

**Name:** Lorena Sainz-Maza Lecanda

**Date:** 11/11/2024

**Course:** Introduction to Programming with Python

**GitHub:** <https://github.com/lorenasml/IntroToProg-Python>

# Assignment 05 – Advanced Collections and Error Handling

## Introduction

In this assignment, I use PyCharm IDE to write a Python script that allows us to register multiple students, process data using dictionaries and save the data in a CSV file. The program uses constants, variables, print statements, string formatting, `while` and `for` loops, conditional logic, lists, dictionaries, and CSV data files. In addition, it adds exception handling constructs to handle errors more gracefully. This essay reviews the steps I took to complete the assignment and includes short observations about my performance (i.e. where I got stuck and how I resolved it).

## Steps & Observations

*Note: Since we're reusing code from past assignments, this section will primarily focus on new code I wrote to complete this assignment.*

### 1. Write the script

Like Assignment 4, I set the value of my constants and then I declare and initialize the variables. Note that, for this assignment, I introduce the `student_data` variable as a dictionary `dict = {}`.

```
21 # Define the Data Variables
22
23 student_first_name: str = ""
24 student_last_name: str = ""
25 course_name: str = ""
26 csv_data: str = ""
27 file = None
28 menu_choice: str = ""
29 student_data: dict = {}
30 students: list = []
```

Before starting the while loop, I process the data by opening the `Enrollments.csv` file in “read” mode. I iterate through each line (`each_row`) in the file and split it using a comma (,) and create a dictionary row with the extracted data, where `student_data[0]` represents the student’s first name, `student_data[1]` represents their last name, and `course_name[2]` (with `.strip()` to remove any additional spaces) represents the course

name. To append the dictionary rows to the list, I use the `append()` function and then I close the file using `close()`.

```
34     file = open(FILE_NAME, "r")
35     for row in file.readlines(): # Reads all the lines in a file and returns a list
36         # Transform the data from the file
37         student_data = row.split(',')
38         student_data = {"student_first_name": student_data[0],
39                        "student_last_name": student_data[1],
40                        "course_name": (student_data[2].strip())} # Load it into our collection (list of dictionary rows)
41         students.append(student_data)
42     file.close()
```

The output of this step is a list of dictionary rows, which I print when users' selects `menu choice == 2`. However, since this format is not very user friendly, I also add a second print statement that transforms the output in a more human-readable fashion using a `for` loop that simply prints the value of the keys in new lines.

```
84     # Present the current data
85     elif menu_choice == "2":
86         print("-"*50)
87         print(students)
88         print("-"*50)
89         for student in students:
90             print(f"{student['student_first_name']}, {student['student_last_name']}, {student['course_name']}")
91         print("-"*50)
92         continue
```

```
Enter a menu option (1-4): 2
-----
[{'student_first_name': 'Lorena', 'student_last_name': 'Hollis', 'course_name': 'Python 100'}]
-----
Lorena, Hollis, Python 100
-----
```

The program allows the user to enter multiple students' names, last names and their course name using `menu_choice == 1`. In this step, I process every student entry as a dictionary row (I cover error handling in the next section).

```
55     # Prompt user to input data
56     if menu_choice == "1":
57         try:
58             student_first_name = input("What is the student's first name? ")
59             if not student_first_name.isalpha():
60                 raise ValueError("The first name should not contain numbers.")
61
62             student_last_name = input("What is the student's last name? ")
63             if not student_last_name.isalpha():
64                 raise ValueError("The last name should not contain numbers.")
65
66             course_name = input("Which course are you enrolled in? ")
67             student_data = {"student_first_name": student_first_name,
68                            "student_last_name": student_last_name,
69                            "course_name": course_name}
70             students.append(student_data)
71             print(students)
```

```

Enter a menu option (1-4): 1
What is the student's first name? Mike
What is the student's last name? Hollis
Which course are you enrolled in? Python 500
[{'student_first_name': 'Mike', 'student_last_name': 'Hollis', 'course_name': 'Python 500'}]

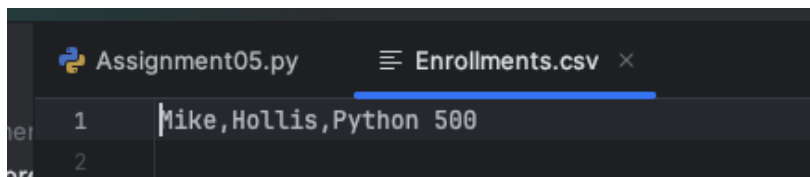
```

