

Name: Sohail Nassiri

Date: August 31st, 2024

Course: IT FDN 110 B Su 24 - Foundations of Programming: Python

GitHub URL: <https://github.com/int221/IntroToProg-Python>

Assignment 05 – Creating a Python Script Using Conditional Logic, Looping, Dictionaries and Exception Handling

Introduction

This assignment requires the use of constants, variables, and a variety of functions, in a Python script. In addition, it incorporates the use of string formatting, while and for loops, programming menus, conditional logic, dictionaries, exception handling and comma-separated value (CSV) files to work with data. The purpose of the Python script is to allow users to enter and display multiple registrations for students and their associated courses along with the ability to read and save the data to/from a CSV file if desired.

Creating the Python Script

Using the PyCharm IDE, a Python script is created. The code begins with the script header, as shown in **Figure 1**, in order to give some background information on the purpose of the script. In this case, this script is being written in order to complete the requirements of Assignment 5 demonstrating the use of important concepts in Module 05 being:

- Dictionaries
- Exception handling
- String formatting
- Functions: split, strip, append, readlines
- Conditional logic
- Looping
- Reading, writing and storing data to/from a CSV file

```
# ----- #
# Title: Assignment05
# Desc: This assignment demonstrates using dictionaries, files, and exception handling
# Change Log: (Who, When, What)
#   Sohail Nassiri,08/31/2024,Created Script
# ----- #
```

Figure 1. Script Header

First, the Main section is started by defining the data constants (see **Figure 2**), which are designated in upper case. Also, type hints are used to allow the reader to know the data type associated with each constant or variable. The following constants are assigned:

- The course registration program menu (see below), is assigned to MENU (string).

---- 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

- The name of the .csv file, 'enrollments.csv', is assigned to FILE_NAME (string)

```
# 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" # Set the csv file name
```

Figure 2. Defining the Data Constants

Next, the data variables are defined and set to empty strings or objects as follows (see **Figure 3**):

- student_first_name (string) is set to empty string
- student_last_name (string) is set to empty string
- course_name (string) is set to empty string
- csv_data (string) is set to empty string
- file (object) is set to None
- menu_choice (string) is set to empty string
- student_data (dict) is set to an empty dictionary
- students (list) is set to an empty list

```
# 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.
student_data: dict = {} # one row of student data
students: list = [] # a table of student data
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.
```

Figure 3. Defining the Data Variables

Then, the user is prompted to select from the course registration program menu. The options allow the user to enter multiple registrations for students and their courses, display the data collected, read and save data to/from a CSV file, and/or exit the program. In addition, exception handling is incorporated in order to guide the user through possible errors that may be encountered. This is carried out as follows (see **Figures 4a-4c**):

- When the program begins, the data in “enrollments.csv” is automatically read into a two-dimensional table (a list of dictionary rows) by doing the following:
 - The file is opened using the open() function and read using “r” mode.
 - A for loop is used to iterate and read through each line in the file with the readlines function
 - The split function is used to split the string into a dictionary row using a comma separator
 - Each key-value pair is defined. Dictionaries use this format. For example, see below:
 - “FirstName”: student_data[0]
 - The strip function is used to remove any extra whitespace and carriage returns
 - The append function is used to add the data from the file to the two-dimensional table (students)
 - A try-except-finally block is added within the code to serve the following functions:
 - Try – Attempts to run the code in the respective block
 - Except – Runs if exception occurs in the Try block to allow the user to know the cause of the error. In this case, the specific exception is the FileNotFoundError if the .csv file is not found. A general Exception error is added as well in case any other exception that is not specifically called out arises.
 - Finally – Runs regardless of whether code successfully executes or if exception is raised to ensure a specific action is carried out. In this case, it is to make sure the .csv file is closed before performing any further actions.

```

# When the program starts, read the file data into a list of dictionary rows (table)
# Extract the data from the file
try:
    file = open(FILE_NAME, "r") # Reads file
    for row in file.readlines(): # Reads through each line
        # Transform the data from the file
        student_data = row.split(",") # Splits string into a list of dictionary rows using a comma separator
        student_data = {"FirstName": student_data[0],
                        "LastName": student_data[1],
                        "CourseName": student_data[2].strip()} # Removes unnecessary spaces and carriage returns
        # Load it into our collection (list of dictionary rows)
        students.append(student_data) # Adds data to two-dimensional table
    file.close()
except FileNotFoundError as e: # Raises exception if file is not found
    print("Text file must exist before running this script!\n") # Prints custom message
    print("-- Technical Error Message -- ")
    print(e, e.__doc__, type(e), sep="\n") # Provides details of error
except Exception as e: # Raises any other general exception that is not specifically called out
    print("There was a non-specific error!\n")
    print("-- Technical Error Message -- ")
    print(e, e.__doc__, type(e), sep="\n")
finally:
    if file:
        file.close() # Closes file regardless of whether code successfully executes or not

