John Lewandowski

07/30/24

Foundations of Python

Assignment 05

https://github.com/lewandj1/IntroToProg-Python-Mod05.git

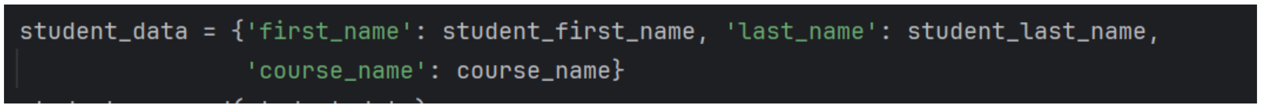
Advanced Collections and Error Handling

Introduction

This report documents some of the features I learned to use within Python regarding the fourth assignment in the Foundations of Python course at the University of Washington. Key features included were as follows: Using dictionaries with file data, Reading and writing to a JSON file, Error handling by custom errors, and uploading to Github. In each section I will briefly explain what the feature is and how I implemented the feature.

Using dictionaries with file data

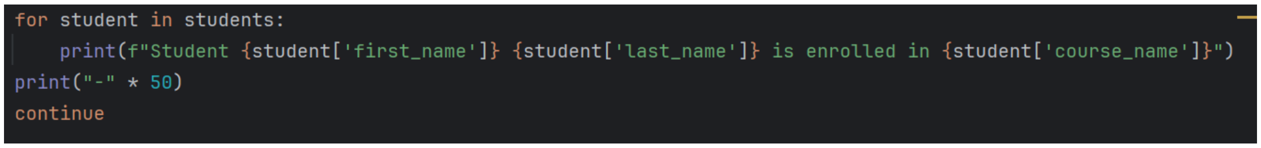
For this assignment the primary use of data was with dictionaries. As such, the following keys were used to properly call the data once extracted from the JSON file or input by the user. Shown in **Figure 1**, you can see the keys used are: first_name, last_name, and course_name.



```
student_data = {'first_name': student_first_name, 'last_name': student_last_name,
                'course_name': course_name}
```

Figure 1: Example of dictionary use.

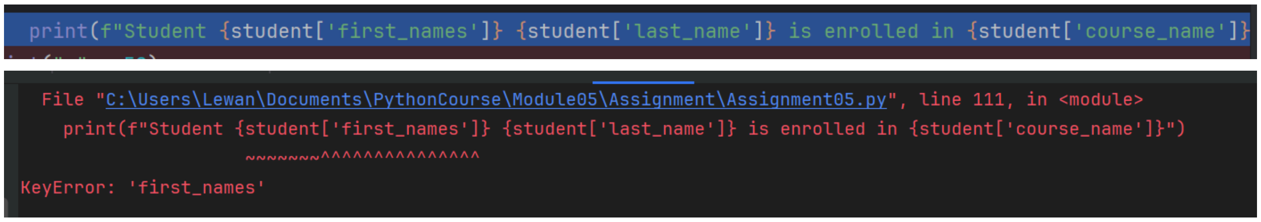
In this case the student data was loaded from the JSON file (covered in next section) and converted into a list of 'students' that will later be used for adding to the JSON file. Now that the keys have been established they can now be used for individual callouts during printing, shown in **Figure 2**.



```
for student in students:
    print(f"Student {student['first_name']} {student['last_name']} is enrolled in {student['course_name']}")
print("-" * 50)
continue
```

Figure 2: Example of Calling values using keys.

Most notable is how the value is no longer being called out in comparison to using lists, where the location is being called. This greatly simplifies the process of grabbing data so long as the key is properly called. Any small difference when calling the key will generate an error (W3schools.com).