The program also allows the user to save the new student entries into the “Enrollments.csv” file. The code opens FILE\_NAME containing “Enrollments.csv” file in write mode and adds the values of each dictionary key into the csv file.

```

95     elif menu_choice == "3":
96         try:
97             with open(FILE_NAME, "w") as file:
98                 for student in students:
99                     file.write(f"{student['student_first_name']},{student['student_last_name']},{student['course_name']}\n")
100            file.close()

```



The screenshot shows a code editor with two tabs: 'Assignment05.py' and 'Enrollments.csv'. The 'Enrollments.csv' tab is active and displays the text 'Mike,Hollis,Python 500' on the first line. Below it, the second line is empty.

It also displays the stored data by printing each row.

```

110         with open(FILE_NAME, "r") as file:
111             for row in file:
112                 print(row.strip())

```

```

Enter a menu option (1-4): 3
The following students have been successfully enrolled:
Mike,Hollis,Python 500

```

## 1.1. Error handling

This assignment also asked to provide error handling using `try-except` constructs. First, I added `try-except` to my code in order to provide structured error handling when the file is read into the list of dictionary rows. The code below triggers the error message “*Text file must exist before running this script*” when there is no “Enrollments.csv” file created. In addition, it also prints the exception object and the documentation string of the exception type. The error output is shown below.

```

33     try:
34         file = open(FILE_NAME, "r")
35         for row in file.readlines(): # Reads all the lines in a file and returns a list
36             # Transform the data from the file
37             student_data = row.split(',')
38             student_data = {"student_first_name": student_data[0],
39                             "student_last_name": student_data[1],
40                             "course_name": (student_data[2].strip())} # Load it into our collection (list of dictionary rows)
41             students.append(student_data)
42         file.close()
43     except FileNotFoundError as e:
44         print("Text file must exist before running this script!\n")
45         print("-- Technical Error Message -- ")
46         print(e, e.__doc__, type(e), sep="\n")
47         print("Creating file...")

```

```

/usr/local/bin/python3.13 /Users/lorena/Documents/PythonClass/Module05/Assignment05.py
Text file must exist before running this script!

-- Technical Error Message --
[Errno 2] No such file or directory: 'Enrollments.csv'
File not found.
<class 'FileNotFoundError'>
Creating file...

```

The program also provides structured error handling when the user enters a first name or last name if the first name or last are not alphabetic. If the user enters a number, the program outputs the user-friendly message *"The first name should not contain numbers"*. And it does the same thing if the user enters numbers as the last name input. Note that, since these are custom conditions, I use the `if not then raise` combination. In this part of the code, lines 77-80, I also add a catch-all exception for any other non-specific errors.

```

57     try:
58         student_first_name = input("What is the student's first name? ")
59         if not student_first_name.isalpha():
60             raise ValueError("The first name should not contain numbers.")
61
62         student_last_name = input("What is the student's last name? ")
63         if not student_last_name.isalpha():
64             raise ValueError("The last name should not contain numbers.")
65
66         course_name = input("Which course are you enrolled in? ")
67         student_data = {"student_first_name": student_first_name,
68                         "student_last_name": student_last_name,
69                         "course_name": course_name}
70         students.append(student_data)
71         print(students)
72     except ValueError as e:
73         print(e) # Prints the custom message
74         print("-- Technical Error Message -- ")
75         print(e.__doc__)
76         print(e.__str__())
77     except Exception as e: #catch-all
78         print("There was a non-specific error!\n")
79         print("-- Technical Error Message -- ")
80         print(e, e.__doc__, type(e), sep="\n")

```

```

Enter a menu option (1-4): 1
What is the student's first name? 2433
The first name should not contain numbers.
-- Technical Error Message --
Inappropriate argument value (of correct type).
The first name should not contain numbers.

```

```

Enter a menu option (1-4): 1
What is the student's first name? Lucas
What is the student's last name? 3456
The last name should not contain numbers.
-- Technical Error Message --
Inappropriate argument value (of correct type).
The last name should not contain numbers.

