

Jacob Widner

7/29/2024

IT FDN 110 A Su 24: Foundations of Programming: Python

Assignment05

GitHub link: <https://github.com/JakeWidner/IntroToProg-Python-Mod05>

Using JSON Files and Error Handling

Introduction

In this document I will explain the process I used to complete the fifth assignment for this course. The script used to complete this assignment builds on what was used for the previous assignments and adds new concepts such as reading and writing data using a JSON a file, using dictionaries, and error handling.

Designing the Code

Like the previous assignments, this script asks the user for input. This assignment asks the user to choose an option from a menu. The options include opportunities for user input, printing output to the screen, saving data to a file, and exiting the program. This assignment also reads data from a JSON file that was previously saved to append the data. An important difference between this assignment and the last assignment is that the data read into the program and entered by the user are saved as dictionaries instead of lists as well as adding in error handling functionality. In this section, I will discuss the concepts used in the execution of this task.

The Script Header

In the instructional materials for the first module, we were introduced to the concept of the script header. The script header is usually placed at the top or “head” of the script and consists of a series of commented lines that hold useful information about the script. These elements include a title, description, and a change log.¹ The header for this assignment (Figure 1) contains the same basic information used in the previous assignments.

```
# ----- #
# Title: Assignment05
# Desc: This assignment demonstrates using dictionaries,
#       files, and exception handling
# Change Log: (Who, When, What)
#   JWidner,7/24/2024,Created Script, added JSON functionality
#   JWidner,7/28/2024,Added in exception handling
# ----- #
```

Figure 1: Assignment05 Script Header.

¹ Root, Randal, Creating Python Scripts: Script Headers, *Mod01-Notes*, p17.

Importing JSON Library

This assignment requires the manipulation of JSON files.² To use JSON files in your script you must first import the module that contains the appropriate methods for manipulating JSON files³ (Figure 2). The io module was also required to be added to manipulate files (taken from Lab03).

```
# import json library
import json
import io as _io
```

Figure 2: JSON Module Import

Defining the Constants

We were introduced to the concepts of variables, constants and data types⁴ as well as type hints⁵ in the first module. For this assignment, a set of constants were defined including the menu that will be displayed to the user and the name of the JSON file we will be using to write the data to (Figure 3).

```
# 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.
-----
'''

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

Figure 3: Assignment05 Constants.

Defining the Variables

For this assignment, I used eight variables, as instructed. Five variables were defined as empty strings⁶, one was defined as an empty list⁷, one was defined as an empty dictionary⁸, and one variable was defined as “_io.TextIOWrapper”. All variables would be filled with values later in the code (Figure 4).

```
# Define the Data Variables and constants
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 = {} # one row of student data
students: list = [] # a table of student data
csv_data: str = "" # Holds combined string data separated by a comma.
file = _io.TextIOWrapper # Holds a reference to an opened file.
menu_choice: str = "" # Hold the choice made by the user.
```

Figure 4: Assignment05 Variables.

² Root, Randal. JSON Files, *Mod05-Notes*, p13.

³ Root, Randal. JSON Files: Working with JSON Files, *Mod05-Notes*, p15.

⁴ Root, Randal, Program Data: Constants, Variables and Data Types, *Mod01-Notes*, p27.

⁵ Root, Randal, Data Types: Data Type Naming Conventions, *Mod01-Notes*, p33.

⁶ Root, Randal. Common Types of Errors, *Mod02-Notes*, p28.

⁷ Root, Randal. Data Collections: Lists, *Mod04-Notes*, p17.

⁸ Root, Randal. Data Collections: Dictionaries, *Mod04-Notes*, p21.

Extracting Data from the JSON File

In the first set of code, we extract data from a JSON file by opening the file in read mode, extract the data using methods imported from the JSON module, and place it inside a dictionary⁹ (Figure 5). For this assignment we were also required to add error handling in our code using a Try-Except block¹⁰. This new code allows the script to continue to run when errors are encountered. For this code section, we are able to print an error message to the user if the JSON file is unable to be opened.

