Assignment05_David_Levinson

David Levinson
8/27/2024
Foundations of Programming: Python
Assignment 05

https://github.com/dl-11/IntroToProg-Python.git

Introduction

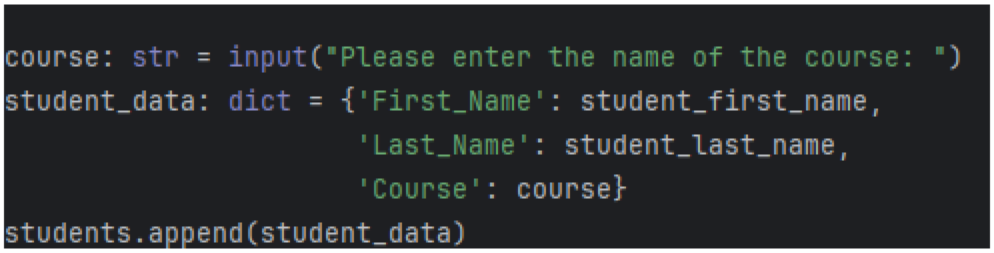
This week my family went to Disneyland, so although there weren't class challenges, I felt behind schedule in watching the course content and writing code. The previous few weeks had allowed me to establish a routine of watching the course materials in the first few days and then work on code over the weekend. Because of the vacation, it compressed my schedule.

From a class perspective, we learned about Exception handling, Github, Dictionaries and Files, and lastly JSON files. I spent a lot of time thinking about lists and dictionaries and trying to think about use cases for the various types. With that, I jumped into writing code.

Dictionaries

This week's starter file got us going from last week's delivery point. We were introduce to key value pairs and dictionaries. These are a robust way to store data and retrieve it.

I started working on the code refactoring from using lists to dictionaries first. I kept the input statements that captured the student's name and course. Instead of adding them to a list, this week we moved them over to a dictionary as shown in Figure 1.

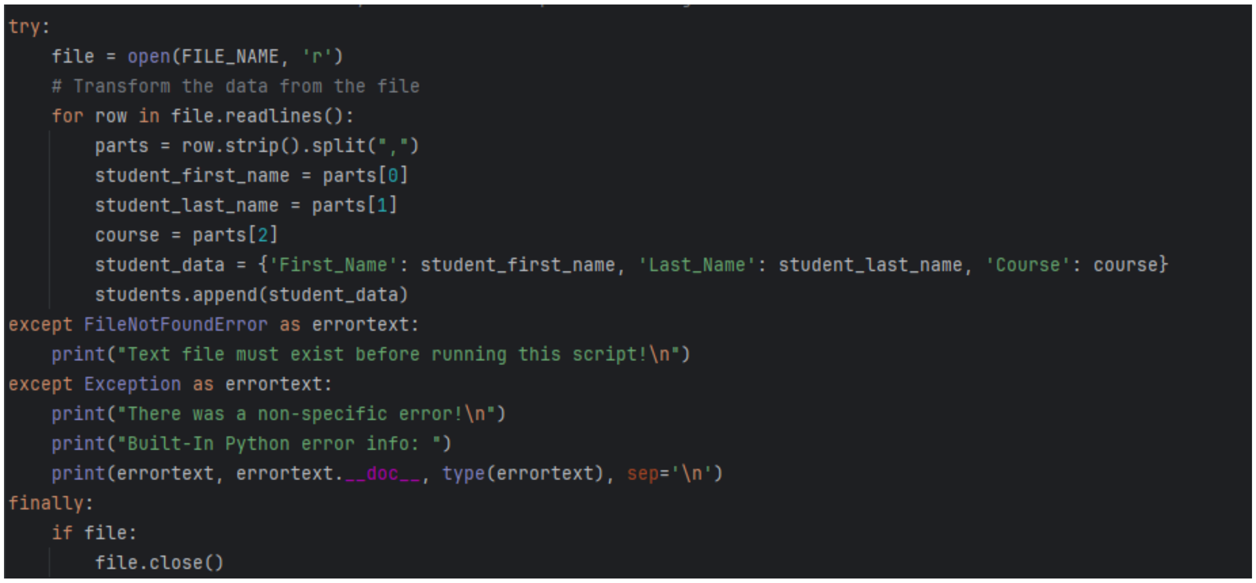


```python
course: str = input("Please enter the name of the course: ")
student_data: dict = {'First_Name': student_first_name,
                      'Last_Name': student_last_name,
                      'Course': course}
students.append(student_data)
```

