Name: Michael Keel, mkeel

Date Submitted: 05/21/25

Course: Foundations of Programming: Python

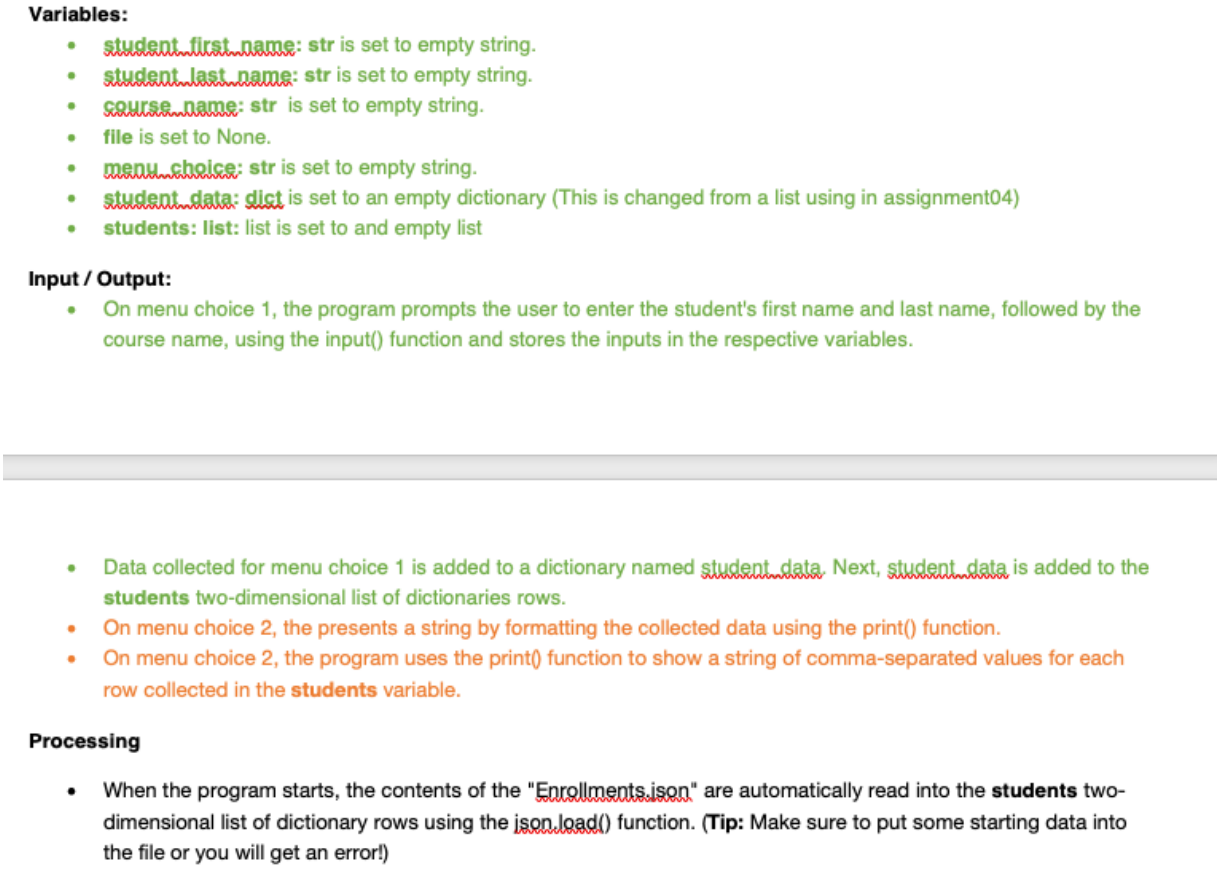
# Assignment 05

## Introduction

In this document I will review the events from Week 5 of Foundations of Programming: Python. I started this week with review of the Module Notes and Videos to clarify the specifics of data collections, libraries, and the incorporation of JSON files. I have the convenience of working with .json files almost daily, so this week was useful to see the introduction of the topic from the ground floor.

