Mark Stoetzl
11/14/2023
IT FDN 110 AAu 23:Foundations Of Programming Python
Assignment 05

GitHub Link: https://github.com/Stetz33/IntroToProg-Python-Mod05

# Creating Python Program Assignment 05

## Introduction

Assignment #5 focuses on taking the familiar student registration file that has been used for the last several assignments and programming it to read, process and store data in a dictionary format. The python JSON format will be used for this process. JSON has become the industry standard for data exchange between online services and other API's. The second part of assignment #5 focuses on incorporating error handling code into the program. By creating error handling routines the program becomes easier to use, more stable and easier to de-bug.

## CONSTANTS

The first step involves adding a JSON library to the assignment. This is done by incorporating the following statement at the beginning of the program: **import json** This loads the JSON library into the python program. In the data constants the FILE_NAME: str = "Enrollments.csv" is changed to the following value: FILE_NAME: str = "Enrollments.json"

## FILE OPEN STATEMENT

The file open statement is modified in two significant ways. The first change involves modifying the program to look for a file with a .json extension. The second change involves adding the following error handling language: try: ..except...except...finally. This code sequence tells the program to try the first set of code and if there are any exceptions print out the applicable error message. Then finally if none of the above programming works close the file. Illustration #1 below shows the actual code used to program this sequence.

```
try:
    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()
```

Illustration #1 – try: except except finally error handling

## MENU OPTION #1

Menu option #1 is changed by adding a “key” for each input variable to the student list. JSON uses key values to keep track of the variable information. In this case the key values are “FirstName” “Lastname” and “CourseName”. The code containing the key values is shown below in the highlighted red box in illustration #2. Illustration #2 also shows how an error handling sequence is created to make sure that only letters can be input into the FirstName and LastName variables.

```
if menu_choice == "1":
    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 = {"FirstName": student_first_name,
                   "LastName": student_last_name,
                   "CourseName": course_name}
        students.append(student)
    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')
    continue
```

Illustration #2 – variable input key values and error handling code

## MENU OPTION #2

Menu option #2 changes are involved adding the key three different  key values defined earlier in the program to the print statement. The code is shown below in illustration #3

```
# Process the data to create and display a custom message
print("-"*50)
for student in students:
    print((student["FirstName"], student["LastName"], student["CourseName"]))
print("-"*50)
continue
```

Illustration #3 – Print statement key values

## MENU OPTION #3

The third menu option actually writes the input data to a file, it also includes the error handling sequence:  try:…except…..except….finally. The JSON format writes to the file using the json.dump(students, file) command where students is the name of the list. The error handling sequence tries to write the file and if there are any exceptions  it initiates a technical error message and logs the specific details. Then finally it checks to see if the file is closed and if it is not it closes it. The code for this section of the program is shown below in figure #4.  The final option #4 to stop the program is left unchanged.

```
elif menu_choice == "3":
    try:
        file = open(FILE_NAME, "w")
        json.dump(students, file)
        file.close()
        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 file.closed == False:
            file.close()

    continue
```

Illustration #4 – File save and error handling

## Summary

Using the JSON dictionary for reading and storing the data simplified the code for the program. The use of program keys also made the code easier to follow and understand. The error handling language on the other hand made the code a lot more complicated, but I see how it increases the stability of the program and speeds up the debugging process.  It also makes it more intuitive for the user to solve any operation errors that they may be making.