Name: Michael Keel, mkeel

Date Submitted: 05/28/25

Course: Foundations of Programming: Python

GitHub: https://github.com/mikeel4real/IntoToProg-Python-Mod06

Assignment 06

Introduction

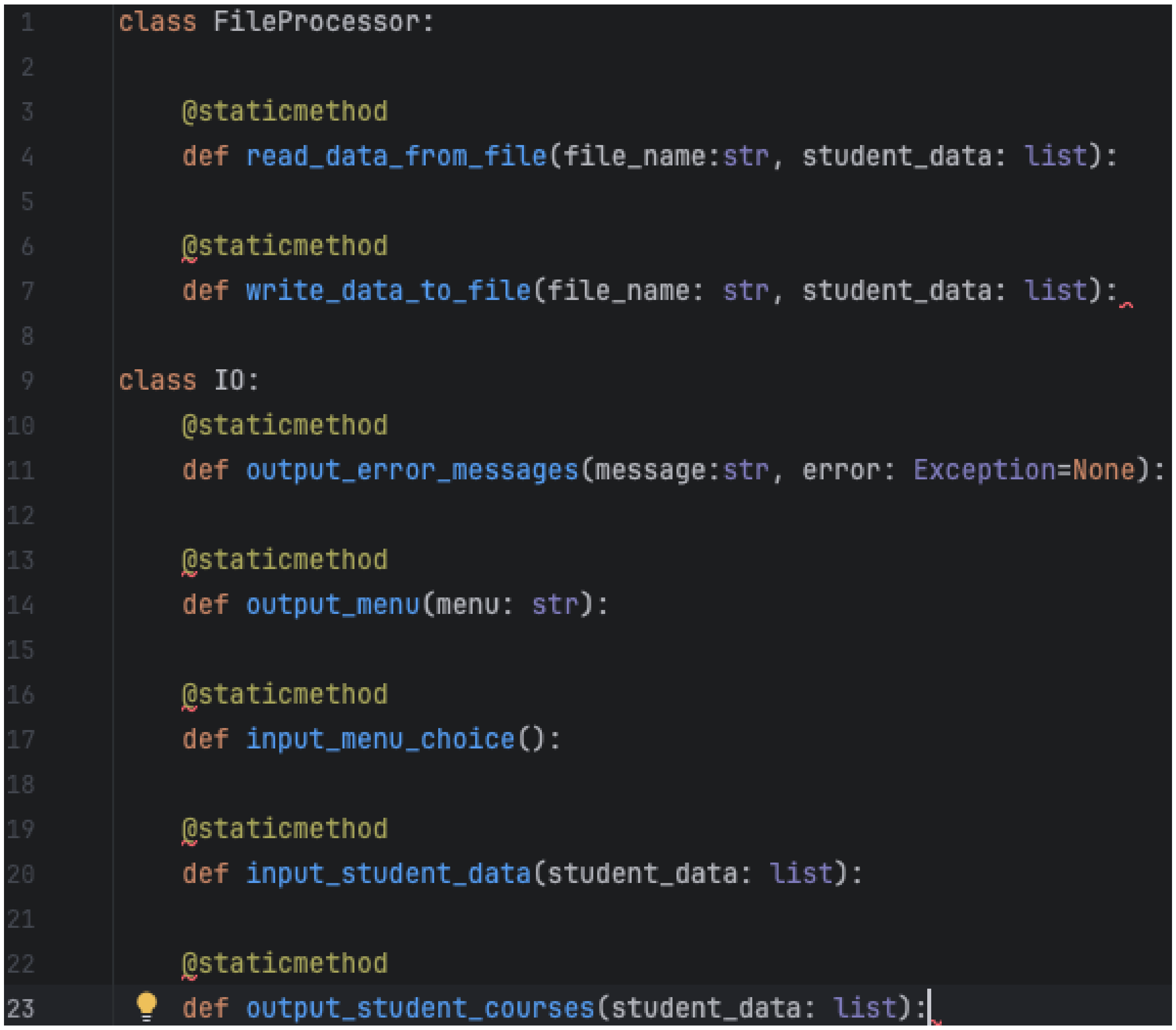
In this document I will review the events from Week 6 of Foundations of Programming: Python. I started this week with review of the Module Notes and Videos to clarify the specifics of how functions are incorporated into classes, global and local variables, and separations of concern. I found this week to be enlightening because we aimed to package the contents of Week 5 into better chunks.