```
# When the program starts, read the file data into a list of lists (table)
# Extract the data from the file
try:
    file = open(FILE_NAME, "r")
    students = json.load(file)
    file.close()
except FileNotFoundError as e:
    print(f"File {FILE_NAME} not found.")
    print("File must exist before running this script!\n")
    print("Built-In Python error info: ")
    print(e, e.__doc__, type(e), sep='\n')
except Exception as e:
    print("There was an error opening the file!")
    print("Built-In Python error info: ")
    print(e, e.__doc__, type(e), sep='\n')
finally:
    if not file.closed:
        file.close()
```

Figure 5: Data Extraction Code.

The While Loop

The first part of the code establishes a loop using the while loop concept we learned in a previous module¹¹. This tells the script to continue to run the following code while a specific conditional statement is true (Figure 6). I changed the code from the starter to run the loop while menu choice does not equal 4. This allowed me to reduce the overall script length as there was no longer a need for a “break” in the loop.

```
# Present and Process the data
while menu_choice != "4":

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

Figure 6: Establish the loop.

Data Entry

If the user selects the first option, the user is given an opportunity to enter data that is saved into pre-defined variables. This is done using conditional statements¹². The data is then added to a previously defined dictionary using the append method.¹³ (Figure 7). This section also required error handling to

⁹ Root, Randal. JSON Files: Working with JSON Files, *Mod05-Notes*, p15.

¹⁰ Root, Randal. Structured Error Handling (Try-Except), *Mod05-Notes*, p19.

¹¹ Root, Randal. Loops: The while Loop, *Mod03-Notes*, p16.

¹² Root, Randal. Controlling Program Flow: The if-elif-else Statement, *Mod03-Notes*, p10.

¹³ Root, Randal. Data Collections: Dictionaries, *Mod04-Notes*, p21.

prevent the user from entering numbers as part of a first or last name. This section utilizes custom error messages¹⁴ based on which section of the code caused the error.

```
# Input user data
if menu_choice == "1": # This will not work if it is an integer!
    try:
        # Check that the first name input does not include numbers
        student_first_name = input("Enter the student's first name: ")
        if not student_first_name.isalpha():
            raise ValueError("The first name should not contain numbers.")
        # Check that the last name input does not include numbers
        student_last_name = input("Enter the student's last name: ")
        if not student_last_name.isalpha():
            raise ValueError("The last name should not contain numbers.")
        course_name = input("Please enter the name of the course: ")
        student_data = {"FirstName": student_first_name, "LastName": student_last_name, "CourseName": 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) # Prints the custom message
        print("-- Technical Error Message -- ")
        print(e.__doc__)
        print(e.__str__())
    except Exception as e:
        print("There was a non-specific error!\n")
        print("Built-In Python error info: ")
        print(e, e.__doc__, type(e), sep='\n')
    continue
```

Figure 7: Option 1, data entry.

Presenting Data to the User

If the user chooses option 2, the data that was entered by the user and appended into a dictionary variable is then printed out to the user. This is a useful troubleshooting technique which helps verify the data once it has been saved to the JSON file (Figure 8). When building this script, I noticed an error in the provided JSON file. To handle this, I used a Try-Except block to provide an appropriate error message.

```
# Present the current data
elif menu_choice == "2":
    # Process the data to create and display a custom message
    for student in students:
        try:
            print("-" * 50)
            print(f"Student {student['FirstName']} {student['LastName']} is enrolled in {student['CourseName']}")
            print("-" * 50)
        except KeyError as e:
            print(f"Key mismatch. Please ensure keys in {FILE_NAME} are correct.")
            print(f"Key {e} not found in {student}")
            print(f"Correct the keys in {FILE_NAME} and restart the program.")
            print("Built-In Python error info: ")
            print(e, e.__doc__, type(e), sep='\n')
    continue
```

Figure 8: Print out data to the user.