```

Figure 4a. Reading Data in CSV File and Adding it to a Two-Dimensional Table with Exception Handling

- A while loop is set to true, which results in an infinite loop until exited by a break. Inside the while loop, the course registration program menu is printed and the user is asked what selection they would like to make.

- Conditional logic is used, specifically if, else-if (elif) and else statements, to carry out the proper action based on the selection made:
 - If the user selects “1”, they are prompted to enter the student’s first and last name along with the course name. The data collected is displayed and added to the two-dimensional table. In addition, if a non-alphabetical name is entered for the first and last name, a ValueError is raised to notify the user and prompt them to try again. Again, the try-except block is incorporated to achieve this.
 - If the user selects “2”, the current data collected is displayed. This is done by using a for loop to iterate through each row of the table and print all of the data in the dictionary table using a f-string.
 - If the user selects “3”, the current data collected by the user is written to the ‘enrollments.csv’ file that was created earlier in the script and displayed as well. The file is opened using the open() function. In this case, the file is opened using write mode (‘w’). The .csv file is written with the content of the csv_data variable using the write() function. Finally, the file is saved and closed using the close() function. Similar to when the .csv file was being read earlier in the script, a try-except-finally block is used in case the file is not found.
 - If the user selects “4”, the while loop is exited by introducing a break. In addition, this ends the program and the user is notified with a print statement.
 - If the user enters in an invalid selection, the user is notified with a print statement and prompted to select from the menu again.
- As a result of the infinite while loop, the user will continuously be prompted to make selections from the menu until the choice of exiting the program is made.

```
# 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!
        try:
            student_first_name = input("Enter the student's first name: ")
            if not student_first_name.isalpha(): # Requires user to input alphabetical name
                raise ValueError("The first name should not contain numbers.") # Raises error if non-alphabetical
            # character is entered
            student_last_name = input("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 name of the course: ")
            student_data = {"FirstName": student_first_name,
                           "LastName": student_last_name,
                           "CourseName": course_name}
            students.append(student_data) # Table is appended with data from list of dictionary row
            print(
                f"You have registered {student_first_name} {student_last_name} for {course_name}." # Displays inputted
                # registration
            )
        except ValueError as e:
            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")
        continue
```

Figure 4b. Receiving Inputs from the User, Presenting Data to the User, and Processing Data to a CSV File using Dictionaries, Exception Handling, Loops and Conditional Logic

```

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

    # Process the data to create and display a custom message
    print("-" * 50)
    for student in students: # Iterates through each row of table
        print(
            f"Student {student['FirstName']} {student['LastName']} is enrolled in {student['CourseName']}"
        )
    print("-" * 50)
    continue

# Save the data to a file
elif menu_choice == "3":
    try:
        file = open(FILE_NAME, "w") # Opens file in write mode
        for student in students:
            csv_data = f"{student['FirstName']},{student['LastName']},{student['CourseName']}\n"
            file.write(csv_data) # Writes the content of the csv_data variable to the file
        file.close()
        print("The following data was saved to file!")
        for student in students:
            print(
                f"Student {student['FirstName']} {student['LastName']} is enrolled in {student['CourseName']}"
            )
    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:
            file.close()
        continue

# Stop the loop
elif menu_choice == "4":
    break # out of the loop
else:
    print("Please only choose option 1, 2, or 3") # Returns user to menu if invalid selection made

print("Program Ended")

```

Figure 4c. Receiving Inputs from the User, Presenting Data to the User, and Processing Data to a CSV File using Dictionaries, Exception Handling, Loops and Conditional Logic

Running the Python Script

To start, when running the script, the user is prompted to select from the course registration program menu.

