

Traci McGrew

Randall Root

IT FDN 110 A

Assignment 05 due 5/15/24

<https://github.com/tmcgrew/IntroToProg-Python-Mod05>

Advanced Collections and Error Handling

Introduction

In this paper I will go over how I completed creating a Python program like the program in Assignment04, but which also demonstrates using data processing using dictionaries and exception handling. I will also touch on some concepts from the module, and my observations while doing the assignments in the module.

Lists

Lists are multiple values treated as a single object. Lists are surrounded by brackets `[]`. An example of a list is `my_fruits = ['bananas', 'apples', 'oranges', 'grapes']`. You can access the different items on the list individually by calling them by their index position in the list. For example, `print(my_fruits[0])` would print *bananas*. Lists are also mutable.

Dictionaries

Dictionaries are like a list of *key : value* pairs. They are declared using this syntax:
`cars: dict = {"make": "Hyundai", "model": "Kona", "year": 2019, "mileage": 50000}`

You access dictionaries not using an index like a list but using the key value of the key : value pair. For example, `print(car["make"])` would print "Hyundai". You can also loop through them using either keys or values or both using the `.item()` function. That looks like

```
for key, value in cars.items():

    print(f" {key} = {value}")
```

Since variable names are snake_casing it is a good idea to use TitleCasing for the keys that way you can differentiate them. Also, since using JSON files double quotes "" should be used around the key's name.

Read Data from a File Into a Dictionary

Reading a CSV file and putting it into a dictionary involves opening it in read mode "r"

```
file = open(FILE_NAME, "r")
```

Extracting the data from the file line by line

```
for row in file.readlines():
```

Transform the data from the file by splitting it at the commas and putting each row into a list and stripping off \n at the end. Then, adding the list it made to a dictionary. Add dictionary then to table (list of dictionaries).

```
list = row.split(',') #returns a list could also strip it here with row.strip().split(',')

instead of below
```

```
dict = {"KeyName": list[0], "AnotherKey": list[1], "OtherKey": list[2].strip()}

table.append(dict)
```

Close the file.

```
file.close()
```

Writing a Dictionary to a File

Writing a dictionary to a CSV file involves opening it in write mode “w” or append mode “a”:

```
file = open(FILE_NAME, “w”)
```

For each row in the table (list of dictionaries), you need to use the dictionary row to create a comma separated string and write the string you made into the file. Make sure to put a \n at the end of the string you create.

for row in table:

```
    string = f”{row[“Key”]}, {row[“AnotherKey”]}, {row[“OtherKey”]} \n”.
```

```
    file.write(string)
```

Close the file to save the information.

```
file.close()
```

Templates

Most professional integrated development environments including PyCharm have a way to make templates. This can help make your code uniform and consistent and make it easier to do script headers in each file. In the lecture, Professor Root showed that the way to save something as a template was under the *Tools* menu in PyCharm. However, it’s now under the *File* menu.

Try – Except

Try – Except is a form of structured error handling. It helps improve your scripts by managing errors and gives you the ability to provide a more user-friendly error message.

JavaScript Object Notation File (JSON)

JSON files have key – value pairs like dictionaries. Keys need to be strings and enclosed in double “ “ quotes. However, the values can be other data types. Data is organized with { } just like a Python dictionary for objects and square brackets [] like a list for lists. Also, just like a dictionary, commas separate the key-value pairs. An example would be:

```
{  
  "Make": "Hyundai",  
  "Model": "Kona",  
  "Mileage": 50000",  
  "Used": True,  
  "BodyColors": ["bluish grey", "grey"],  
}
```

CSV vs JSON files

Whether to use a JSON file or CSV file depends on what you are trying to do. JSON files have key, value pairs and lend themselves to more complex and nested data structures. Because of the key-value pairs, it's more readable by humans. Some of the uses of JSON files are for data exchange, configuration files, data storage, and API responses. CSV files are flat files that store rows and columns. They don't support nested structures instead they mainly store data as plain text separated by a delimiter which is usually a comma. Since CSV files are structured this way, they suited for large datasets that use a tabular structure like databases and

spreadsheets. CSV files are often used for data import and export for analysis and reporting. They are often much smaller as well.

