Name: Ronnie Sokha

Date: 11/18/23

Course: Foundations of Python Programming

GitHub Link: https://github.com/RSokha/IntroToProg-Python-Mod05

Assignment 05: Advanced Collections & Error Handling

Introduction

This assignment introduces a different type of collections called a dictonary. In the last assignment, we learned and practiced using lists, which is the idea of storing collections of data using brackets [] in Python. Dictionaries are similar, except that they store data values in key value pairs. Additionally, what differentiates a dictionary from a list is that a dictionary is ordered, changeable, and does not allow duplicates. For the sake of brevity, the steps of creating and utilizing a dictionary are similar steps, except we use swirly brackets. Lastly, we will show how to catch exceptions. That is, how we identify and handle any type of invalid entries or errors that may arise in our script.

Collecting The Data

First, we define our constants and variables that we will use throughout our program as shown in Figure 1.

Figure 1.

```
# Define the Data Constants
MENU: str = '''
---- 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.
-----------------------------------------
'''
FILE_NAME: str = "Enrollments.json"


# Define the Data Variables
student_first_name: str = ''  # Holds the first name of a student entered by the user.
student_last_name: str = ''  # Holds the last name of a student entered by the user.
course_name: str = ''  # Holds the name of a course entered by the user.
student_data: dict[str, str] = {}  # one row of student data
students: list = []  # a table of student data
csv_data: str = ''  # Holds combined string data separated by a comma.
```

```
file = None  # Holds a reference to an opened file.
menu_choice: str  # Hold the choice made by the user.
parts: list[str]
```

Figure 2 – we open our Json file, but now we turn our student data into a dictionary. Additionally, we handle our exceptions using a try catch to catch any errors with opening our file. For instance, if the file does not exist, we indicate this error.

Figure 2.

```
try:
    file = open(FILE_NAME, "r")
    for row in file.readlines():
        # Transform the data from the file
        parts = row.strip().split(',')
        student_first_name = parts[0]
        student_last_name = parts[1]
        course_name = parts[2]
        student_data = {'first_name': student_first_name, 'last_name':
student_last_name, 'course_name': course_name.strip()}
        # Load it into our collection (list of lists)
        students.append(student_data)
except FileNotFoundError:
    print('The file does not exist.')
    open(FILE_NAME, 'w')
except Exception as e:
    print("There is an error", type(e), e)
finally:
    if file and not file.close:
        file.close()
```

In figure 3 -- we follow the same logic and steps as the previous assignment. Let's show the menu and allow the user to input an option. Recall – we will run a while loop with a nested if else statement that says if the user chooses this option, then display this output. This is iterated for options 1-4; else, the entry will throw an exception. In other words, if the option is not 1-4, then the entry is invalid. First, we execute the logic for option 1, which allows the user to register a student and append that to the Enrollments.json file. Additionally, we had an if-not statement inside to only allow users to set parameters on the type of characters a user can enter for the student's first and last name.

Figure 3.

```
# Present and Process the data
while True:
    # Present the menu of choices
    print(MENU)
    menu_choice = input("What would you like to do: ")

    # Input user data
```

```
    if menu_choice == "1":  # This will not work if it is an integer!
        try:
            student_first_name = input("Enter the student's first name: ")
            if not student_first_name.isalpha():
                raise ValueError("Student first name must be alphabetic")
            student_last_name = input("Enter the student's last name: ")
            if not student_last_name.isalpha():
                raise ValueError("Student's last name must 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)
            print(f"You have registered {student_first_name} {student_last_name} for
{course_name}.")
        except ValueError as e:
            print(e)
```

Figure 4 – we show the data that has been entered and check it.

Figure 4.

```
# Present the current data
elif menu_choice == "2":

    # Process the data to create and display a custom message
    print("-" * 50)
    for student in students:
        print(f"Student {student['first_name']} {student['last_name']} is enrolled in
{student['course_name']}")
    print("-" * 50)
    continue
```

Figure 5 -- the student's information is entered from option 1, we want to iterate over our list and write and save the data to our Enrollments.json file. After this is complete, we always want to close out our file. Also, since we are using a dictionary, we make the correct adjustments from [] brackets to {} brackets and also implement our try-catch statements.

Figure 5.

```
elif menu_choice == "3":
    try:
        file = open(FILE_NAME, "w")
        for student in students:
            csv_data =
f"{student['first_name']},{student['last_name']},{student['course_name']}\n"
            file.write(csv_data)
        file.close()
    except Exception as e:
        print("Error saving file")
        print(e)
```

```
    finally:
        if file and not file.closed:
            file.close()
        print("The following data was saved to file!")
        for student in students:
            print(f"Student {student['first_name']} {student['last_name']} is enrolled
in {student['course_name']}")
        continue
```

Lastly, we allow the user to exit the program or break the loop, with option 4 by using the break command.

Figure 6.

```
# Stop the loop
elif menu_choice == "4":
    break  # out of the loop
```

We end with an else statement to handle exceptions; i.e., a user enters an invalid option.

Figure 7.

```
    else:
        print("Please only choose option 1, 2, or 3")

print("Program Ended")
```

Conclusion

This assignment was very similar to assignment 4, which introduced lists. Besides adjusting syntax for dictionary, we added more try-except syntax to handle exceptions.