Figures 5a – 5c and Figures 6a – 6b show what occurs with each selection as described in the “Creating the Python Script” section when running in the PyCharm IDE and command prompt, respectively. This example includes if the user enters a non-alphabetical name, multiple student registrations, and if the starting .csv file does not exist. When properly running the script, to prevent an error from occurring when reading the enrollments.csv file, the following starting data was entered: Vic,Vu,Python 100.

```
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'>

---- 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.
-----

What would you like to do: 1
Enter the student's first name: Vic
Enter the student's last name: Vu
Please enter the name of the course: Python 100
You have registered Vic Vu for 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.
-----

What would you like to do: 1
Enter the student's first name: J4
-- Technical Error Message --
The first name should not contain numbers.
Inappropriate argument value (of correct type).
<class 'ValueError'>

---- 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.
-----
```

Figure 5a. Running the Python Script from the PyCharm IDE

```

---- 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.
-----

What would you like to do: 1
Enter the student's first name: John
Enter the student's last name: D4
-- Technical Error Message --
The last name should not contain numbers.
Inappropriate argument value (of correct type).
<class 'ValueError'>

---- 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.
-----

What would you like to do: 1
Enter the student's first name: John
Enter the student's last name: Doe
Please enter the name of the course: Math 100
You have registered John Doe for Math 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.
-----

What would you like to do: 3
The following data was saved to file!
Student Vic Vu is enrolled in Python 100
Student John Doe is enrolled in Math 100

```

Figure 5b. Running the Python Script from PyCharm IDE

```
---- 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.  
-----  
  
What would you like to do: 4  
Program Ended  
  
Process finished with exit code 0
```

Figure 5c. Running the Python Script from PyCharm IDE


```

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

---- 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.
-----

What would you like to do: 1
Enter the student's first name: Vic
Enter the student's last name: Vu
Please enter the name of the course: Python 100
You have registered Vic Vu for 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.
-----

What would you like to do: 1
Enter the student's first name: J4
-- Technical Error Message --
The first name should not contain numbers.
Inappropriate argument value (of correct type).
<class 'ValueError'>

---- 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.
-----

What would you like to do: 1
Enter the student's first name: John
Enter the student's last name: D4
-- Technical Error Message --
The last name should not contain numbers.
Inappropriate argument value (of correct type).
<class 'ValueError'>

---- 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.
-----

What would you like to do: 1
Enter the student's first name: John
Enter the student's last name: Doe
Please enter the name of the course: Math 100
You have registered John Doe for Math 100.

```

Figure 6a. Running the Python Script from the Command Prompt

```

---- 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.
-----

What would you like to do: 2
-----
Student Vic Vu is enrolled in Python 100
Student John Doe is enrolled in Math 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.
-----

What would you like to do: 3
The following data was saved to file!
Student Vic Vu is enrolled in Python 100
Student John Doe is enrolled in Math 100

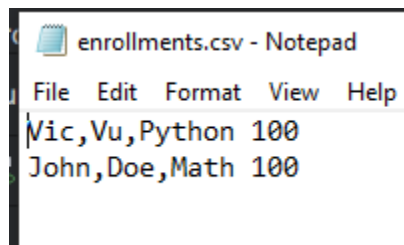
---- Course Registration Program ----
Select from the following menu:
  1. Register a Student for a Course.
  2. Show current data.
  3. Save data to a file.
  4. Exit the program.
-----

What would you like to do: 4
Program Ended

```

Figure 6b. Running the Python Script from the Command Prompt

As shown below in **Figure 7**, when selecting Option #3, the data is written and saved to the 'enrollments.csv' file.



```

enrollments.csv - Notepad
File Edit Format View Help
Vic,Vu,Python 100
John,Doe,Math 100

```

Figure 7. Data Stored in .CSV File After Running the Python Script

Summary

The goal of this assignment was to demonstrate the creation of a Python script that prompts the user to select from a course registration program menu in order to allow for entering and displaying multiple registrations for students and their associated courses using dictionaries and exception handling along with the ability to read data from/save data to a CSV file. Constants, variables, various functions (`input()`, `print()`, `open()`, `close()`, `split`, `strip`, `append`, etc.), string formatting, while and for loops, conditional logic, dictionaries, exception handling and reading data from/writing data to CSV files are utilized in this script in order to achieve the desired outcome.