again to add a second student. This time, I entered “Sue Jones” for the class “Python 200”.



```
---- 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
-------------------------------------

Please make a selection: 1
Enter the student's first name: Sue
Enter the student's last name: Jones
Enter the course name: Python 200
```

Fig. 1.16 Selecting "1 – Register a student for a course" and entering a second student.

I entered "2" to select "Show current data" and the program printed out the first name, last name, and the course name for both students. Then it returned me to the Menu.

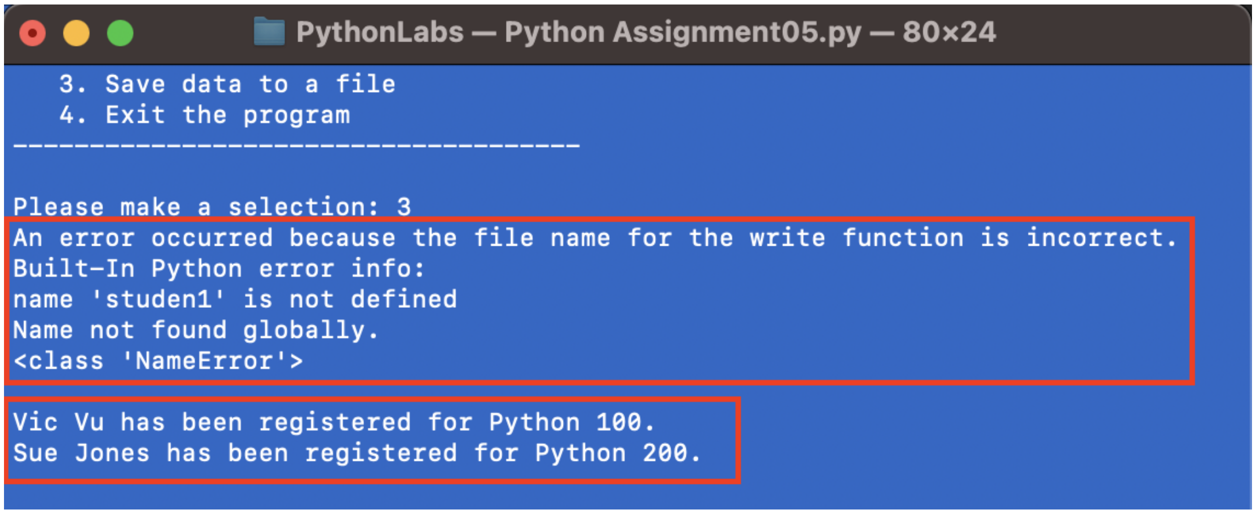


```
---- Course Registration Program ----
  Select from the following menu:
   1. Register a student for a course
   2. Show current data
   3. Save data to a file
   4. Exit the program
-------------------------------------

Please make a selection: 2

The current data is:
Vic Vu, Python 100
Sue Jones, Python 200
```

Fig. 1.17 Selecting "2 – Show current data".

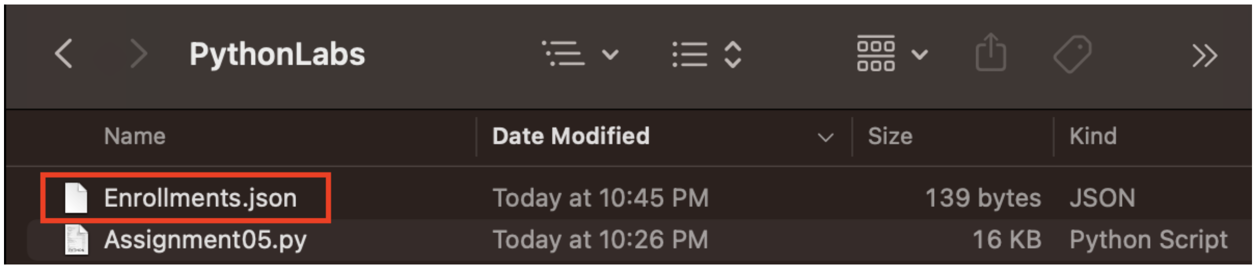
At the next Menu prompt, I entered "3" to select "Save data to a file". The NameError exception was automatically triggered and a custom message was printed out as well as the default message for the error. No other error occurred and the program continued.

