Andrea Martinez

November 13, 2024

Python 100

Assignment 05

Assignment 05 – Programming with Python Dictionaries

Introduction

In module 05 of this course, one of the main objectives was to use dictionaries. In this document, I'll briefly describe my initial thoughts about using dictionaries as well as the differences from using lists. In addition, I'll describe the steps taken to manage errors within a script and how the try, except, and finally blocks work. I'll also demonstrate the differences between using .csv files and .json files, and how .json files can be a more powerful file specially for unstructured data. Finally, I will present my impressions from using version-control data in GitHub.

Reading a File

Starting this assignment was very straightforward because it used the code that I wrote during assignment 04 as the foundation. This was true because there were only some minor differences between these two assignments such as the addition of dictionaries, try-except conditions, and writing to a .json file.

Before the initialization of the variables and constants of the script, I made sure that I imported the dependencies needed for my code such as the TextIO Module which allows to read and write to a file, and the JSON module, which allows to work with JSON data. After this, I proceeded to open the .json file defined under constant FILE_NAME in read mode under a try-block to check for any errors. I opened it in read mode because the .json file already existed and contained data in the same folder structure. In the try-block, I added a for loop to read each line, and saved the data in a list of dictionaries under the variable "student_data". I then appended the data and closed the .json file. I learned that the convention to read a file differs between file types. For example, to open a csv file, I would need to use a for-loop to parse through the data and use the readlines() function. On the other hand, to read all the data in a .json file, all I had to use was json.load() because the data was already saved in dictionaries.

Error Handling while Reading a File

A new concept introduced in this module was error handling through the use of the try-except blocks. Like mentioned in DEV Community: "it is good practice to use try-except blocks because it helps identify exactly which section of the code might be causing an exception, making debugging easier and ensuring that the developer does not inadvertently catch exceptions that weren't initially intended to handle." (https://dev.to/enter?state=new-user&bb=126495, 2024). In the try-except block, I added a few errors to flag the user with issues while reading the data in the JSON file. The first except block I added was if there was not an existing file in the file structure, and it used the 'FileNotFoundError' error. Then, to satisfy the homework requirements, I also added the IndexError, which was used to show the user that the data in the file did not contain the expected three columns or answers. Within that IndexError except block, I added a while loop to give the user a chance to enter the correct data in the file again without crashing the code. I also added an Exception in case there was an unknown error in the code, and instead of crashing the code, the console would show the error and still continue running the code.

Saving Data in Dictionaries

The first thing I did inside of the big while loop that showed the menu was displaying the options with the print() function, prompting the user to make a selection. I stored the user selection under the variable 'menu_choice' as a string to be later used in the if-statement. After the user made a selection, I nested an if-statement inside of the while loop with a series of choices. If choice #1 was selected from the menu, I asked the user to register a student for a class with three input functions learned in module 01. I then made a list with the information entered by the user, and subsequently appended the values to a list called 'student'. I chose to append that data with the append() function in case the next user selection was also a '1', such that I had all the registered students saved under the same list of dictionaries. Under option '1', I also added a try-except conditional to catch any errors in case the entered data was incorrect. For example, if the data entered contained special characters other than letters , it would show a message to enter it again.

If menu choice #2 was selected, I displayed the current students registered for the class. So, for example, if I selected choice #1 three different times to register three different students, their names and classes would get printed with choice #2. For this purpose, I had to create a for loop such that the code would individually parse through each row in the list and print all the rows of students.

