Akashleena Chaudhuri

15 May 2024

IT FDN 110 B Sp 24: Foundations of Programming: Python

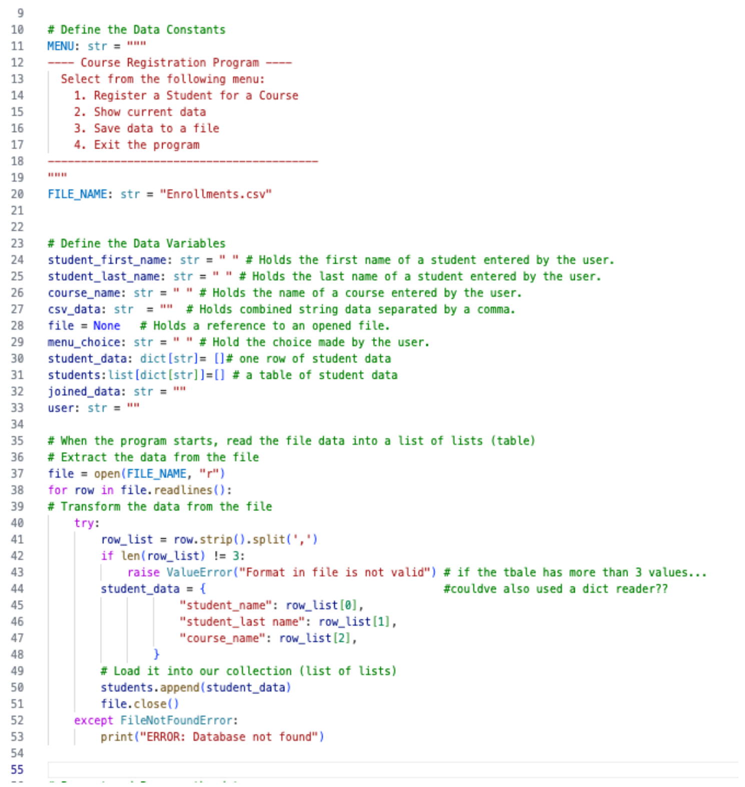
Assignment 05

# Advanced Collections and Error Handling

## Introduction

This paper will be going through a step-by-step process on how I completed Assignment 05 and my learnings along the way. It focuses on concepts like dictionaries and error handling.

## The assignment script:



```python

# 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
-----------------------------------------
"""
FILE_NAME: str = "Enrollments.csv"


# Define the Data Variables
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.
csv_data: str  = ""  # Holds combined string data separated by a comma.
file = None   # Holds a reference to an opened file.
menu_choice: str = " " # Hold the choice made by the user.
student_data: dict[str]= []# one row of student data
students:list[dict[str]]=[] # a table of student data
joined_data: str = ""
user: str = ""

# When the program starts, read the file data into a list of lists (table)
# Extract the data from the file
file = open(FILE_NAME, "r")
for row in file.readlines():
# Transform the data from the file
    try:
        row_list = row.strip().split(',')
        if len(row_list) != 3:
            raise ValueError("Format in file is not valid") # if the tbale has more than 3 values...
        student_data = {                                    #couldve also used a dict reader??
                    "student_name": row_list[0],
                    "student_last name": row_list[1],
                    "course_name": row_list[2],
                }
        # Load it into our collection (list of lists)
        students.append(student_data)
        file.close()
    except FileNotFoundError:
        print("ERROR: Database not found")