Class Handling

After review of the Module Videos, I gravitated towards Anubhaw's video that covers *Separation of Concern*. Conveniently that was the last video I watched prior to looking at the assignment details for the week. I watched this video an additional two times to pick up on things that were unclear the first time. As a review, we learned that classes allow us to bundle similar functions and parameters together to make the overall program easier to understand and simpler to debug each step. In Assignment 06, we collected our functions from Assignment 05 together by the action they intend to perform. Each class grouped our functions into buckets of either data input/output or data processing.

Program Assembly

I followed the same method described in the Separations of Concern Video where we defined the class first and then conformed each step of the previous assignment into the appropriate bucket. An outline of the intended functions to be parsed can be seen in Figure 1. Once I had the outline in place, the bulk of the time spent on the Assignment was spent on adapting the previous functions into this class format. I did not foresee the number of typos I would accumulate during the translation of each method. I also lost sight of multiple global versus local variables along the way which delayed my code from working. The "main body" of the program is simple to write once all the cleanup from the classes is complete.

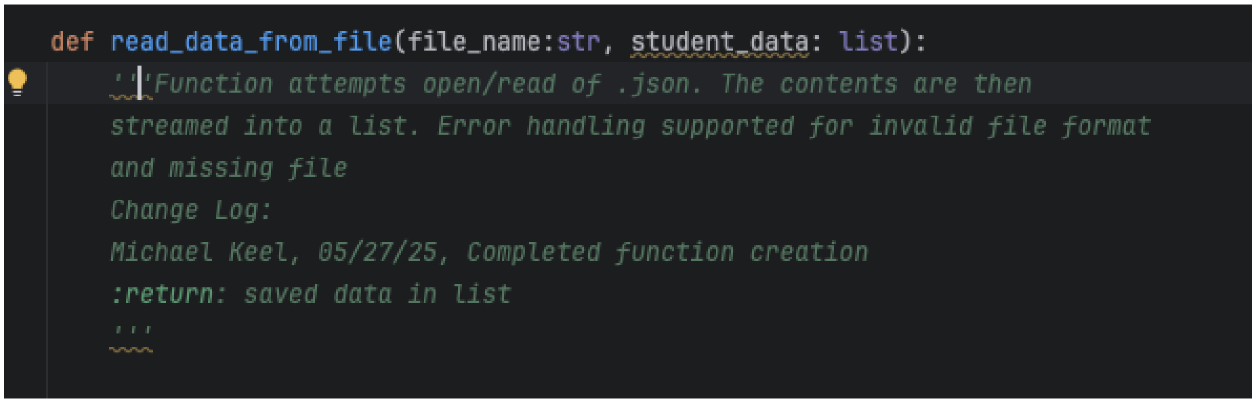


```python
class FileProcessor:

    @staticmethod
    def read_data_from_file(file_name:str, student_data: list):

    @staticmethod
    def write_data_to_file(file_name: str, student_data: list):

class IO:
    @staticmethod
    def output_error_messages(message:str, error: Exception=None):

    @staticmethod
    def output_menu(menu: str):

    @staticmethod
    def input_menu_choice():

    @staticmethod
    def input_student_data(student_data: list):

    @staticmethod
    def output_student_courses(student_data: list):
```

Figure 1. Class Outline of Functions

Document Strings

I did not prepare document strings initially for each entry of Change Log where I added a function or class. I did add these after office hours and review of peer's contributions in GitHub. I do see the added benefits of this going forward. Additionally, these additions highlighted for me the warning seen in Figure 2. PyCharm expects triple double-quoted strings for docstrings.