The external "Enrollments.json" file was created in the same directory as the "Assignment05.py" program and the contents of the student List were written to it. Then it printed out the line "[student first name] [student last name] has been registered for [course name]." The output was printed out for both students. Then it returned me to the Menu.



```
   3. Save data to a file
   4. Exit the program
-------------------------------------

Please make a selection: 3
An error occurred because the file name for the write function is incorrect.
Built-In Python error info:
name 'studen1' is not defined
Name not found globally.
<class 'NameError'>

Vic Vu has been registered for Python 100.
Sue Jones has been registered for Python 200.
```

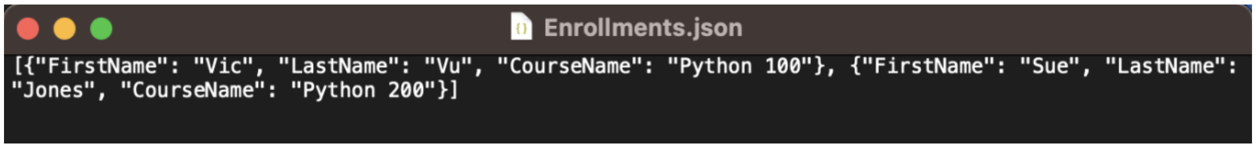
Fig. 1.18 Selecting "3 – Save data to file".



Name	Date Modified	Size	Kind
Enrollments.json	Today at 10:45 PM	139 bytes	JSON
Assignment05.py	Today at 10:26 PM	16 KB	Python Script

Fig. 1.19 Enrollments.json file in PythonLabs folder.

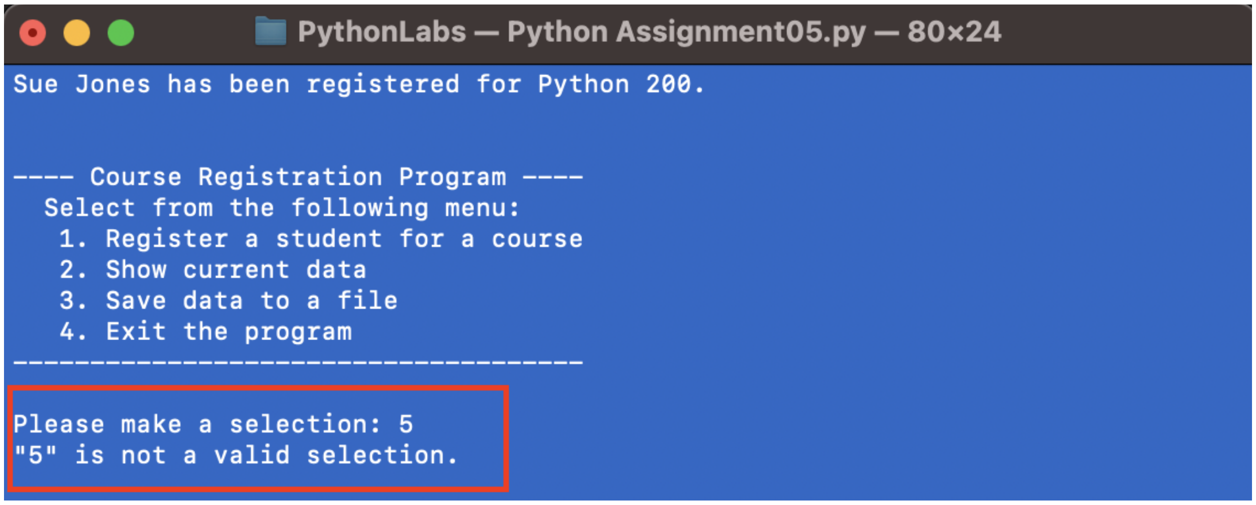
I opened the Enrollments.json file in TextEdit and verified that both registered students appear in the file.



```
[{"FirstName": "Vic", "LastName": "Vu", "CourseName": "Python 100"}, {"FirstName": "Sue", "LastName":
"Jones", "CourseName": "Python 200"}]
```

Fig. 1.19 Data in the Enrollments.json file.