Python JSON Module

Python's JSON module makes it much more straightforward to read and write from a table (list) of dictionaries to a .json file and read from a .json file into a dictionary. Examples of both are below:

```
import json #at top of module
```

```
#reads from a file
```

```
file = open(FILE_NAME, "r")
```

```
list = json.load(file)
```

```
file.close ()
```

```
#writes to a file
```

```
file = open(FILE_NAME, "w")
```

```
json.dump(list, file)
```

```
file.close ()
```

Program Assignment 05

I created a program that was like the program in Assignment04, but builds on those concepts to also demonstrate using data processing using dictionaries, JSON files, and exception

handling. This includes reading from a CSV file and making a dictionary with each row of the information it got and putting those rows into a table (list of dictionaries).

```

38 student_data: dict[str,str] = {} # one row of student data now a dict instead of a list
39 students: list[dict[str,str]] = [] # table of student data (list of dictionaries)
40
41 # When the program starts, read the file data into a list of lists (table)
42 file = open(FILE_NAME, "r")
43 # Extract the data from the file
44 for row in file.readlines():
45     # Transform the data from the file
46     student_data = row.split(',') #could also row.strip().split(',') here instead of below
47     #make a dictionary
48     student_data = {"FirstName": student_data[0], "LastName": student_data[1], "CourseName": student_data[2].strip()}
49     # Load it into our collection (list of dictionaries)
50     students.append(student_data)
51 file.close()
52

```

Figure 1 Instead of reading it into a list of lists it now reads it into a dictionary

It also involved writing a dictionary to a CSV file by converting each row of the table (list of dictionaries) to a string first.

```

91
92 # open() function to open a file in the desired mode ("w" for writing, "a" to append).
93 file = open(FILE_NAME, "w")
94
95 for student in students:
96     # Create a string representation for each student's data
97     csv_data = f"{student['FirstName']},{student['LastName']},{student['CourseName']}\n" # \n important here
98
99     # Write the data to the file
100     file.write(csv_data)
101
102 # print("Data Recorded!\n")
103 # closes the file to save the information
104 file.close()
105

```

Figure 2 Writing a dictionary to a CSV file by converting each row of the table (list of dictionaries) to a string first

This was then reworked using the JSON module in Python. That allowed me to read and write to a JSON file instead of a CSV file and directly using a table (list of dictionaries). The JSON module needed to be imported at the top of the module to be able to use it. Figures 3 and 4 show how it can now be read and written easier as a dictionary without converting it.

```

49 student_data: dict[str, str] = {} # one row of student data now a dict instead of a list
50 students: list[dict[str, str]] = [] # table of student data (list of dictionaries)
51
52 try:
53
54     # When the program starts, read the file data into a list of dictionaries (table)
55     file = open(FILE_NAME, "r")
56     # Extract the data from the file
57     students = json.load(file) # must import json above
58     # now students contains the parsed JSON data as a Python list of dictionaries
59     # students is the list and student are the dictionaries.
60

```

Figure 3 Reading a dictionary from a file using the JSON module

```

135
136
137
138
139
140
141
142
    try:
        # Save the data to a file
        # open() function to open a file in the desired mode ("w" for writing, "a" to append).
        file = open(FILE_NAME, "w")
        json.dump(students, file)
        # closes the file to save the information
        file.close()

```

Figure 4 Writing a dictionary to a file using the JSON module

The program takes input from the user both to register a student and save that info to a JSON file, but displays a message about a student's registration for a Python course and those previously registered already by what was previously saved to the JSON file.