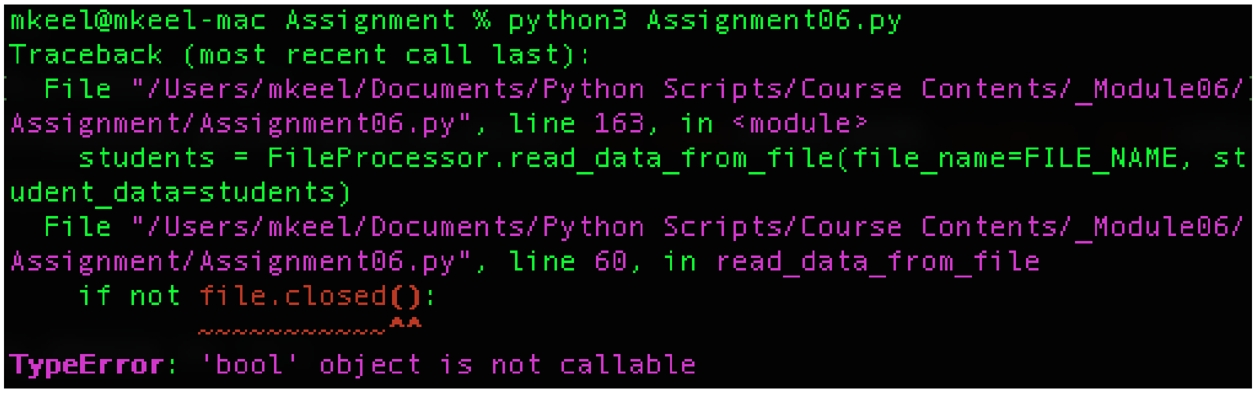


```
def read_data_from_file(file_name:str, student_data: list):
    '''Function attempts open/read of .json. The contents are then
    streamed into a list. Error handling supported for invalid file format
    and missing file
    Change Log:
    Michael Keel, 05/27/25, Completed function creation
    :return: saved data in list
    '''
```

Figure 2. Triple single-quotes display as a warning.