Figure 1: Assigning the variables that were input to a dictionary

Exception Handling

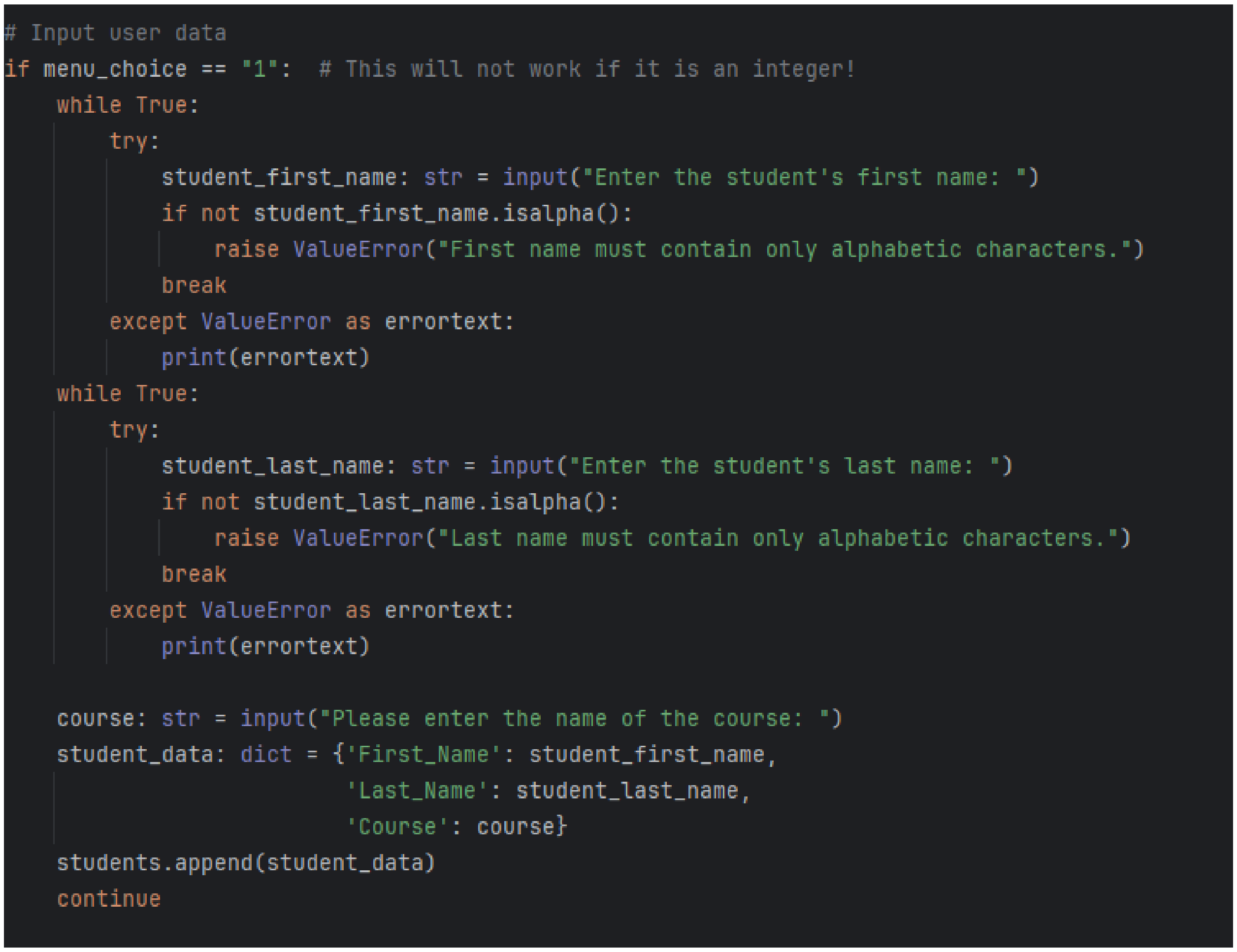
We brought in some exception handling techniques to improve the user experience. Instead of the program crashing, it gives more guidance on any problems it encounters. We learned how to handle file exceptions in the demo and I incorporated those learnings as we read the csv file into memory. This is shown in Figure 2 with the new try and except statements. In office hours, our instructor Luis Conejo Alpizar helped me improve the code to check if the file existed and needed to be closed.



```python
try:
    file = open(FILE_NAME, 'r')
    # Transform the data from the file
    for row in file.readlines():
        parts = row.strip().split(",")
        student_first_name = parts[0]
        student_last_name = parts[1]
        course = parts[2]
        student_data = {'First_Name': student_first_name, 'Last_Name': student_last_name, 'Course': course}
        students.append(student_data)
except FileNotFoundError as errortext:
    print("Text file must exist before running this script!\n")
except Exception as errortext:
    print("There was a non-specific error!\n")
    print("Built-In Python error info: ")
    print(errortext, errortext.__doc__, type(errortext), sep='\n')
finally:
    if file:
        file.close()
```