The program displays a menu of choices. It continues by way of a while loop to continually let the user register until they decide to quit. There is a series of if statements that check to see what option is chosen and act accordingly. The program starts by reading from the JSON file that was already saved and making a dictionary with each row of the information it got and putting those rows into a table (list of dictionaries). There is also now exception handling when the reading of the JSON file is attempted. Figure 5 shows that exception handling where it reads from the JSON file.

```

52  ✓ try:
53
54      # When the program starts, read the file data into a list of dictionaries (table)
55      file = open(FILE_NAME, "r")
56      # Extract the data from the file
57      students = json.load(file) # must import json above
58  ✓  # now students contains the parsed JSON data as a Python list of dictionaries
59      # students is the list and student are the dictionaries.
60
61  ✓ except FileNotFoundError as e:
62      print("Text file must exist before running this script!\n")
63      print("-- Technical Error Message -- ")
64      print(e, e.__doc__, type(e), sep='\n')
65      print("Creating file since it doesn't exist")
66      file = open(FILE_NAME, "w")
67      json.dump(students, file) #putting empty list in upon creation
68  ✓ except JSONDecodeError as e:
69      print("-- Technical Error Message -- ")
70      print(e, e.__doc__, type(e), sep='\n')
71      print("Data in file isn't valid. Resetting it...")
72      file = open(FILE_NAME, "w")
73      json.dump(students, file) #putting empty list in upon creation
74  ✓ except Exception as e:
75      print("There was a non-specific error!\n")
76      print("-- Technical Error Message -- ")
77      print(e, e.__doc__, type(e), sep='\n')
78  ✓ finally:
79      if file.closed == False:
80          file.close()
81

```

Figure 5 Exception handling where it reads from a JSON file

If the user chooses menu item 1, the program prompts the user to enter their first and last name and then the course name. It gets these by the use of `input()` and variables. There is now exception handling around the first and last name in case the user enters non-alpha numeric input on either of those fields. However, entering the information on this option, doesn't save what they entered to a file, so I prompt the user to then choose option number 3 to complete their registration. Not visible to the user, it also takes what was entered into the variables and makes a

dictionary row. It then enters that row it made into a table (list of dictionaries) by appending the row just made to it.

If user chooses item 2, it presents all the collected data both from the JSON file and that has been entered by the user in item 1, by looping through the rows of the table (list of dictionaries) and presenting each row in a f-string. This is similar functionality to assignment04. It's just now using dictionaries instead of only lists and grabbing it from a JSON file instead of a CSV.

Choosing menu item 3, the program writes the table (list of dictionaries) to the JSON file. It also displays to the user what is written to the file by looping through the information and presented by way of a formatted f-string. This also now has exception handling when it attempts to write to the JSON file.

To quit the program, menu option 4 is chosen. This is necessary or the while loop will keep going.

The module ran great in PyCharm but in running it in Terminal I kept getting the error shown in figure 6.

```
print(f'{student["FirstName"]} {student["LastName"]} is registered for {student["CourseName"]}.'.)
SyntaxError: f-string: unmatched '['
((base) tracimcgre@Tracis-Air A05 % python "Assignment05.py"
File "/Users/tracimcgre/Desktop/UW/IT_FDN_110_A/My_Work/My_Module_05/A05/Assignment05.py", line 128
print(f'{student["FirstName"]} {student["LastName"]} is registered for {student["CourseName"]}.'.)
SyntaxError: f-string: unmatched '['
```

Figure 6 Error message trying to run the module in Terminal

There didn't seem to be anything wrong with the f-string but kept trying different things.

Eventually putting the keys in single quotes inside the bracket worked.

Figures 7 and 8 show the module running in PyCharm and figures 9 and 10 in Terminal.