Testing

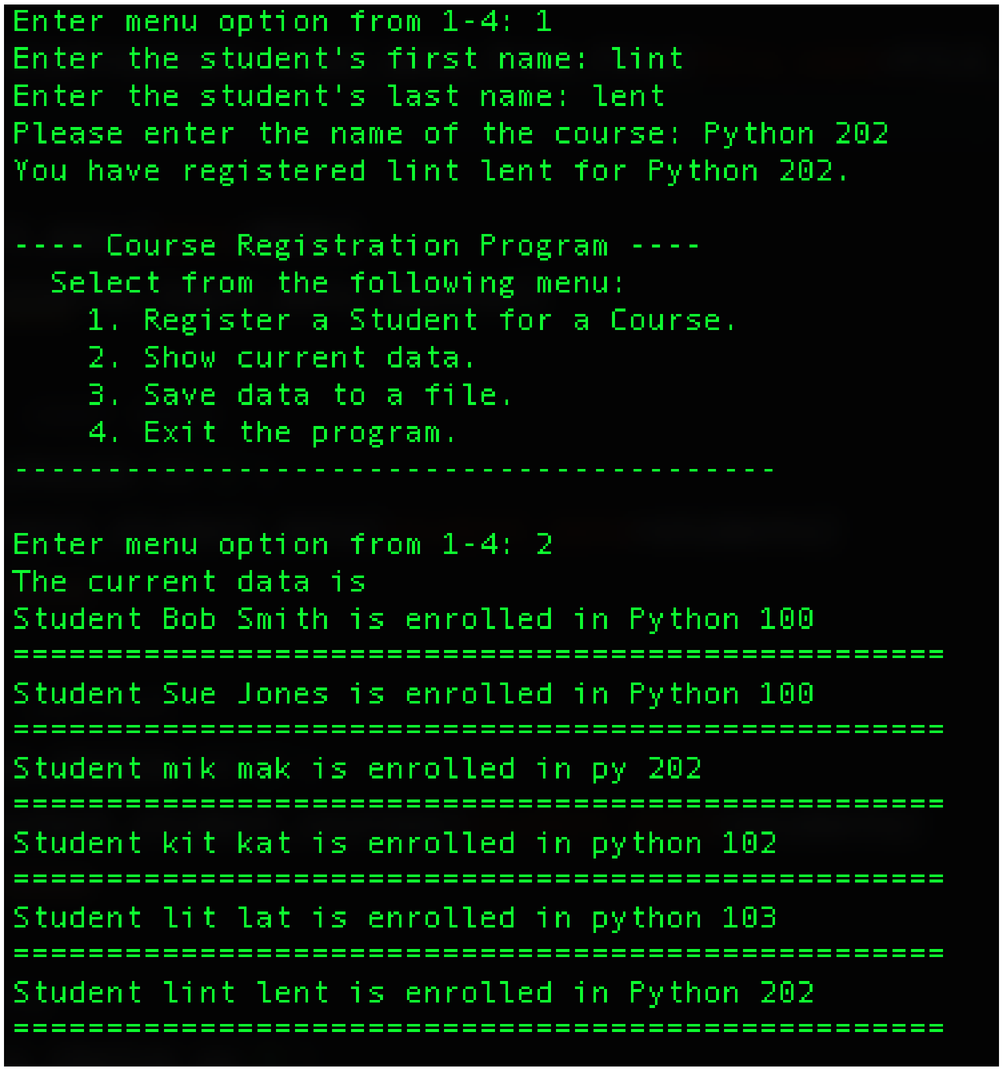
This week's testing easy enough to see through PyCharm when I elected variables that did not exist either globally or locally. Even once I had everything working in PyCharm I discovered more typos through my terminal when ran as seen in Figure 3.



```
mkeel@mkeel-mac Assignment % python3 Assignment06.py
Traceback (most recent call last):
  File "/Users/mkeel/Documents/Python Scripts/Course Contents/_Module06/
Assignment/Assignment06.py", line 163, in <module>
    students = FileProcessor.read_data_from_file(file_name=FILE_NAME, st
udent_data=students)
  File "/Users/mkeel/Documents/Python Scripts/Course Contents/_Module06/
Assignment/Assignment06.py", line 60, in read_data_from_file
    if not file.closed():
           ~~~~~~~~~~~^^
TypeError: 'bool' object is not callable
```

Figure 3. Typos experienced in testing.

These typos cleared when I removed incorrect text and reran. A clean run through menu options 1 and 2 can be seen in Figure 4.



```
Enter menu option from 1-4: 1
Enter the student's first name: lint
Enter the student's last name: lent
Please enter the name of the course: Python 202
You have registered lint lent for Python 202.

---- 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 menu option from 1-4: 2
The current data is
Student Bob Smith is enrolled in Python 100
==================================================
Student Sue Jones is enrolled in Python 100
==================================================
Student mik mak is enrolled in py 202
==================================================
Student kit kat is enrolled in python 102
==================================================
Student lit lat is enrolled in python 103
==================================================
Student lint lent is enrolled in Python 202
==================================================
```

Figure 4. Intended response from program through options 1 and 2.

In menu option 3, my new entry was correctly saved to “Enrollments.json” so I started to clean up my program to be more aesthetically pleasing to a third party or my own future use.

Summary

As an aside, I did find more utility this week in looking at my peer’s contributions to GitHub. I forgot multiple trailing colons and had many spelling errors that became obvious once I attempted to understand why a peer’s code worked and mine did not. Similarly to last week, this week was a convenient pivot of information we already possessed into a new method or packaging that increases utility for future tasks.