Data File Manipulation

If the user chooses option 3, the data is saved to a JSON file. This section of code utilizes the concepts of opening, writing data to, and closing a JSON file (Figure 9).¹⁵ This section also uses error handling in case the data does not correctly save to the JSON file.

¹⁴ Root, Randal. Structured Error Handling: Raising Custom Errors, *Mod05-Notes*, p22.

¹⁵ Root, Randal. JSON Files: Working with JSON Files, *Mod05-Notes*, p15.

```

# Save the data to a file
elif menu_choice == "3":
    try:
        file = open(FILE_NAME, "w")
        json.dump(students, file, indent=1)
        file.close()
        print("The following data was saved to file!")
        for student in students:
            print(f"Student {student['FirstName']} {student['LastName']} is enrolled in {student['CourseName']}")
        continue
    except TypeError as e:
        print("Please check that the data is a valid JSON format\n")
        print("-- Technical Error Message -- ")
        print(e, e.__doc__, type(e), sep='\n')
    except Exception as e:
        print("-- Technical Error Message -- ")
        print("Built-In Python error info: ")
        print(e, e.__doc__, type(e), sep='\n')
    finally:
        if not file.closed:
            file.close()

```

Figure 9: File Manipulation Code.

Closing the Loop and Ending the Program

The next section of code executes if the user chooses option 4 or an option not recognized by the conditional statements. If the user chooses option 4, the loop stops and the program ends. If the user chooses an option not recognized by the conditional statements, the script prompts the user to enter a new option (Figure 10).

```

# Stop the loop
elif menu_choice == "4":
    print("Program Ended")
else:
    print("Please only choose option 1, 2, or 3")

```

Figure 10: Final conditional statements to close the loop and end the program.