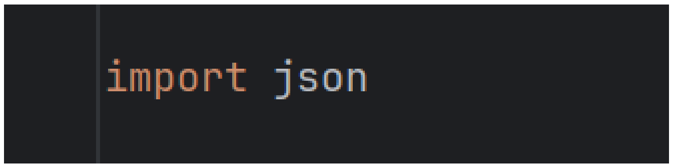
```
print(f"Student {student['first_names']} {student['last_name']} is enrolled in {student['course_name']}
```

```
  File "C:\Users\Lewan\Documents\PythonCourse\Module05\Assignment\Assignment05.py", line 111, in <module>
    print(f"Student {student['first_names']} {student['last_name']} is enrolled in {student['course_name']}")
                     ~~~~~~~^^^^^^^^^^^^^^^
KeyError: 'first_names'
```

Figure 3: Example of using the wrong key value and the error generated in response.

Reading and Writing to a JSON file

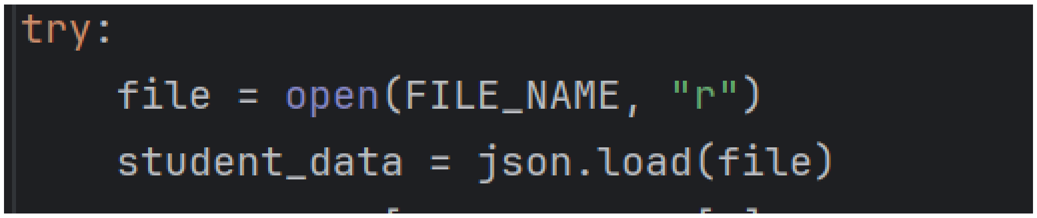
To use the JSON file we must first import the JSON package through a built-in package called JSON (GeeksforGeeks.org, 2022). See **Figure 4** for the method I used in this assignment.



```
import json
```

Figure 4: Importing JSON package.

Now that the package is ready to be used we first want to pull our information from the JSON file on hand. The function I chose to use is json.load(), this takes the JSON file and loads it into a dictionary (GeeksforGeeks.org, 2022).

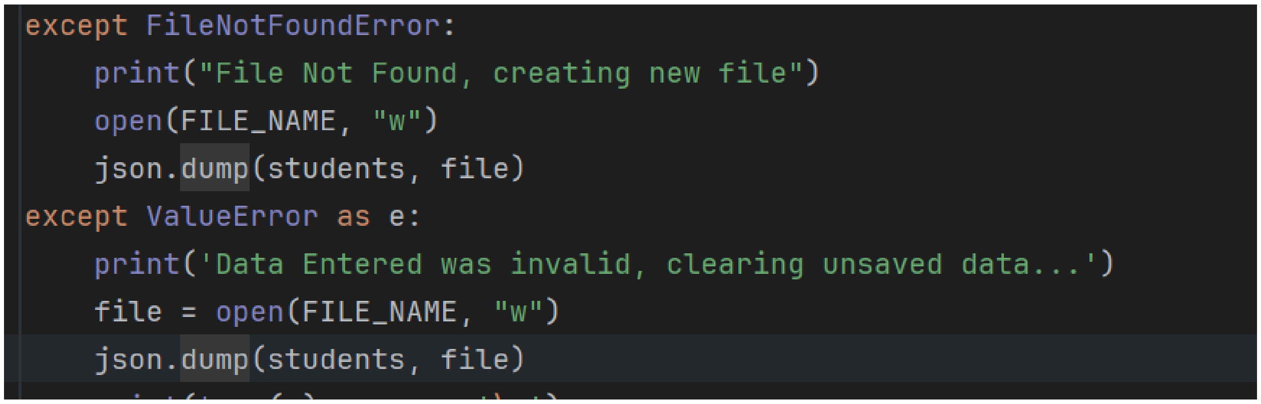


```
try:
    file = open(FILE_NAME, "r")
    student_data = json.load(file)
```

Figure 5: Loading JSON file to a dictionary 'student_data'

At this point that is about all there is to it for reading from a JSON file.