```

Lastly, the program also provides structured error handling when the dictionary rows are written to the file. In this case, I added a `KeyError` to make sure the user adds a dictionary key as well as the catch-all exception to flag any other errors.

```

95     elif menu_choice == "3":
96         try:
97             with open(FILE_NAME, "w") as file:
98                 for student in students:
99                     file.write(f"{student['student_first_name']},{student['student_last_name']}\n")
100             file.close()
101         except KeyError as e:
102             print("Please make sure your dictionary key exists!\n")
103             print("-- Technical Error Message -- ")
104             print(e, e.__doc__, type(e), sep="\n")
105         except Exception as e: #catch all
106             print("-- Technical Error Message -- ")
107             print("Built-In Python error info: ")
108             print(e, e.__doc__, type(e), sep="\n")
109         print("The following students have been successfully enrolled:")
110         with open(FILE_NAME, "r") as file:
111             for row in file:
112                 print(row.strip())
113         continue

```

```

Enter a menu option (1-4): 3
Please make sure your dictionary key exists!

-- Technical Error Message --
'Lucas'
Mapping key not found.
<class 'KeyError'>

```

## 2. Run the program

Let's run the program to show:

- Menu option #1: The program takes the user's input for a student's first, last name, and course name.
- Iterate through option #1: The program allows users to enter multiple registrations (first name, last name, course name).

```
Enter a menu option (1-4): 1
What is the student's first name? Lorena
What is the student's last name? Hollis
Which course are you 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
-----

Enter a menu option (1-4): 1
What is the student's first name? Lucas
What is the student's last name? Hollis
Which course are you enrolled in? Python 200
```

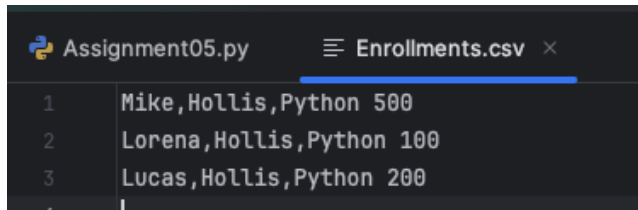
- Menu option #2: The program displays the user's input for a student's first, last name, and course name.
- Iterate through option #2: The program allows users to display multiple registrations (first name, last name, course name).

Here, the program prints the user's input in a dictionary row as well as the key values only.

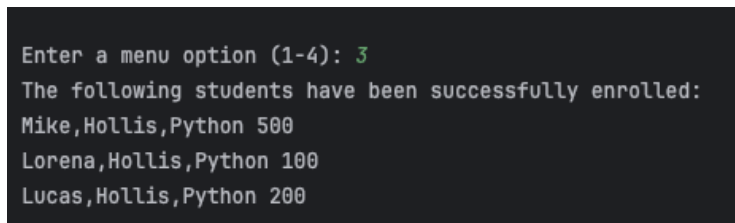
```
Enter a menu option (1-4): 2
-----
[{'student_first_name': 'Mike', 'student_last_name': 'Hollis', 'course': 'Python 500'}]
-----
Mike, Hollis, Python 500
Lorena, Hollis, Python 100
Lucas, Hollis, Python 200
-----
```

- Menu option #3: The program saves the user's input for a student's first, last name, and course name to a CSV file. (check this in a simple text editor like notepad.)
- Menu option #3: The program allows users to save multiple registrations to a file (first name, last name, course name).

The program saves the data to the csv file and also displays the stored student information in the console.



```
Assignment05.py  Enrollments.csv x
1 Mike,Hollis,Python 500
2 Lorena,Hollis,Python 100
3 Lucas,Hollis,Python 200
```