```

## Part 1

### First error handling:

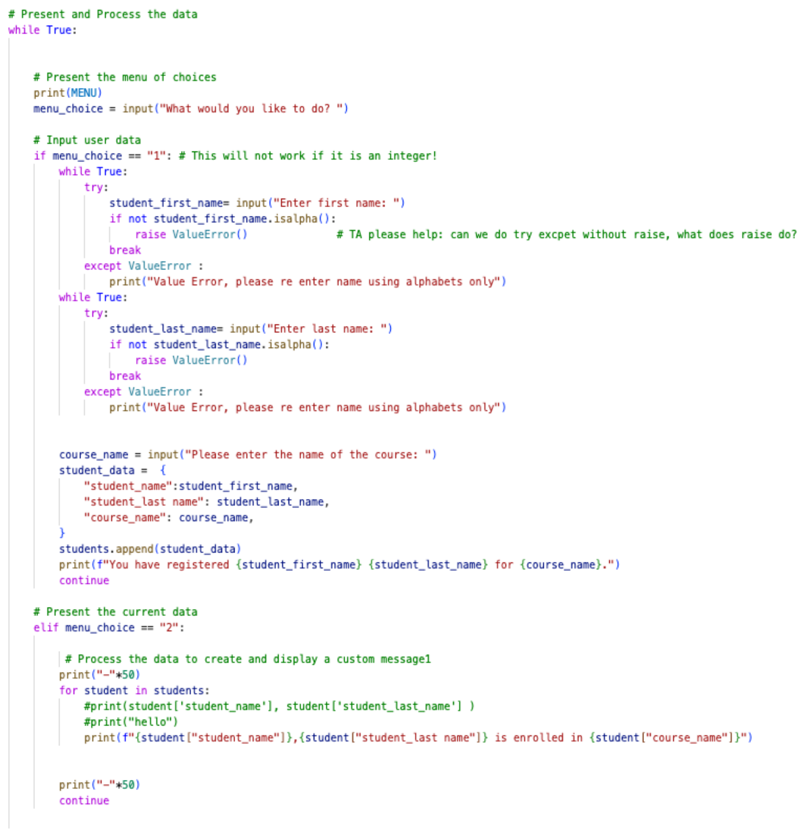
This part is reading and converting existing file’s values to a list of dictionaries.
The first error handling helps navigating if the existing file’s date is not 3 values.

For more or less than 3 values. the script is not equipped to handle and the dictionary I have set up has 3 keys!
This error handling ensure we don’t get a traceback.
This error handling also helps if the file is not found for some reason.

**FOR loop and .strip and .split:**
The code takes in values from each row from the existing files, using a **FOR** loop.
Then it used **.strip and .split** to format into a rows and for splitting with comma
.apped is used to add to the list

**Dictionary:**
Each value (represented in square brackets [0][1]…)in the rows is inserted to a **key** made for the dictionary

---



```python
# Present and Process the data
while True:


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

    # Input user data
    if menu_choice == "1": # This will not work if it is an integer!
        while True:
            try:
                student_first_name= input("Enter first name: ")
                if not student_first_name.isalpha():
                    raise ValueError()              # TA please help: can we do try excpet without raise, what does raise do?
                break
            except ValueError :
                print("Value Error, please re enter name using alphabets only")
        while True:
            try:
                student_last_name= input("Enter last name: ")
                if not student_last_name.isalpha():
                    raise ValueError()
                break
            except ValueError :
                print("Value Error, please re enter name using alphabets only")


        course_name = input("Please enter the name of the course: ")
        student_data =  {
            "student_name":student_first_name,
            "student_last name": student_last_name,
            "course_name": course_name,
        }
        students.append(student_data)
        print(f"You have registered {student_first_name} {student_last_name} for {course_name}.")
        continue

    # Present the current data
    elif menu_choice == "2":

        # Process the data to create and display a custom message1
        print("-"*50)
        for student in students:
            #print(student['student_name'], student['student_last_name'] )
            #print("hello")
            print(f"{student["student_name"]},{student["student_last name"]} is enrolled in {student["course_name"]}")


        print("-"*50)
        continue
```

## Part 2

**Second error handling:**
Here I have set up the input function to only take in alphabetic values using **.isalpha**

This part was hard, as the code would state the error then just move on with the rest of the script with the wrong non alphabetic value.