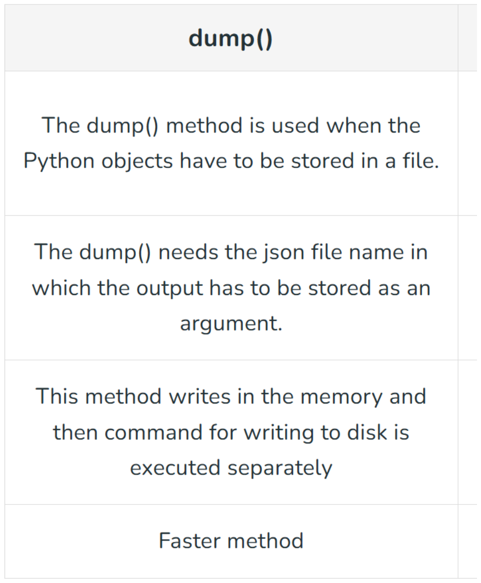
Similarly, when writing to a JSON file we need to first set the permission for writing and then push the data to the file. For this use-case the json.dump() function was used, shown in **Figure 6**.



```
except FileNotFoundError:
    print("File Not Found, creating new file")
    open(FILE_NAME, "w")
    json.dump(students, file)
except ValueError as e:
    print('Data Entered was invalid, clearing unsaved data...')
    file = open(FILE_NAME, "w")
    json.dump(students, file)
```

Figure 6: Use case of json.dump() function.

The dump function takes the memory in the file and pushes it to the file that's callout (GeeksforGeeks, 2024).



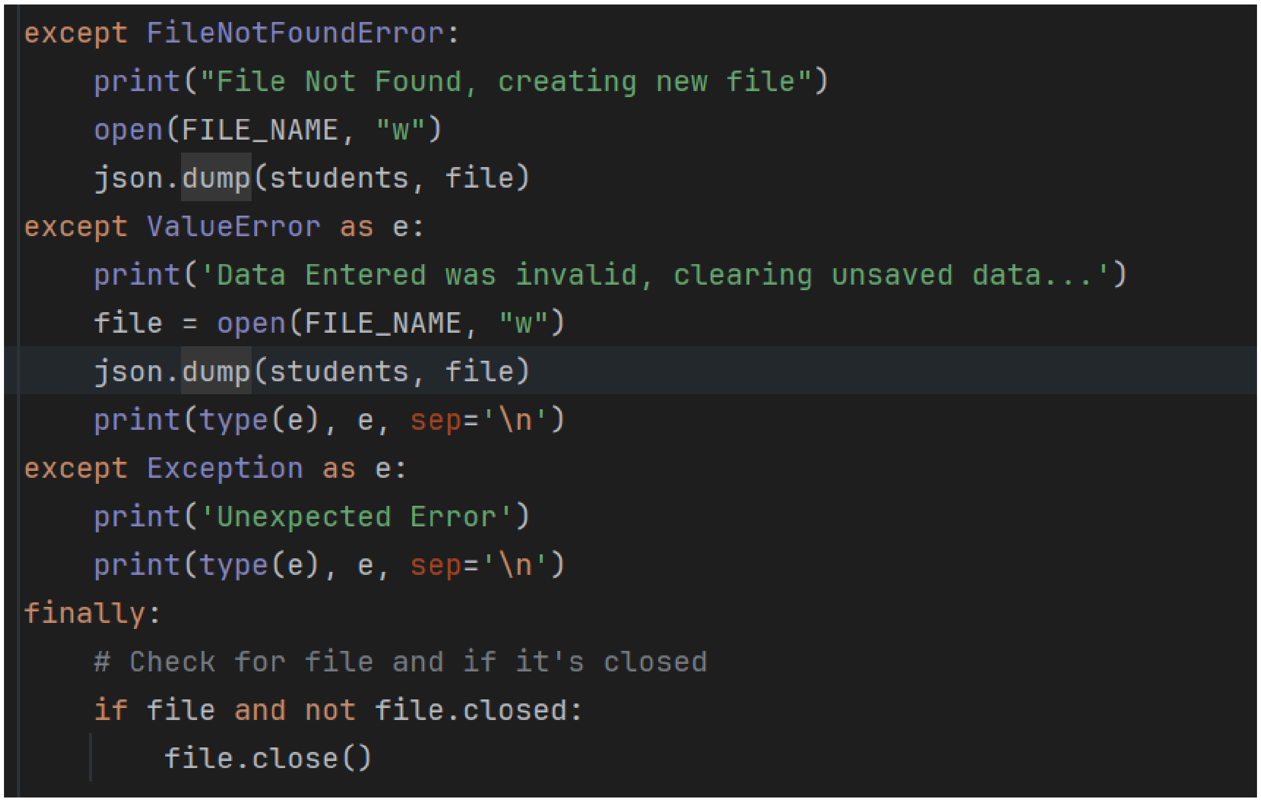
dump()
The dump() method is used when the Python objects have to be stored in a file.
The dump() needs the json file name in which the output has to be stored as an argument.
This method writes in the memory and then command for writing to disk is executed separately
Faster method