## Continuations and Pseudocode

The collection of tasks, inputs, outputs, tests, and error handling now exceeds a routine level of information that can readily be accessed by a readthrough of the Assignment. To aid in my own project management, I have taken to color coordination of completion of each line.



**Variables:**

- **student_first_name: str** is set to empty string.
- **student_last_name: str** is set to empty string.
- **course_name: str** is set to empty string.
- **file** is set to None.
- **menu_choice: str** is set to empty string.
- **student_data: dict** is set to an empty dictionary (This is changed from a list using in assignment04)
- **students: list:** list is set to and empty list

**Input / Output:**

- On menu choice 1, the program prompts the user to enter the student's first name and last name, followed by the course name, using the input() function and stores the inputs in the respective variables.

- Data collected for menu choice 1 is added to a dictionary named student_data. Next, student_data is added to the **students** two-dimensional list of dictionaries rows.
- On menu choice 2, the presents a string by formatting the collected data using the print() function.
- On menu choice 2, the program uses the print() function to show a string of comma-separated values for each row collected in the **students** variable.

**Processing**

- When the program starts, the contents of the "Enrollments.json" are automatically read into the **students** two-dimensional list of dictionary rows using the json.load() function. (**Tip:** Make sure to put some starting data into the file or you will get an error!)

**Figure 1. Ongoing implementation workflow**

In addition to the above, I also attempted to complete the tasks that go together as a group. For example, the error handling tasks all have similar frameworks, so I left those to be added separately from the execution tasks of the code. I also started this week's assignment with my previous week's assignment as I have in the past but decided quickly to pivot into using the starter.py provided instead. I thought the pseudocode provided was more useful than my comments previously.

## Labs Review

I normally review the labs within the span of the week and find moderate utility when it comes to preparation of the assignment. I do think this week was exceptionally similar from Labs to Assignment (Figure 2). Unknown if that was by design or not but thought I would bring it up as an anecdote.

```
30      #Below pulled straight from Lab02 for r op to .json
31      try:
32          file = open(FILE_NAME, 'r')
33          students=json.load(file)
34          file.close()
35      except Exception as e:
36          #I think this prompt is the minimum for most cases of errors
37          print('An error occurred while attempting to read the .json file')
38      finally:
39          if file.closed == False:
40              file.close()
```

**Figure 2. A near copy/paste job from Lab02**

## Exceptions

The use of error handling this week was my first introduction at tailoring the presentation layer when inputs do not meet the expected criteria. I am going to use the common error samples used by the instructors until further notice because I think I could spend an unlimited number of hours crafting other errors that are probably not relevant to most users. As I continue in this course I hope to find where the balance between verbose error handling and excessively worded code lies. For example, I commonly incorrectly space between character strings as in Figure 3 and it would add convenience to my life if I had an internal error handler that caught these.

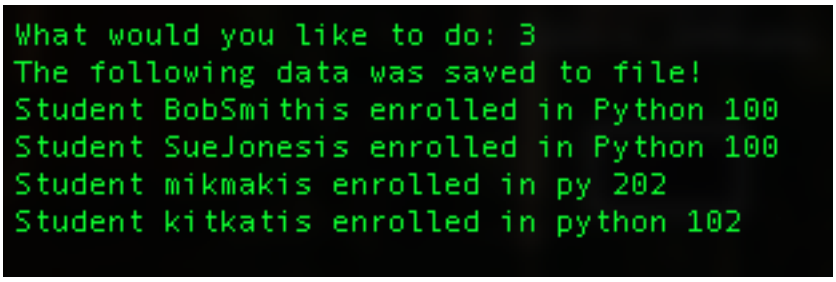



**Figure 3. Improper concatenation caught during test executions**

## Summary

As a trailing comment about GitHub, I use an internal repository server in my job that mirrors all the functions contained within, so I am moderately versed on the use and application. Otherwise, this week's content appropriately introduced JSON format and the utility to me. The overall definitions for the different data categories bleed together in my current understanding. I think the following weeks will serve as a useful reiteration of the differences. I also enjoyed the presentation or error handling and our responsibilities as code authors.