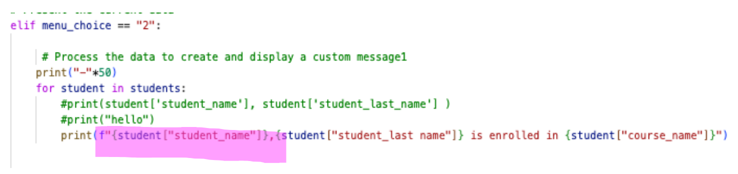
I had to create a while to so that it would keep showing the message until it recied the correct value from the user.

**Dictionary:**

Here I have restated the dictionary in order to add in more values to the list!
Earlier the **KEY** i.e student_name would take in **Value** i.e row_list[1], now it would take in **Value** i.e the input we got from the user i.e student_name

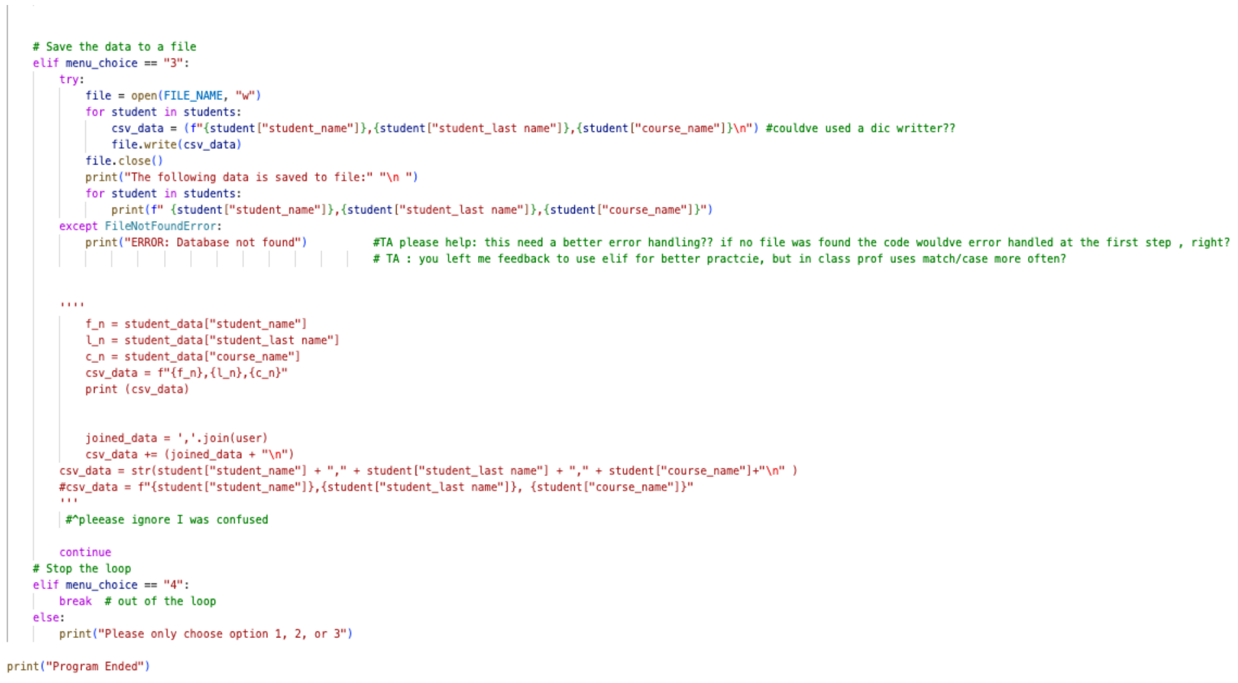
**Option 2:**

For op 2, writing the new print statement using dictionary nomenclature was quite tricky.



```
elif menu_choice == "2":

    # Process the data to create and display a custom message1
    print("-"*50)
    for student in students:
        #print(student['student_name'], student['student_last_name'] )
        #print("hello")
        print(f"{student["student_name"]},{student["student_last name"]} is enrolled in {student["course_name"]}")
```



```
    # Save the data to a file
    elif menu_choice == "3":
        try:
            file = open(FILE_NAME, "w")
            for student in students:
                csv_data = (f"{student["student_name"]},{student["student_last name"]},{student["course_name"]}\n") #couldve used a dic writter??
                file.write(csv_data)
            file.close()
            print("The following data is saved to file:" "\n ")
            for student in students:
                print(f" {student["student_name"]},{student["student_last name"]},{student["course_name"]}")
        except FileNotFoundError:
            print("ERROR: Database not found")         #TA please help: this need a better error handling?? if no file was found the code wouldve error handled at the first step , right?
                                                       # TA : you left me feedback to use elif for better practcie, but in class prof uses match/case more often?


        ''''
            f_n = student_data["student_name"]
            l_n = student_data["student_last name"]
            c_n = student_data["course_name"]
