James Eardley

February 16, 2024

IT FDN 100 A

Assignment 05

Create a Program

Introduction

In this paper I will detail how I created a Python program that builds on the things learned in the prior modules by introducing error handling, use of JSON files, and dictionaries.

Creating the Program

When starting to write a program in any programming language, including Python, it's important to create a header that provides a title of the program, a description of what the program does, and a change log. This is helpful for allowing other users to more clearly understand the purpose of a script and who authored it.

```
# ------------------------------------------------------------------------------------------ #
# Title: Assignment03
# Desc: This assignment demonstrates using conditional logic and looping
# Change Log: (Who, When, What)
#   RRoot,1/1/2030,Created Script
#   James Eardley, 1/31/2024, Created Script
# ------------------------------------------------------------------------------------------ #
```

Figure 1.1: Header displaying the title of the assignment, description, and change log.

Some changes were made to the existing constant for the filename. This time, we used a JSON file (denoted with the extension .json) to structure our data. JSONs allow us to save data in a hierarchy (instead of a flat structure, like a csv) and work well to connect with the open and closed ends of different programming languages and systems. It is java based.

We also used a dictionary for the student date this time as it allows us to use more structure than a table. We can define the names of the columns aka keys. Different elements of the dictionary are easily accessed via keys, items and values.

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

# 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
file = None  # Holds a reference to an opened file.
menu_choice: str  # Hold the choice made by the user.
json_data: str = '' #Holds the contents of the json file
```

Figure 1.2: Define a constant for the file (JSON) which we will save to the registrations to. Here we use a JSON. We also established student_data as a dictionary this time use the open curly brackets to create an empty dictionary.

Next I used a combination of a package called json and some error handling to open the JSOB and load the data contained within it to the students list using the json.load function. The error handling first tries the string of code and catches any errors, which are explained using the exception handling.

```
try:
    import json
    file = open(FILE_NAME, 'r')
    students = json.load(file)
    file.close()

except FileNotFoundError as e:
    print("Text file must exist before running this script!\n")
    print("-- Technical Error Message -- ")
    print(e, e.__doc__, type(e), sep='\n')
except Exception as e:
    print("There was a non-specific error!\n")
    print("-- Technical Error Message -- ")
    print(e, e.__doc__, type(e), sep='\n')
finally:
    if file.closed == False:
        file.close()
```

Figure 1.3: Error handling using the Try-Except error handling methodology

Some error handling was added to the menu item 1 functionality. The code will test for known errors, but also uses some custom error types that test to make sure there isn't a numeric value included in the name of the first or last student name. When completed successfully, the information is added to a dictionary called student_data, which includes keys for the information.

```
    # Input user data
    if menu_choice == "1":  # This will not work if it is an integer!
        try:
            # Input the data
            student_first_name = input("What is the student's first name? ")
            if not student_first_name.isalpha():
                raise ValueError("The first name should not contain numbers.")

            student_last_name = input("What is 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}.")
            continue
        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("-- Technical Error Message -- ")
                print(e, e.__doc__, type(e), sep='\n')



    # Present the current data
```

Figure 1.4: Use error handling for the student's registration information and load it to a dictionary named student_data. Append it to the student list using the .append function.

Testing the Program

When done correctly, the script runs, prompting the user with a menu of items to choose from. Choosing 1 allows the user to register a student with a class. This can be run over and over to allow for the registration of multiple students. Action 2 displays the current data that has been input. Action 3 sabes the data to a csv. Finally, action 4 breaks the loop and exits the menu. Action steps 1 and 3 now utilized error handling.

Figure 1.7: The output of the script as displayed in the Console of the IDE. I tested the error handling by including a number in the first name.

The script was also run in cmd prompt to ensure that the program could be run in multiple ways.

```
---- 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
What is the student's first name? james
What is the student's last name? eardley
Please enter the name of the course: python
You have registered james eardley for python.

---- 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
--------------------------------------------------
First Name: James, Last Name: Eardley, Course: Python 100
First Name: Chloe, Last Name: Palmer, Course: Python 200
First Name: james, Last Name: eardley, Course: python
--------------------------------------------------

---- 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

---- 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
```

Summary

Assignment 05 built on the prior assignments by including the use of error handling, dictionaries, and JSON files.