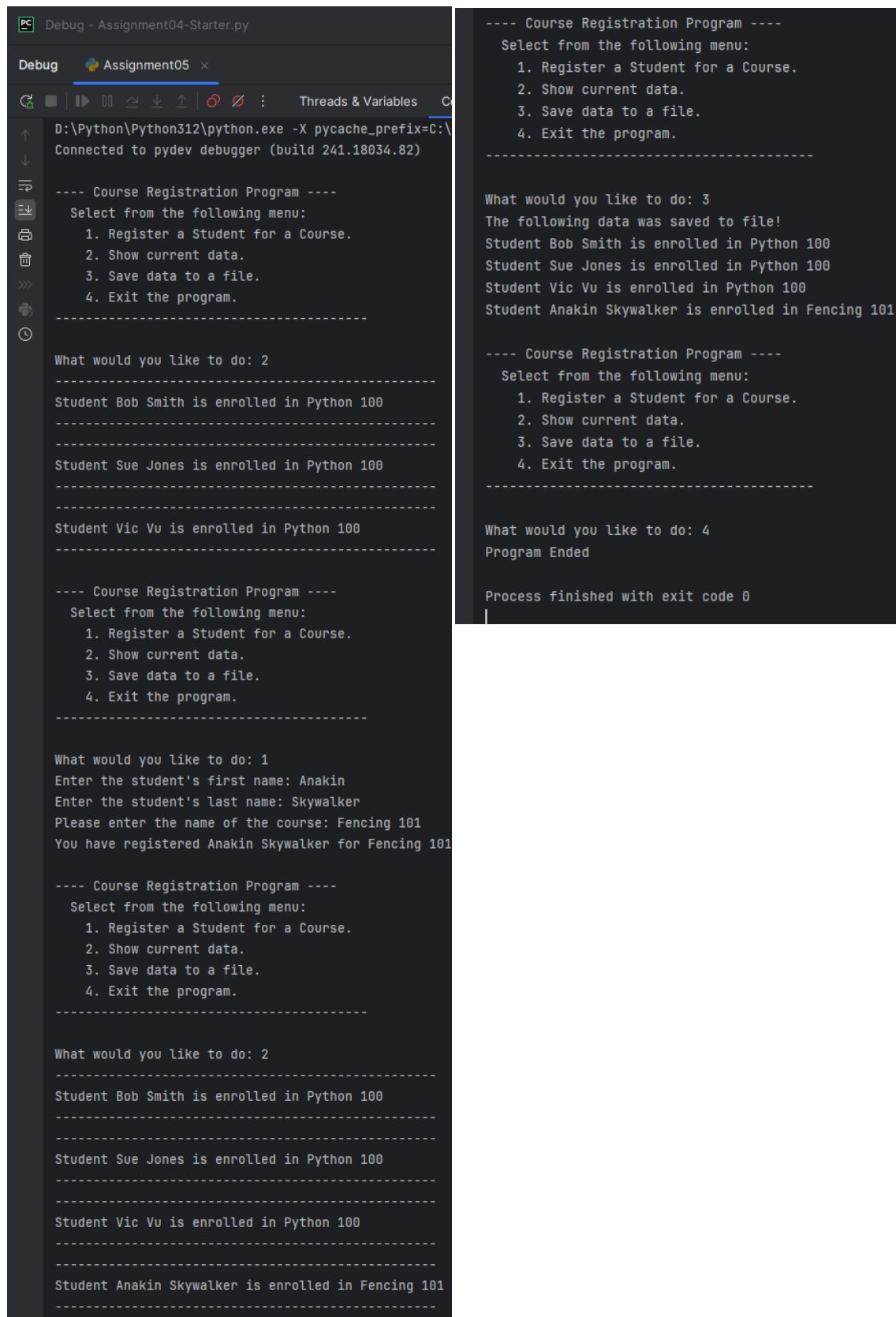
Outputs

This assignment outputs a menu to the user, asks for input from the user, and outputs the data including any data read in from the JSON file. This assignment also outputs error messages based on exception handling. To test the script, we are asked to run it using both IDE¹⁶ and the command console¹⁷. For this assignment, PyCharm¹⁸ was used as the IDE. The results of these tests can be found below in Figures 11 and 12 respectively. Additionally, I tested the error handling. The results of these tests can be found below in Figures 13 and 14.

¹⁶ Root, Randal, Integrated Development Environments: IDLE (Integrated Development and Learning Environment), *Mod01-Notes*, p14.

¹⁷ Root, Randal, The Command Shell, *Mod01-Notes*, p10.

¹⁸ Root, Randal, PyCharm, *Mod03-Notes*, p3.



```
Debug - Assignment04-Starter.py
Assignment05 x
D:\Python\Python312\python.exe -X pycache_prefix=C:\
Connected to pydev debugger (build 241.18034.82)

---- 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: 2
-----
Student Bob Smith is enrolled in Python 100
-----
Student Sue Jones is enrolled in Python 100
-----
Student Vic Vu is enrolled in Python 100
-----

---- 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: Anakin
Enter the student's last name: Skywalker
Please enter the name of the course: Fencing 101
You have registered Anakin Skywalker for Fencing 101

---- 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: 2
-----
Student Bob Smith is enrolled in Python 100
-----
Student Sue Jones is enrolled in Python 100
-----
Student Vic Vu is enrolled in Python 100
-----
Student Anakin Skywalker is enrolled in Fencing 101
-----

---- 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: 3
The following data was saved to file!
Student Bob Smith is enrolled in Python 100
Student Sue Jones is enrolled in Python 100
Student Vic Vu is enrolled in Python 100
Student Anakin Skywalker is enrolled in Fencing 101

---- 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: 4
Program Ended

Process finished with exit code 0
```

Figure 11: PyCharm Output.

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PowerShellLatest

PS E:\Documents\Important\UW Python class\_Module05\Assignment>

---- 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: 2

-----
Student Bob Smith is enrolled in Python 100
-----

Student Sue Jones is enrolled in Python 100
-----

Student Vic Vu is enrolled in Python 100
-----

---- 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: Anakin
Enter the student's last name: Skywalker
Please enter the name of the course: Fencing 101
You have registered Anakin Skywalker for Fencing 101.