            csv_data = f"{f_n},{l_n},{c_n}"
            print (csv_data)


            joined_data = ','.join(user)
            csv_data += (joined_data + "\n")
        csv_data = str(student["student_name"] + "," + student["student_last name"] + "," + student["course_name"]+"\n" )
        #csv_data = f"{student["student_name"]},{student["student_last name"]}, {student["course_name"]}"
        '''
         #^pleease ignore I was confused

        continue
    # Stop the loop
    elif menu_choice == "4":
        break  # out of the loop
    else:
        print("Please only choose option 1, 2, or 3")

print("Program Ended")
```

## Part 3

**Third error handling:**
I was very confused on what error endling to add here.As the file not found could've been addressed in the first part

**Dictionary:**
I was trying way too hard to convert dictionary of lists to a comma separated string using the .join method.

After several attempts I realized a method to pick values of the dictionary as string to add to csv files..

**For Loop:**
Had to use For loops twice , once to save to file
Then again to save to display what is saved.

---

## The code had 2 main parts:
## Sharing notes I made during class

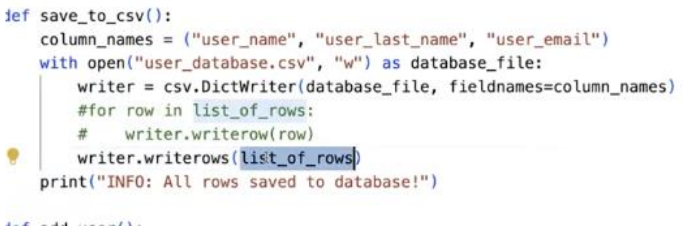
1. **Dictionary**
    - It has 2 parts a **key** and a **value**

      Row = {}
      Key. {"ID" : 1, "Name":}

      It is represented with curly brackets.

      Unlike lists where each positioned is [0][1] etc
      A dictionary values can be pulled in using the **key** name:

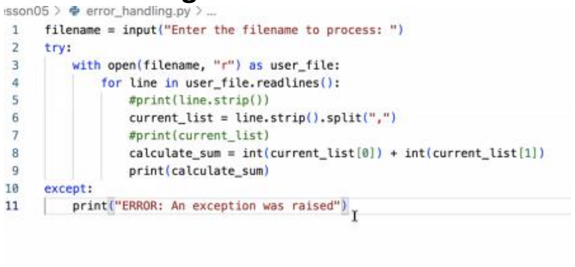
    - There can be a better method than I used to read and write a dictionary to file , like so:



```
def save_to_csv():
    column_names = ("user_name", "user_last_name", "user_email")
    with open("user_database.csv", "w") as database_file:
        writer = csv.DictWriter(database_file, fieldnames=column_names)
        #for row in list_of_rows:
        #    writer.writerow(row)
        writer.writerows(list_of_rows)
    print("INFO: All rows saved to database!")
```