Figure 7: Definition of Dump() function for JSON files. (GeeksforGeeks, 2024)

Error handling by custom errors

In this assignment I used error handling in three different sections, each with similar uses, but for different reasons.

The first case was for initializing the file. The two main concerns here was whether the file I am calling data from exists and secondly whether the data that is being input is compatible. Shown in **Figure 8**, you can see for the file existing problem FileNotFoundError was used and for Invalid data the ValueError callout was used.

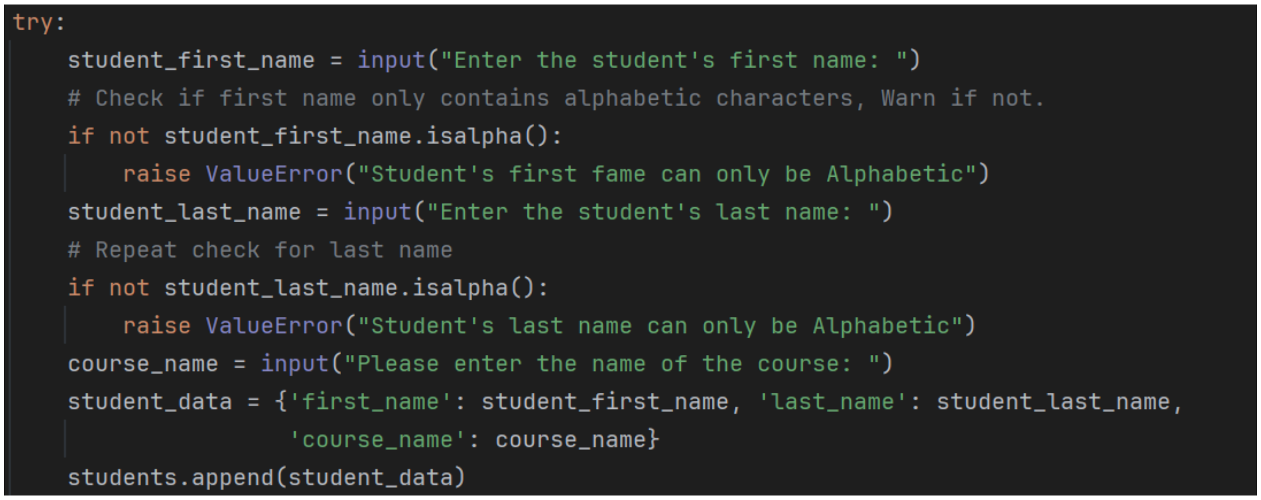


```python
except FileNotFoundError:
    print("File Not Found, creating new file")
    open(FILE_NAME, "w")
    json.dump(students, file)
except ValueError as e:
    print('Data Entered was invalid, clearing unsaved data...')
    file = open(FILE_NAME, "w")
    json.dump(students, file)
    print(type(e), e, sep='\n')
except Exception as e:
    print('Unexpected Error')
    print(type(e), e, sep='\n')
finally:
    # Check for file and if it's closed
    if file and not file.closed:
        file.close()
```

Figure 8: Examples of Error Handling for First case.

It is important to understand that when an error in encountered, "Python will raise an exception error. This exception error will crash the program if you don't handle it. In the except clause, you can determine how your program should respond to exceptions." (Python, R 2024). The Try-Except-Finally method was what I chose to use as there were multiple exceptions that were possible. In each case of an error, the except stores the value as 'e' then a print function is used to display the error, I often added a custom text to accommodate the error to help the user understand the cause.