Figure 11 shows the contents of the JSON file Enrollments.json To view the code of my

module, here is a link to my GitHub repository <https://github.com/tmcgrew/IntroToProg-Python-Mod05>

```
--- 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
-----

Your selection?: 1
Please enter your first name: Billy
Please enter your last name: Bowler
Please enter the course name: Python 100

Thank you! Please now select '3' to save the registration to a file.

--- 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
-----

Your selection?: 2
Bob Smith is registered for Python 100.
Jenny Jones is registered for Python 100.
Steve Stoneman is registered for Python 100.
Libby Ludtke is registered for Python 100.
Billy Bowler is registered for Python 100.

Please now select '3' to save the registrations you entered to a file.
```

Figure 7 Module running in PyCharm

```
--- 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  
-----  
Your selection?: 3  
Bob Smith is fully registered for Python 100.  
Jenny Jones is fully registered for Python 100.  
Steve Stoneman is fully registered for Python 100.  
Libby Ludtke is fully registered for Python 100.  
Billy Bowler is fully registered for 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  
-----  
Your selection?: 4  
  
Process finished with exit code 0
```

Figure 8 Continuation of figure 7

```
[(base) tracimcgreww@Tracis-Air A05 % 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
-----
Your selection?: 1
Please enter your first name: Gerry
Please enter your last name: George
Please enter the course name: Python 100

Thank you! Please now select '3' to save the registration to a file.

--- 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
-----
Your selection?: 2
Bob Smith is registered for Python 100.
Jenny Jones is registered for Python 100.
Steve Stoneman is registered for Python 100.
Libby Ludtke is registered for Python 100.
Billy Bowler is registered for Python 100.
Gerry George is registered for Python 100.

Please now select '3' to save the registrations you entered to a file.
```

Figure 9 Module running in Terminal

```

--- 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
-----

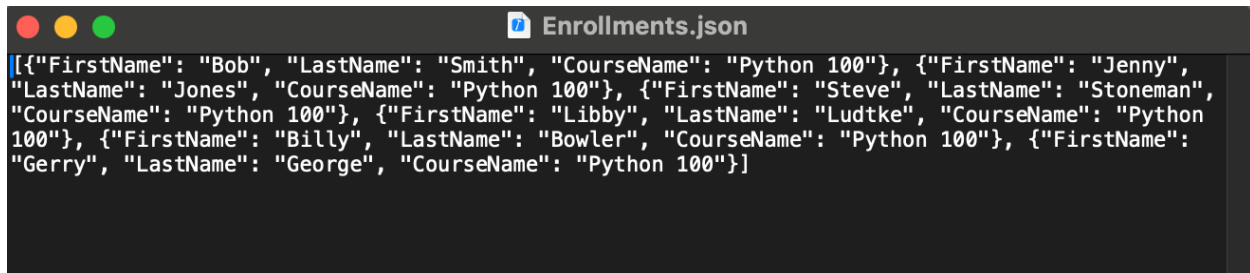
Your selection?: 3
Bob Smith is fully registered for Python 100.
Jenny Jones is fully registered for Python 100.
Steve Stoneman is fully registered for Python 100.
Libby Ludtke is fully registered for Python 100.
Billy Bowler is fully registered for Python 100.
Gerry George is fully registered for Python 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
-----

Your selection?: 4
(base) tracimcgrew@Tracis-Air A05 %

```

Figure 10 Continuation of figure 9



```

Enrollments.json
[{"FirstName": "Bob", "LastName": "Smith", "CourseName": "Python 100"}, {"FirstName": "Jenny",
"LastName": "Jones", "CourseName": "Python 100"}, {"FirstName": "Steve", "LastName": "Stoneman",
"CourseName": "Python 100"}, {"FirstName": "Libby", "LastName": "Ludtke", "CourseName": "Python
100"}, {"FirstName": "Billy", "LastName": "Bowler", "CourseName": "Python 100"}, {"FirstName":
"Gerry", "LastName": "George", "CourseName": "Python 100"}]

```

Figure 11 Showing the contents of the JSON file Enrollments.json

Summary

I went over some core concepts from the module, and then showed a Python program I created that was like the program in Assignment04, but builds on those concepts to also demonstrate using data processing using dictionaries, JSON files, and exception handling.