Ophir Amon

2/12/2024

Foundations of Programming: Python

Assignment05

GitHub link: https://github.com/OphirAmon13/IntroToProg-Python

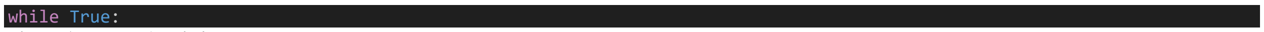
Assignment 05 Steps

Introduction

Our fourth assignment was very similar to the previous assignment, as it dealt with loops, conditional statements, lists, and CSV files. However, this assignment added two new topics: dictionaries and error handling. In this assignment we were required to use a list just like in the previous assignment, only this time the contents of the list were dictionaries. The goal of the program was the same as the previous assignment, as the program needed to be able to present a list of menu options to the user, complete the task the user selected, and then present the menu options back to the user so that they can continue to use the program without stopping. Before I presented any information to the user, I first imported the method "exit()" from the module "sys" and the module "csv" so that I could use the exit() and csv.DictReader functions later on, then I established my constant variables and the variables I will use later on in the code. I then defined all the functions I was planning on using in the code, so that I can call them in my "main".

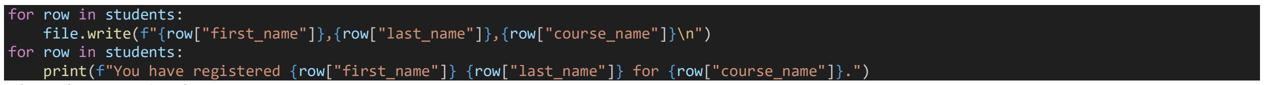
Loops

In general, loops are helpful for when you want to perform a task repeatedly. I used two different types of loops for this program: a 'while' loop and a 'for' loop. 'While' loops are helpful for when you want to iterate your program based on a condition. In this case, my condition was while the program was TRUE (Figure 1). 'For' loops are good for when we know exactly how many times we want to iterate and are therefore good when looping through lists as I did in this program (Figure 2).



```
while True:
```

Figure 1: Python 'While' Loop Example.



```
for row in students:
    file.write(f"{row["first_name"]},{row["last_name"]},{row["course_name"]}\n")
for row in students:
    print(f"You have registered {row["first_name"]} {row["last_name"]} for {row["course_name"]}.")
```

Figure 2: Python 'For' Loop Example

Conditional Statements

Much like a 'while' loop, conditional statements only run if a criterion is met. For this program, I used the match method. This method allows me to take into account however many cases I want by simply labeling the case should be. This program had 5 base cases, based on what menu option the user inputs (Figure 3). The first option allowed the user to register a student for a specific course. The user just needs the first name, last name, and the course name in order to complete the first task. The second task shows the current data that the user has input so far. This can include registration information for multiple students. The third option takes in the input data and creates a CSV file with the data, as well as informing the user that they have registered students for their courses. The fourth option ends the program by using the exit() function. The last option (case _) is for when the user selects an option that is not on the menu. When this option runs, it simply prints the message “Please select a valid option”.

```
match user_choice:
        case "1":
            add_user()
        case "2":
            print_table()
        case "3":
            save_to_csv()
        case "4":
            quit_program()
        case _:
            print("ERROR: Please select a valid option")
```

Figure 3: Python Conditional Statements Examples

CSV Files

As mentioned above, the third option of choices was to create a CSV text file with the information retrieved from the user. To do this, I used the ‘open’ function and set it to the ‘w’ (writing) mode. The open() function open a new file with any desired name and type. In this case, I opened a CSV file named ‘Enrollments.csv’ and began editing the contents of the file (Figure 4).

```
def save_to_csv(): # Saves all information to CSV file
    with open(FILE_NAME, "w") as file:
        for row in students:
            file.write(f"{row["first_name"]},{row["last_name"]},{row["course_name"]}\n")
        for row in students:
            print(f"You have registered {row["first_name"]} {row["last_name"]} for {row["course_name"]}.")
```

Figure 4: Python Open() Function Example.