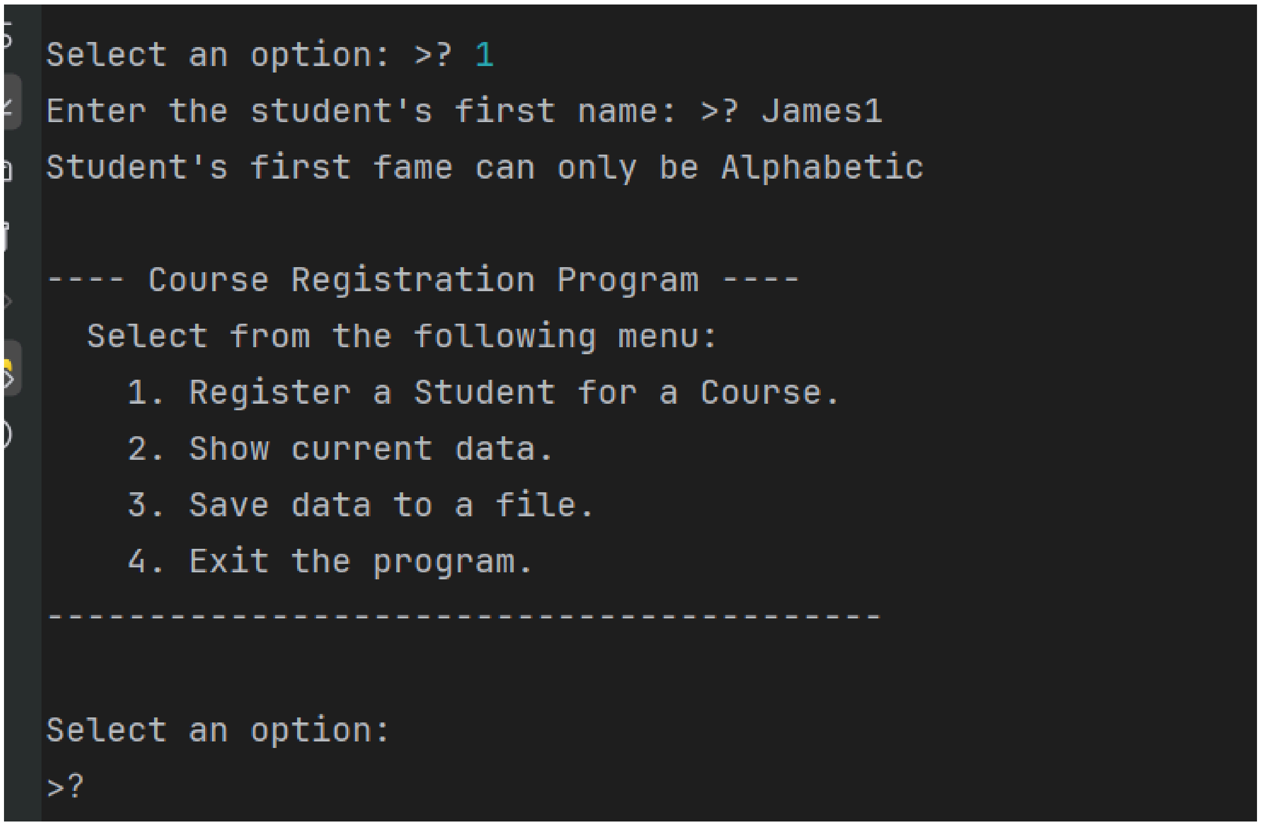
The second use of Error exceptions was for the user input data. Referencing Python, R and the notes for this assignment, an If-Raise-If not-raise format was developed. For this case, see **Figure 9** for the associated code.



```python
try:
    student_first_name = input("Enter the student's first name: ")
    # Check if first name only contains alphabetic characters, Warn if not.
    if not student_first_name.isalpha():
        raise ValueError("Student's first fame can only be Alphabetic")
    student_last_name = input("Enter the student's last name: ")
    # Repeat check for last name
    if not student_last_name.isalpha():
        raise ValueError("Student's last name can only be Alphabetic")
    course_name = input("Please enter the name of the course: ")
    student_data = {'first_name': student_first_name, 'last_name': student_last_name,
                    'course_name': course_name}
    students.append(student_data)
```

Figure 9: Example of If-raise- If not- raise error handling.