---- 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: 2

-----
Student Bob Smith is enrolled in Python 100
-----

Student Sue Jones is enrolled in Python 100
-----

Student Vic Vu is enrolled in Python 100
-----

Student Anakin Skywalker is enrolled in Fencing 101
-----

---- 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: 3
The following data was saved to file!
Student Bob Smith is enrolled in Python 100
Student Sue Jones is enrolled in Python 100
Student Vic Vu is enrolled in Python 100
Student Anakin Skywalker is enrolled in Fencing 101

---- 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: 4
Program Ended
PS E:\Documents\Important\UW Python class\_Module05\Assignment>
```

Figure 12: Command Console Output.

```
D:\Python\Python312\python.exe -X pycache_prefix=C:\User
Connected to pydev debugger (build 241.18034.82)
File Enrollments.json not found.
File must exist before running this script!

Built-In Python error info:
[Errno 2] No such file or directory: 'Enrollments.json'
File not found.
<class 'FileNotFoundError'>

---- 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: 4
Program Ended

Process finished with exit code 0
```

```
import sys; print('Python %s on %s' % (sys.version, sys.platform))
D:\Python\Python312\python.exe -X pycache_prefix=C:\User
Connected to pydev debugger (build 241.18034.82)

---- 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: >? Anakin5
The first name should not contain numbers.
-- Technical Error Message --
Inappropriate argument value (of correct type).
The first name should not contain numbers.

---- 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:
>? |
```

Figure 13: PyCharm Error Handling Output

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/powershell

PS E:\Documents\Important\UW Python class\Module05\Ass
File Enrollments.json not found.
File must exist before running this script!

Built-In Python error info:
[Errno 2] No such file or directory: 'Enrollments.json'
File not found.
<class 'FileNotFoundError'>

---- 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:
```

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/powershell

PS E:\Documents\Important\UW Python class\Module05\Ass
File Enrollments.json not found.
File must exist before running this script!

---- 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: Anakin05
The first name should not contain numbers.
-- Technical Error Message --
Inappropriate argument value (of correct type).
The first name should not contain numbers.

---- 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:
```

Figure 14: Command Console Error Handling Output

I then further verified the functionality of the script by opening the file generated by it using Microsoft Notepad (Figure 15).



```
1  [
2    {
3      "FirstName": "Bob",
4      "LastName": "Smith",
5      "CourseName": "Python 100"
6    },
7    {
8      "FirstName": "Sue",
9      "LastName": "Jones",
10     "CourseName": "Python 100"
11   },
12   {
13     "FirstName": "Vic",
14     "LastName": "Vu",
15     "CourseName": "Python 100"
16   },
17   {
18     "FirstName": "Anakin",
19     "LastName": "Skywalker",
20     "CourseName": "Fencing 101"
21   }
22 ]
```

Figure 15: Data File Output.

Both the data output to the user and the data found in the JSON file match, confirming that the script works as intended.

Summary

Using the information gained from reviewing the course materials, I was able to successfully complete this assignment and create a python script that reads data from a JSON file and then presents the user with a menu with multiple options. These options include asking the user for input, presenting data back to the user, saving the data to a JSON file and exiting the program. The program provides messages to the user when errors occur in certain circumstances. This demonstrates my understanding of programming and python concepts introduced in the previous modules and new concepts introduced in this module

including but not limited error handling, reading data and writing from JSON files, creation and manipulation of dictionaries.

This assignment was a bit confusing because the JSON file provided with the assignment starter had an error in one of the keys. We did not go over handling this error, but I added it into my code anyway. Also, the notes never mention the “io” module that you need to import so that the FileNotFoundError error handling block works. The code in the lab answers had it, but it was not talked about in the notes for the lab or the video. This threw me off, but I was able to use the code from the lab to fix this issue. I hope in future modules we discuss the “io” module and its uses. I was unable to find a way to cause the TypeError to trigger to show that it works. The lab also did not cover how to test this error handling.