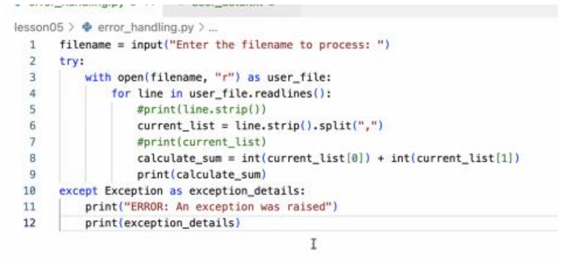
      Csv.dictwritter

2. **Error Handling:**

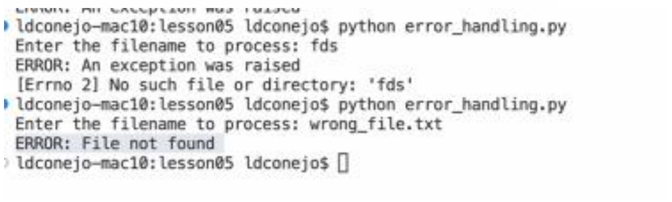


```python
filename = input("Enter the filename to process: ")
try:
    with open(filename, "r") as user_file:
        for line in user_file.readlines():
            #print(line.strip())
            current_list = line.strip().split(",")
            #print(current_list)
            calculate_sum = int(current_list[0]) + int(current_list[1])
            print(calculate_sum)
except:
    print("ERROR: An exception was raised")
```

Try and Except method. Above method is less detailed, the bottom method allows for more detail:

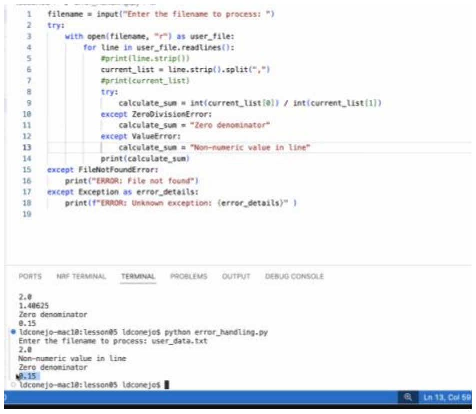


```python
filename = input("Enter the filename to process: ")
try:
    with open(filename, "r") as user_file:
        for line in user_file.readlines():
            #print(line.strip())
            current_list = line.strip().split(",")
            #print(current_list)
            calculate_sum = int(current_list[0]) + int(current_list[1])
            print(calculate_sum)
except Exception as exception_details:
    print("ERROR: An exception was raised")
    print(exception_details)
```



```
ldconejo-mac10:lesson05 ldconejo$ python error_handling.py
Enter the filename to process: fds
ERROR: An exception was raised
[Errno 2] No such file or directory: 'fds'
ldconejo-mac10:lesson05 ldconejo$ python error_handling.py
Enter the filename to process: wrong_file.txt
ERROR: File not found
ldconejo-mac10:lesson05 ldconejo$
```

Try and except can also be nested:



```python
filename = input("Enter the filename to process: ")
try:
    with open(filename, "r") as user_file:
        for line in user_file.readlines():
            #print(line.strip())
            current_list = line.strip().split(",")
            #print(current_list)
            try:
                calculate_sum = int(current_list[0]) / int(current_list[1])
            except ZeroDivisionError:
                calculate_sum = "Zero denominator"
            except ValueError:
                calculate_sum = "Non-numeric value in line"
            print(calculate_sum)
except FileNotFoundError:
    print("ERROR: File not found")
except Exception as error_details:
    print(f"ERROR: Unknown exception: {error_details}" )
```

```
2.0
1.40625
Zero denominator
0.15
ldconejo-mac10:lesson05 ldconejo$ python error_handling.py
Enter the filename to process: user_data.txt
2.0
Non-numeric value in line
Zero denominator
0.15
ldconejo-mac10:lesson05 ldconejo$
```

We can add multiple "excepts" and end with a "finally."

**Summary**

- This assignment was quite hard. I felt the try and except part was easy to understand but to implement than to a while loop was tricky.
  I was confused about using "raise"

- The dictionary concepts were also clear but sticking to the correct syntax for printing, appending and adding was quite hard.
- Still get used to for loop, and while true etc.
- This assignment took way too long but I am glad my concepts are clearer