As you can see, the code is now checking the user input data and verifying whether the values input match what is expected. In this case I was concerned with the data being Alphabetic only. Referencing the notes, isalpha() seemed the logical choice. Upon an error, the program will raise the custom statement and then be prompted to make their new choice, as seen in **Figure 10**.



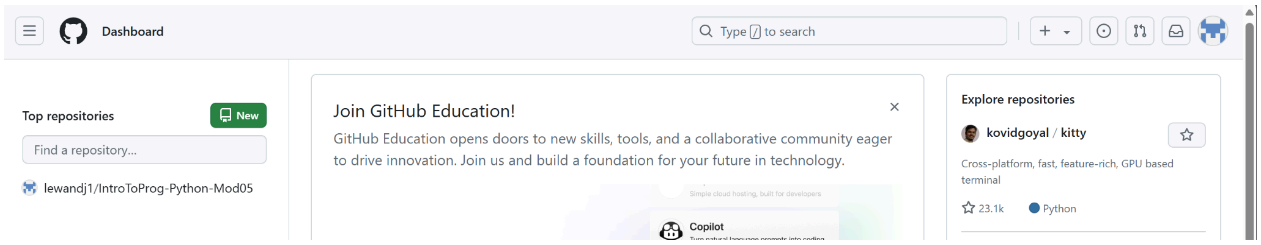
```
Select an option: >? 1
Enter the student's first name: >? James1
Student's first fame can only be Alphabetic

---- 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.
-----------------------------------------


Select an option:
>?
```

Figure 10: Example of Error message for input not being Alphabetic.

The third use for Error handling was when trying to save data to the file. The concern was whether the data is formatted correctly and whether the values are acceptable. Similar to the first case, the Try-Except-Finally method was decided. From this method the only difference was the use of 'TypeError'. This exception checks whether the value type of the data is in the expected format.

Uploading to Github

Lastly, we have the code completed and need a way to share our code so that we may receive critics and valuable feedback on the format, logic, error handling, etc. Once established on the website: Github.com, create an account and you should see a similar screen as the below **Figure 11**.