Lists

Lists are useful for storing multiple pieces of data into one collection. For this program I used lists so that the user can input registration information for multiple students at the same time. The list I used in this assignment stored dictionaries in its contents (Figure 5).

```
student_data = {    # Creates dictionary of the inputted student data
    "first_name": student_first_name,
    "last_name": student_last_name,
    "course_name": course_name
}
students.append(student_data) # Adds dictionary to list of all student data
```

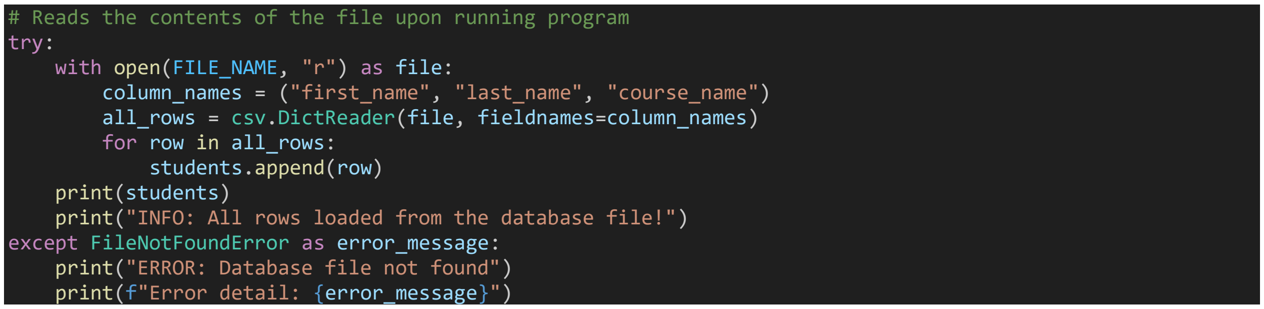
Figure 5: Python Lists/Dictionaries Example.

Dictionaries

Dictionaries are useful for storing data in key-value pairs. For each key in a dictionary there is a corresponding value. For this assignment, the keys were the strings “first_name”, “last_name”, and “course_name”, while the values were the student first name, last name, and course name that the user inputted (Figure 5).

Error Handling

Error handling is a very important aspect for the programmer to account for. Often, while writing code, the program can encounter many errors. In doing so, the program crashes, which is not very desirable. So, to ensure that does not happen, while developing the code, one should anticipate any potential errors that might come up and “handle” them. For example, the code for this assignment had a requirement that whenever the code was run, it read data from a file called Enrollments.csv. However, if the computer running the program did not have such a file stored, the program would crash and a FileNotFoundError would appear. In order to avoid having the program crash, I used a try-except block. The computer will try to execute whatever is under the try block and if it encounters an error accounted for in the except block it will print an error message instead of crashing (Figure 6).



```python
# Reads the contents of the file upon running program
try:
    with open(FILE_NAME, "r") as file:
        column_names = ("first_name", "last_name", "course_name")
        all_rows = csv.DictReader(file, fieldnames=column_names)
        for row in all_rows:
            students.append(row)
    print(students)
    print("INFO: All rows loaded from the database file!")
except FileNotFoundError as error_message:
    print("ERROR: Database file not found")
    print(f"Error detail: {error_message}")
```

Figure 6: Python Error Handling Example.

Summary

In conclusion, this assignment utilized six topics: loops, conditional statements, CSV files, lists, dictionaries, and error handling. I utilized a ‘while’ loop to continuously print the options from the menu to the user and I used a ‘for’ loop in order to relay to the user what task they completed in case three. I used conditional statements to complete a task based on the menu option the user chose. I used the information the user inputted to create a CSV file with the information in it. I used lists to store dictionaries of the student information given by the user. Lastly, I used error handling to ensure that my program would not crash.