Katie Harrison
June 7, 2024
IT FDN 110 B
Assignment 5
Link to GitHub

# Assignment 5 – Advanced Collections and Error Handling

## Introduction

In this assignment, I created a menu and wrote code to execute the functions of the menu. I used dictionaries and a table (list of lists) to organize my data. I also used try-catches to handle errors I anticipated users may encounter while running the code.

## Creating the Program

I used the constants (FILE_NAME and MENU) which we have been using throughout the course. I also used the global variables which the assignment suggested. I also used a few variable names which weren't suggested.

The first option asked users to enter the students' first and last names, and the course they were registering for. I compiled this information into a dictionary entry "student_data" and appended the dictionary to the table "students." I used a try-except statement to catch errors in user's inputs. If users input any numbers while they were giving student names, the ValueError would be thrown. I used a generic error message to catch unanticipated errors.

```
if menu_choice == "1":
    try:
        student_first_name = input("Please enter the student's first name: ")
        if not student_first_name.isalpha():
            raise ValueError("The first name should not contain numbers.")

        student_last_name = input("Please 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 student's course: "))

        student_data = {"First Name": student_first_name, "Last Name": student_last_name, "Course": course_name}
        students.append(student_data)

    except ValueError as value_error_details:
        print(value_error_details)
        print(" -- Technical Error Message -- ")
        print(value_error_details.__doc__)
    except Exception as unspecified_error_details:
        print("There was a non-specific error.\n")
        print("-- Technical Error Message --")
        print(unspecified_error_details,unspecified_error_details.__doc__, type(unspecified_error_details),sep = "\n")
```

The second option allowed users to see all the registrations that had been made. I wrote this code so that it would load the data from the CSV file, would separate the string into list

items, and would connect those list items to dictionary keys to form a dictionary line. Then, I appended the dictionary lines to a table called “student_list”. Next, I printed the table “student_list” by pulling the values from the dictionary keys and converting them to strings, combining them into s string line and printed them out. I added the try-except statements to catch FileNotFound and non-specified errors. the FileNotFound error will be thrown if someone has not yet created the CSV file and tries to run #2. The non-specified error will catch unanticipated errors.

```
# download data from CSV, convert to dictionary format, add to table "students", print collected data
elif menu_choice == "2":
    try:
        with open(FILE_NAME, "r") as file:
            for row in file.readlines():
                student_array:list = row.split(",")
                student_dict:dict = {"First Name" : student_array[0].strip(), "Last Name" : student_array[1].strip(), "Course" : student_array[2].strip()}
                student_list.append(student_dict)

    except FileNotFoundError as file_not_found_details:
        print("File must exist before running this script.\n")
        print("-- Technical Error Message --")
        print(file_not_found_details,file_not_found_details.__doc__, type(file_not_found_details),sep = "\n")
    except Exception as unspecified_error_details:
        print("There was a non-specific error.\n")
        print("-- Technical Error Message --")
        print(unspecified_error_details,unspecified_error_details.__doc__, type(unspecified_error_details),sep = "\n")

    print()
    print("These students have been saved to file: ")
    for row in student_list:
        student_first_name = row["First Name"]
        student_last_name = row["Last Name"]
        course_name = row["Course"]
        print(f'{student_first_name}, {student_last_name}, {course_name}')
```

The third option allows users to save the input data (from #1) to file. I added a try-except statement to catch TypeErrors and non-specified errors. The TypeError will be thrown if someone tries to save data that is not in a proper CSV format. The non-specified error will catch unanticipated errors. I cleared the table “students” after saving the data to file, so that I wouldn’t have repeated student names (from names left in the table after saving).

```
# save data collected in step 1 to CSV
elif menu_choice == "3":
    try:
        with open (FILE_NAME, "a") as file:
            for row in students:
                csv_data = f'{row["First Name"]}, {row["Last Name"]}, {row["Course"]}\n'
                file.write(csv_data)

    except TypeError as type_error_details:
        print("Please check that the data is a valid CSV format\n")
        print(" -- Technical Error Message -- ")
        print("Built-in Python error info: ")
        print(type_error_details, type_error_details.__doc__, type(type_error_details), sep = "\n")
    except Exception as unspecified_error_details:
        print("There was a non-specific error.\n")
        print("-- Technical Error Message --")
        print(unspecified_error_details,unspecified_error_details.__doc__, type(unspecified_error_details),sep = "\n")

    students.clear()
```

The fourth option allowed users to leave the menu system. I used a *break* command.

Katie Harrison
June 7, 2024
IT FDN 110 B
Assignment 5
Link to GitHub

```
elif menu_choice == "4":
    break
```

Finally, I added an else statement in case users pressed any number that wasn't on the menu. It displayed a message asking them to choose an option 1-4.

## Testing the Program

I tested this code on the Python IDE and the Python terminal. In both cases, it worked. This code was very difficult for me to figure out. For a long time, my option #2 would print the same info several times. Eventually, a friend helped me and I saw that I needed to create a new name for my dictionary and table (not the same names I was using in option #1 and #3). This helped to clean up my data – I was only printing the data from the file, not the data that had been entered but not saved (and not the data that had already been saved and was still in my table). This was a great learning experience!

## Summary

I created a *while loop* to guide users through four menu options, used dictionaries and lists to organize the data, and embedded try-except statements to catch any errors users may encounter while using the script.