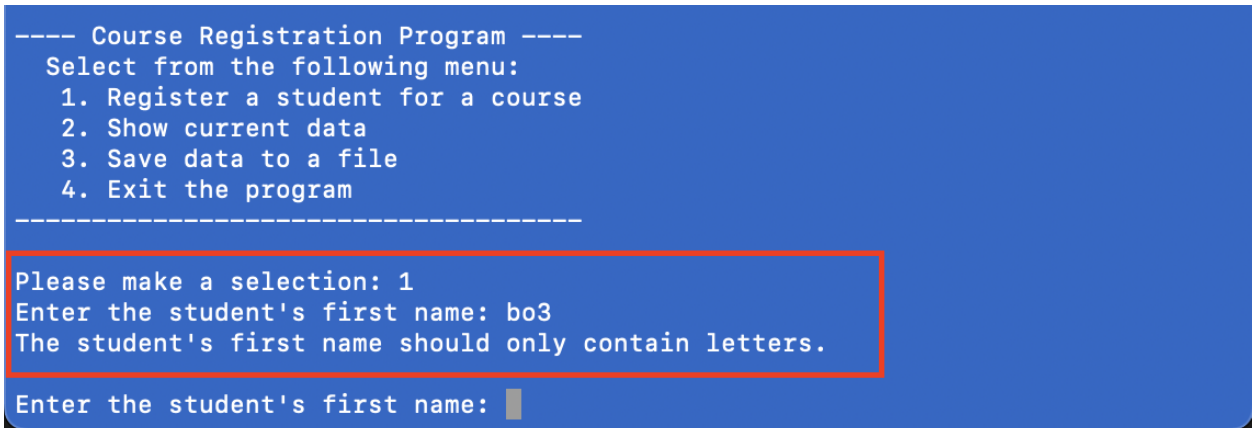
Back in Terminal, at the Menu I entered "5" to test the situation where a user does not make a valid selection. The program printed out "5 is not a valid selection. Please make a valid selection." and returned me to the Menu.



```
Sue Jones has been registered for Python 200.


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

Please make a selection: 5
"5" is not a valid selection.
```

Fig. 1.20 Making an invalid selection.

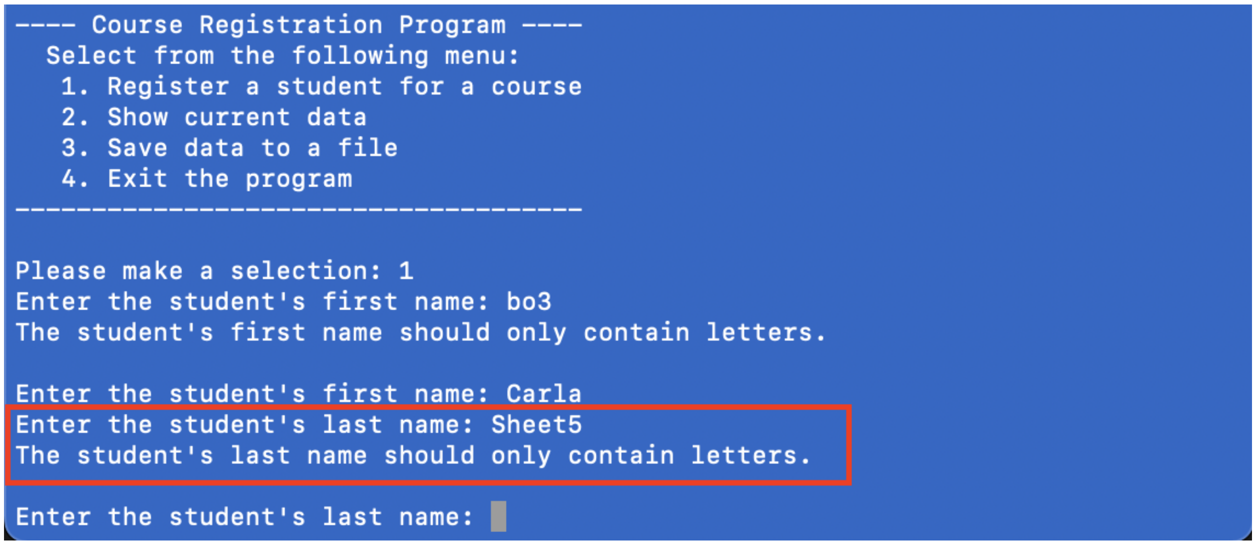
Then I selected "1" again to test my error handling. I verified that a custom error message was printed out if the user tried to enter something other than letters for the student's first name. The user was given another chance to enter a student's first name.



```
---- 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
-------------------------------------

Please make a selection: 1
Enter the student's first name: bo3
The student's first name should only contain letters.

Enter the student's first name:
```

Fig. 1.21 Error handling if user enters a non-alphanumeric character in the student's first name.