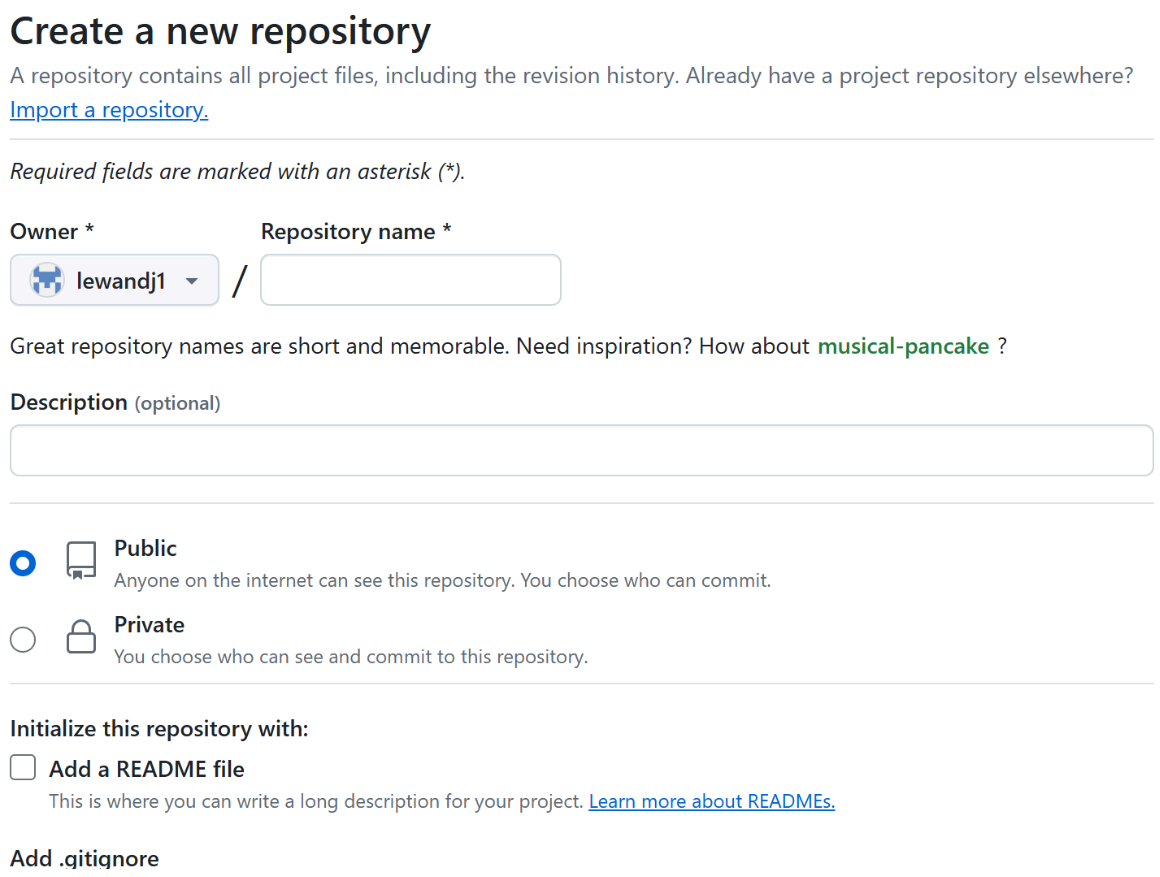
Join GitHub Education!

GitHub Education opens doors to new skills, tools, and a collaborative community eager to drive innovation. Join us and build a foundation for your future in technology.

Cross-platform, fast, feature-rich, GPU based terminal

Figure 11: Github homepage (Github.com, 2024)

From here we need to create a repository to establish a log, title, and create a location link.



Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? Import a repository.

Required fields are marked with an asterisk (*).

Owner * lewandj1 / Repository name *

Great repository names are short and memorable. Need inspiration? How about musical-pancake ?

Description (optional)

(●) Public
Anyone on the internet can see this repository. You choose who can commit.

( ) Private
You choose who can see and commit to this repository.