Figure 2: New try and except statements added to the file reading section of code

Next up for exception handling was to add statements to check if the user had input anything besides alpha characters into the student first or last name fields. This is valuable to learn as I believe users can frequently enter information incorrectly. I wanted to re-prompt if the user entered information incorrectly, so the additional while True statement provided a mechanism to do so. This was in one of my Python books and proved valuable here. Here it is in figure 3.



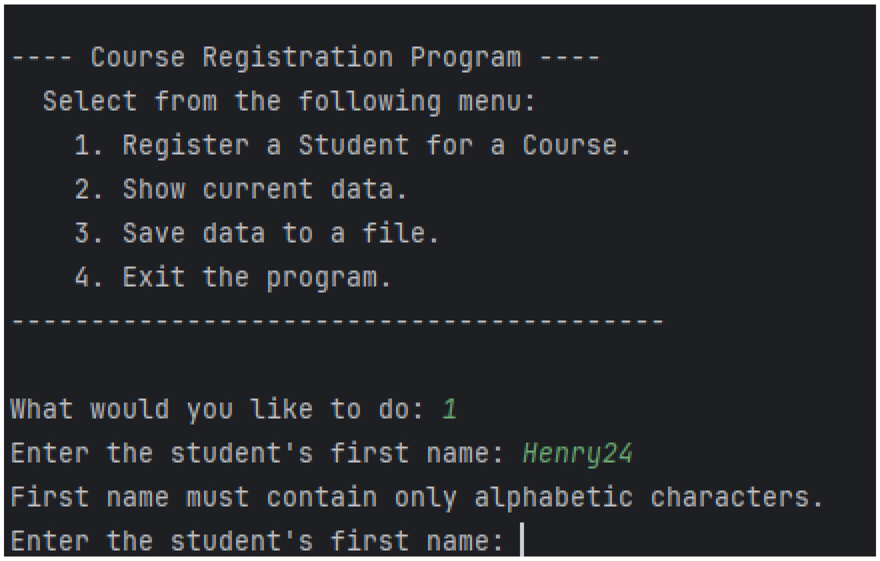
```python
# Input user data
if menu_choice == "1":  # This will not work if it is an integer!
    while True:
        try:
            student_first_name: str = input("Enter the student's first name: ")
            if not student_first_name.isalpha():
                raise ValueError("First name must contain only alphabetic characters.")
            break
        except ValueError as errortext:
            print(errortext)
    while True:
        try:
            student_last_name: str = input("Enter the student's last name: ")
            if not student_last_name.isalpha():
                raise ValueError("Last name must contain only alphabetic characters.")
            break
        except ValueError as errortext:
            print(errortext)

    course: str = input("Please enter the name of the course: ")
    student_data: dict = {'First_Name': student_first_name,
                          'Last_Name': student_last_name,
                          'Course': course}
    students.append(student_data)
    continue
```

Figure 3: Further exception handling for incorrect input in the Student first and last names.

Testing!