I also verified that the same error handling occurred when the user entered the student's last name.

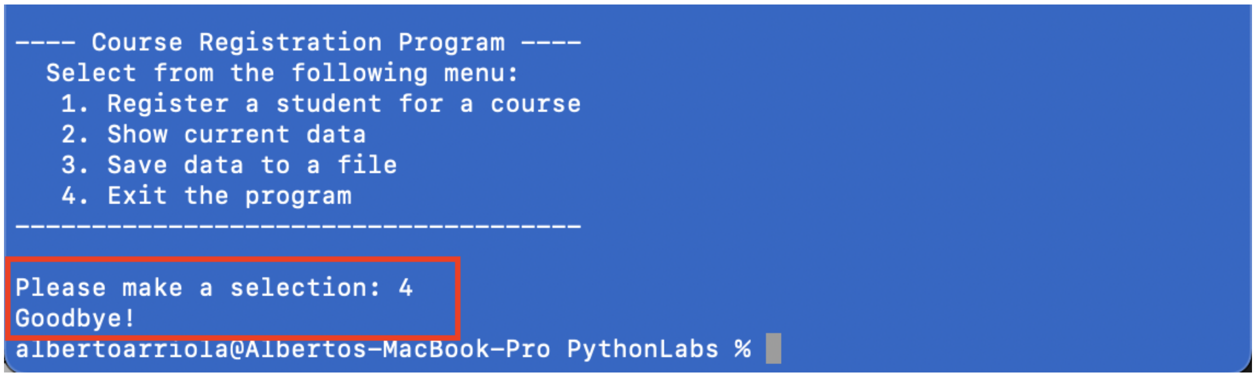


```
---- Course Registration Program ----
  Select from the following menu:
   1. Register a student for a course
   2. Show current data
   3. Save data to a file
   4. Exit the program
-------------------------------------

Please make a selection: 1
Enter the student's first name: bo3
The student's first name should only contain letters.

Enter the student's first name: Carla
Enter the student's last name: Sheet5
The student's last name should only contain letters.

Enter the student's last name: 
```

Fig. 1.22 Error handling if user enters a non-alphanumeric character in the student's last name.

I verified that the program continued after an acceptable first name and last name were entered.

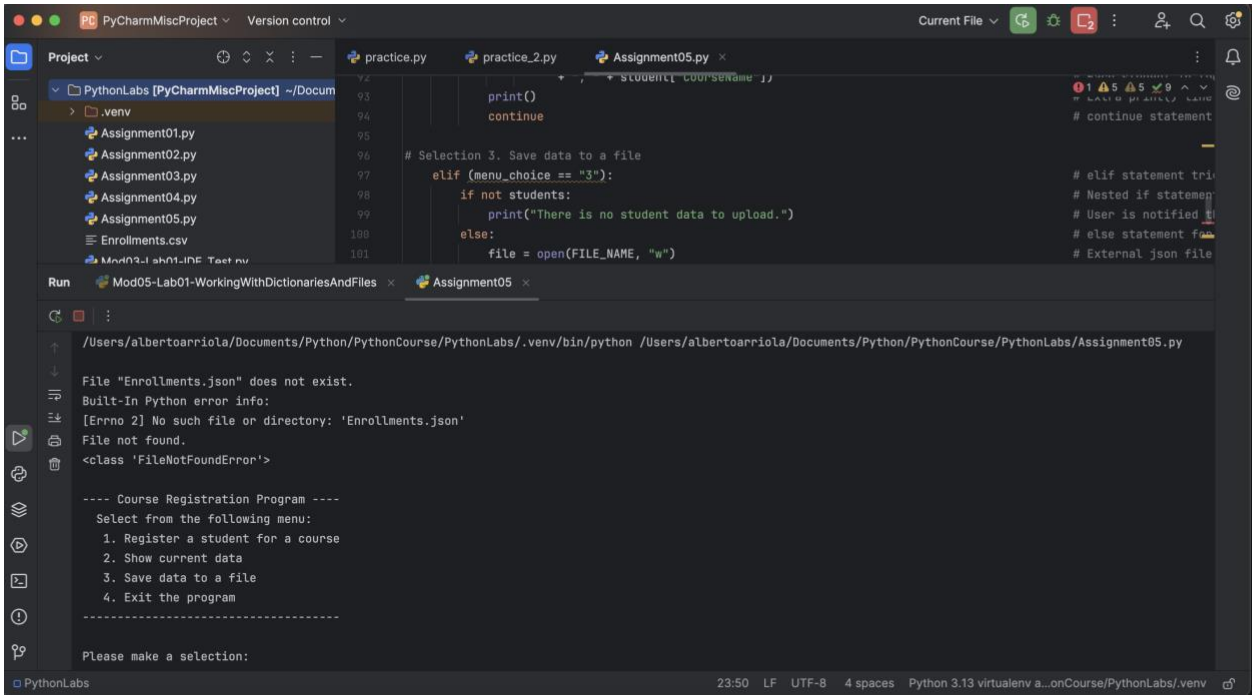
Finally, I entered "4" to select "Exit the program". The program prints the line "Goodbye" and ends.



```
---- 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
-------------------------------------

Please make a selection: 4
Goodbye!
albertoarriola@Albertos-MacBook-Pro PythonLabs % 
```