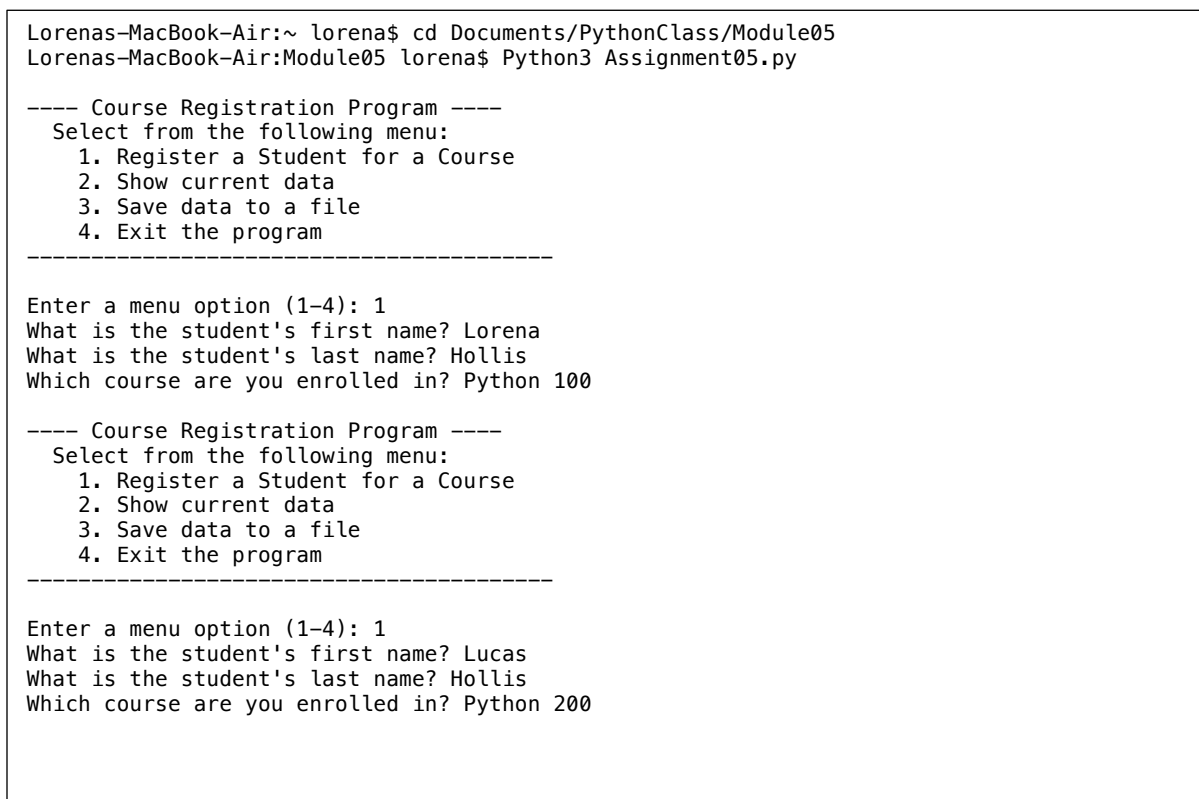
```
Enter a menu option (1-4): 3
The following students have been successfully enrolled:
Mike,Hollis,Python 500
Lorena,Hollis,Python 100
Lucas,Hollis,Python 200
```

After testing the program ran successfully in PyCharm, I reproduced it using Terminal. First, I used `cd` to navigate to the correct directory:

```
cd Documents/PythonClass/Module05
```

And then I run the script:

```
Python3 Assignment05.py
```



```
Lorenas-MacBook-Air:~ lorena$ cd Documents/PythonClass/Module05
Lorenas-MacBook-Air:Module05 lorena$ Python3 Assignment05.py

---- 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
-----

Enter a menu option (1-4): 1
What is the student's first name? Lorena
What is the student's last name? Hollis
Which course are you 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
-----

Enter a menu option (1-4): 1
What is the student's first name? Lucas
What is the student's last name? Hollis
Which course are you enrolled in? Python 200
```

```

---- 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
  -----

Enter a menu option (1-4): 2
-----
[{'student_first_name': 'Mike', 'student_last_name': 'Hollis', 'course_name': 'Python 500'},
{'student_first_name': 'Lorena', 'student_last_name': 'Hollis', 'course_name': 'Python
100'}, {'student_first_name': 'Lucas', 'student_last_name': 'Hollis', 'course_name': 'Python
200'}]
-----
Mike, Hollis, Python 500
Lorena, Hollis, Python 100
Lucas, Hollis, Python 200
-----

---- 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
  -----

Enter a menu option (1-4): 3
The following students have been successfully enrolled:
Mike,Hollis,Python 500
Lorena,Hollis,Python 100
Lucas,Hollis,Python 200

---- 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
  -----

Enter a menu option (1-4): 4
Exiting program

```

## Summary & Reflection

To summarize, Task 5 took me approximately 5-6 hours. The part that I struggled with the most was the different bracketing notations for lists vs dictionaries, I was recurrently making typos that led to errors. Also, I did not find the error handling section very intuitive and struggled to identity custom error patterns to specifically call out in my code. It required anticipating potential user errors that one might not necessarily think about at the time of writing the code. The notes and LabAnswers were very helpful to understand this last section and come up with some examples.