This week I performed lots of testing to make sure the code functioned as expected. This included entering purposely incorrect data to ensure the exception handling worked properly. An example of this is shown in figure 4 below. I entered

Henry24 and received feedback about the acceptable characters, then was re-prompted to enter it.



```
---- 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.
-----------------------------------------

What would you like to do: 1
Enter the student's first name: Henry24
First name must contain only alphabetic characters.
Enter the student's first name: 
```

Figure 4: Testing the exception handling for the input statements

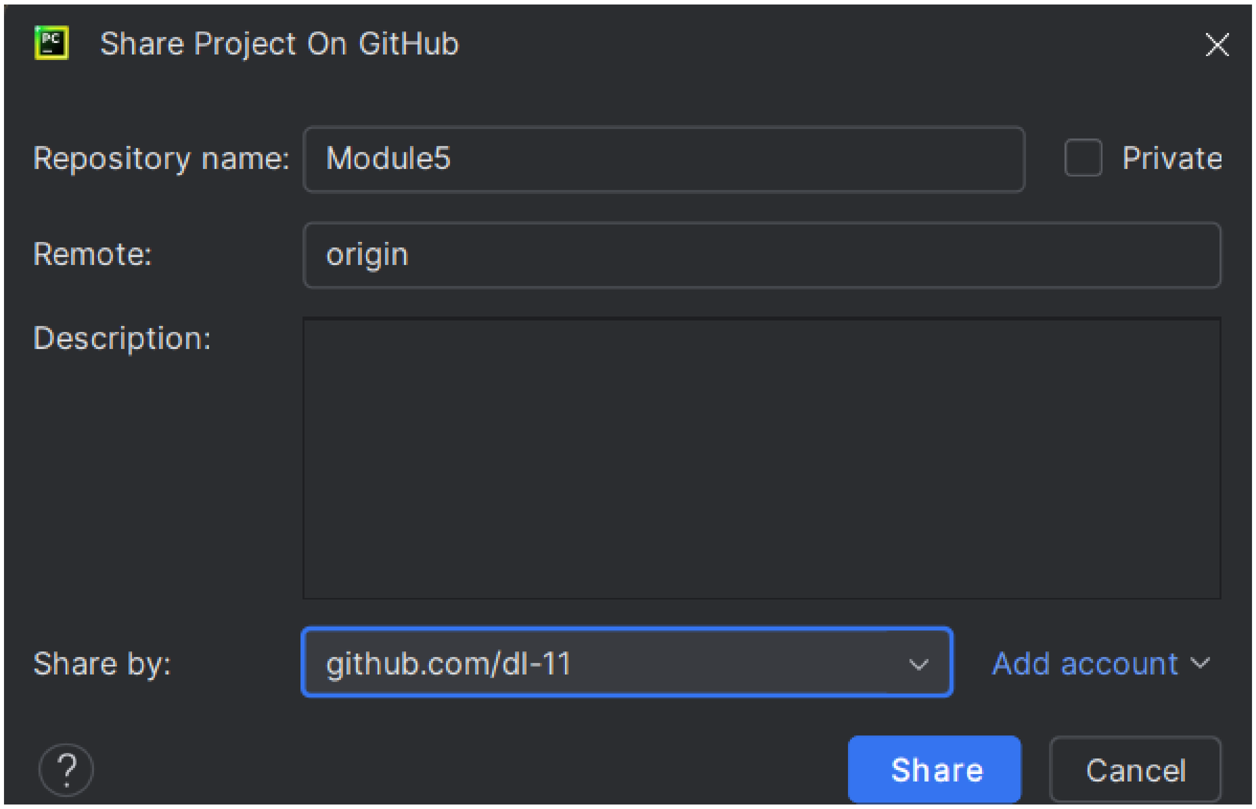
Week 4 Feedback incorporation

This week our instructor's assistant, Kelly Kauffman, provided guidance on the intent for option 3 to write the file using the csv_data variable. I had simply printed the student_data list variable and the output format wasn't as readable as it could have been. This week, I corrected that mistake.

I also had used one of the main variables in a for loop and converted that to a temporary variable name reflecting the use of it for rows.

Github

Knowing we would have to share the project on Github I started a new repository to learn how to use it. I started in Pycharm and connected my Github account to it. Next, I asked Pycharm to create a new repository for this as shown in figure 5.



Repository name: Module5

Private: [ ]

Remote: origin

Description:

Share by: github.com/dl-11

Figure 5: New repository creation in Pycharm

I uploaded the knowledge document and re-named the repository using the website, instead of through Pycharm.

Summary

In summary, this week's new learnings were valuable for better user experience with our application and learning about code management through Git. Through my current role at work, we use Git extensively and it's great to learn more about it firsthand. The addition of dictionaries is also valuable as a new data storage technique.