Fig. 1.23 Selecting "4 – Exit the program".

I also ran the script in PyCharm and verified that each of the selections behaved correctly. I deleted the Enrollments.csv file to verify again that program creates it if it doesn't exist.

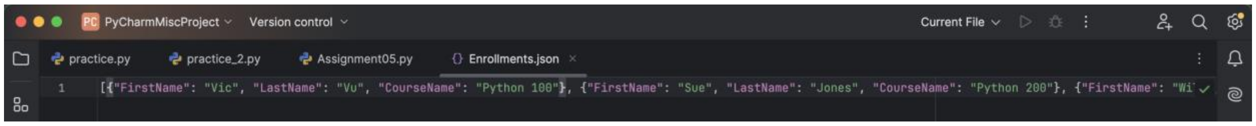


```
# Selection 3. Save data to a file
    elif (menu_choice == "3"):                                    # elif statement tri
        if not students:                                          # Nested if statemen
            print("There is no student data to upload.")          # User is notified t
        else:                                                     # else statement fo
            file = open(FILE_NAME, "w")                           # External json file
```

```
/Users/albertoarriola/Documents/Python/PythonCourse/PythonLabs/.venv/bin/python /Users/albertoarriola/Documents/Python/PythonCourse/PythonLabs/Assignment05.py

File "Enrollments.json" does not exist.
Built-In Python error info:
[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
-------------------------------------

Please make a selection:
```

Fig. 1.24 Execution of the script in PyCharm.

After adding students, I opened the Enrollments.json file in PyCharm to verify that the student data was successfully written to the external file.



```
[{"FirstName": "Vic", "LastName": "Vu", "CourseName": "Python 100"}, {"FirstName": "Sue", "LastName": "Jones", "CourseName": "Python 200"}, {"FirstName": "Wi
```

Fig. 1.25 The uploaded data is written to the Enrollments.json file.

GitHub

I created a personal account in GitHub and uploaded my assignment for Module 5. Here is the link to my project: https://github.com/albertoarriola/IntroToProg-Python-Mod05 .

Summary

In Module 5, I learned how to use Dictionary collections to store data. Dictionaries are much easier to index due to the use of Key-Value pairs. Working with a List of Dictionaries, I was able to convert my assignment from Module 4 into a similar program but using data that is easier to decipher.

I also became familiar with JSON files which seem to go hand in hand with Dictionaries since they also use Key-Value pairs. JSON files are easy to read since each piece of data can be described by the associated key. This is very different from .csv files that can look like a random collection of data separated by commas to a user.

My program continued to evolve not just behind the scenes in terms of the data types and external file used. It now incorporates error handling. The most obvious instance is requiring the user to enter a first and last names that only contain letters. By using Try-Except blocks in my code, it is now less likely to crash due to unexpected issues.

Finally, I created a GitHub account and uploaded a copy of my code to my account on the platform. This allows others to view my program if they want to see how I completed the assignment and give any pointers if they find any issues. GitHub is a very powerful tool for collaboration between programmers.