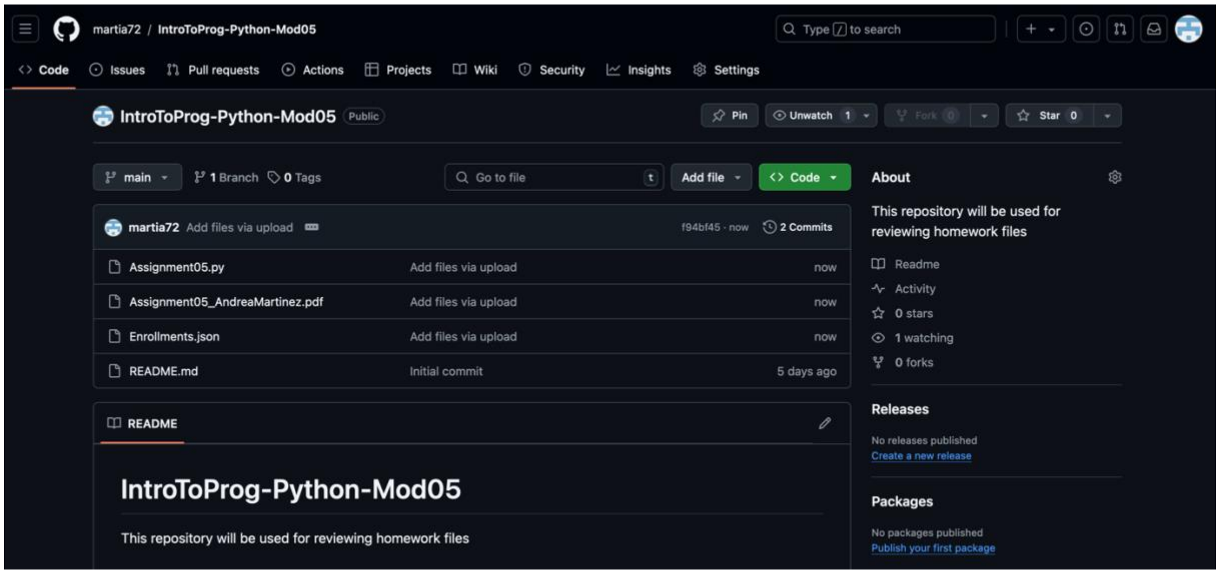
If menu choice #3 was selected, I opened the .json file in write mode and stored all the current students registered for the class into the Enrollments.json file using the json.dump(). To do this step, I implemented a for loop that would individually add all the rows available in the 'students' list until I closed the file. A small printed statement would also get output with choice #3, informing the user that the data is being saved to a .json file. If selection #4 was made, no other operation would be made, and I would just end the program with the break command. Now, even though, this was not a part of the script requirement, I added an "else" statement for any other answer that was not either one of the choices in the menu (1-4). This else statement printed an error message, indicating the selection was not valid and a valid answer must be entered again.

Testing the Program

As a developer, I had to test my code to make sure that it was doing what it was supposed to by meeting all the requirements set by the instructor. So, while I was testing the code, I made sure that I tested out the errors that I created in the try-except blocks. I liked adding different error types inside of the try-except blocks in particular because it allowed me to evaluate each possible error that could cause the code to crash or perform not as designed. After making sure that the code met all the requirements from the instructor, I tested the code in the terminal to ensure that the program would also run without the IDE.

Source Control in GitHub

I really enjoyed learning about GitHub because it was very straightforward to use and very user friendly. I liked the option of adding comments to a file commit because it helped me as the developer know in which version of the file I was on. I also liked how I can make a repository private or public in the server depending on the contents inside of the repository. I can now see how this tool is used for collaborative efforts between organizations because it lets them work on the files without the need of a local network drive that is dangerous for saving code since it overwrites changes. Figure 1 shows how my repository looks for Assignment 05.



IntroToProg-Python-Mod05 Public

martia72 Add files via upload f94bf45 · now 2 Commits

Assignment05.py	Add files via upload	now
Assignment05_AndreaMartinez.pdf	Add files via upload	now
Enrollments.json	Add files via upload	now
README.md	Initial commit	5 days ago

About

This repository will be used for reviewing homework files

README

IntroToProg-Python-Mod05

This repository will be used for reviewing homework files

Figure 1: GitHub Repository for Module 05 Assignment

Summary

In conclusion, the planning, writing, and execution of this code taught me how to properly use dictionaries, which is a very powerful way to index data based on their keys. I learned how to use JSON files, which are very helpful when the data is unstructured, unlike a CSV file which requires data to be uniformly distributed. I also practiced the try-except method, which allowed my code to continue running even though there was an error in a certain area of my code. However, the challenge is that my code must be resilient such that it avoids encountering new errors from the code that failed in the try block. I improved my skills writing for-loops, while-loops, and if-statements. I enjoyed learning the version control aspect programming through the use of GitHub. I can now say that I feel pretty comfortable writing a program that interacts with the user and performs different operations inside and outside of the Pycharm IDE. I continued to apply good practices in coding such as using Booleans, adding comments for documentation, and creating multi-line commands, and formatting strings. Finally, I look forward to continue using GitHub for future codes such that I can keep track of changes.