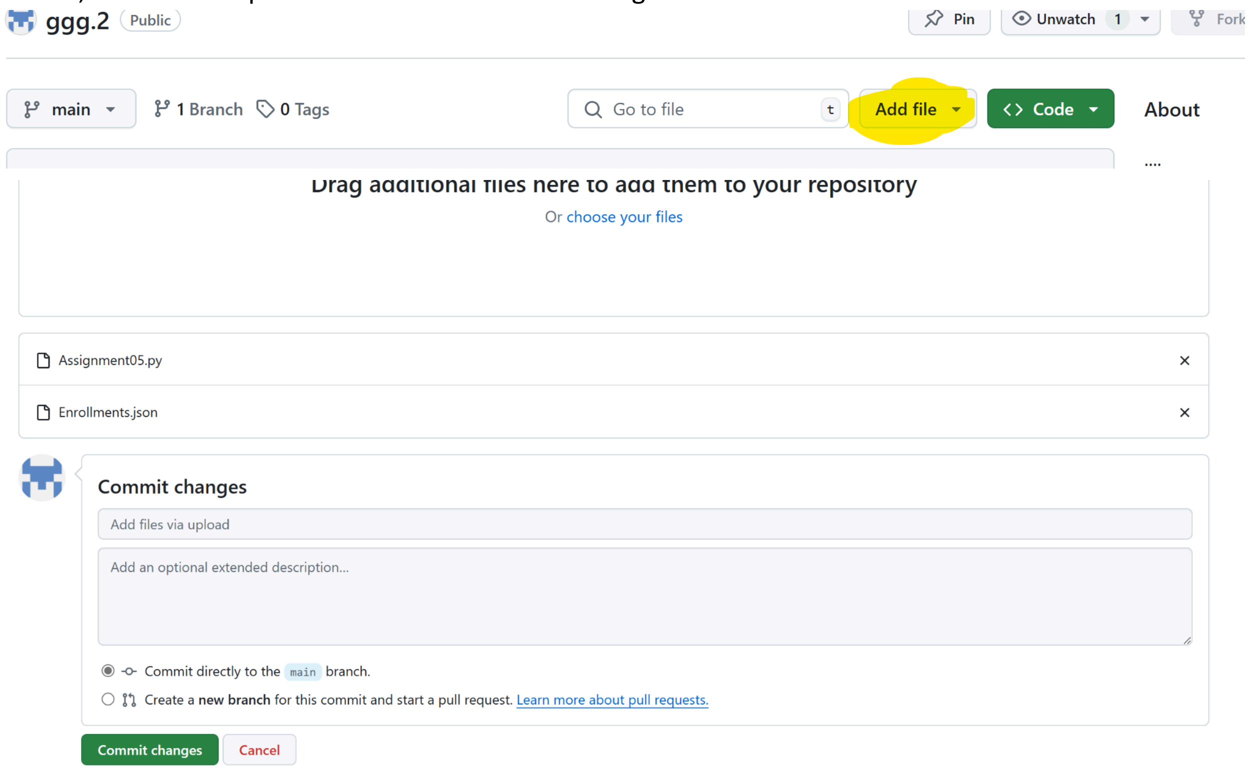
Initialize this repository with:

☐ Add a README file
This is where you can write a long description for your project. Learn more about READMEs.

Add .gitignore

Figure 12: Creating a repository (Github.com, 2024)

At this point we need a name, description, and make the choice to keep the file public or private. Once created, we need to upload the files desired for sharing.



Drag additional files here to add them to your repository

Or choose your files

Assignment05.py

Enrollments.json

Commit changes

Add files via upload

Add an optional extended description...

Commit directly to the main branch.

Create a **new branch** for this commit and start a pull request. Learn more about pull requests.

Figure 13: Adding a file to the repository (Github.com, 2024)

At this point the last step is to use text in the 'Commit changes' section to add to the log for the repository. To note, if the same file name is uploaded repeatedly, Github will write the new version assuming the latest upload is current. In doing this, the Commit changes acts as our revision control and will now display next to the title of the file and the original will be sent to the readme.txt indicating what the changes listed were.

Summary

Following the Module 05 videos, reading/watching the additional videos and modules in tasks 2 and 3 of the assignment file and referencing helpful websites on Python usage, I was able to successfully complete the assignment. I learned how to take a JSON file and decode or write information to it. I also learned the importance of keys for dictionaries and what can go wrong when the wrong key is called out. Additionally, I now have a better understanding of the errors that arise with the use of JSON files and how I can handle the errors as they come and prevent my code from crashing. I also learned the importance of having the foresight to see certain errors and account for them with multiple methods. Finally, I learned how to upload files to Github and to establish and account for revisions, thereby getting a better handle on how my code changed over the course of multiple iterations.These skills greatly increase my ability to develop as a coder and progress through my journey as more advanced techniques are brought to my attention via the next assignment.

References

- *W3Schools.com*. (n.d.). https://www.w3schools.com/python/python_dictionaries.asp
- GeeksforGeeks. (2022, July 19). *Reading and writing JSON to a file in Python*. GeeksforGeeks. https://www.geeksforgeeks.org/reading-and-writing-json-to-a-file-in-python/
- GeeksforGeeks. (2024, June 20). *Json.dump() in Python*. GeeksforGeeks. https://www.geeksforgeeks.org/json-dump-in-python/
- Python, R. (2024, January 29). *Python Exceptions: An Introduction*. https://realpython.com/python-exceptions/
- *GitHub: Let's build from here*. (2024). GitHub. https://github.com/