Name: Niki Way
Date: 03/05/2025
Course: IT FDN 110 A Wi 25: Foundations Of Programming: Python
GitHub: https://github.com/anway0123/IntroToProg-Python

Working with Classes

Introduction

For this assignment, I created a Python program that built on last week's assignment in that, I created a program for students' registration for a Python course, but this week, in addition to the constants, variables, print statements, loops, programming menus, conditional logic, dictionaries, JSON, and exception handling. I added the use of the following: functions, classes, and using the separation of concerns pattern. In this paper, I will go over my process how the script was created and tested as well as my thoughts on the results.

Beginning the Script

This script began with the reader being updated to reflect my name, current date, and what I was does with the script to follow the flow of the change log of Who,When, and What. From there I imported the JSON file I would be working with using the import feature.(Figure 1)

The constants and variables

This process mostly remains unchanged from previous version of this script; however of note there are fewer variables over all. I did add one additional constant for a CSV file as the assignment requested, under bullet point number three under the Test subhead, "The program saves the user's

input for a student's first, last name, and course name to a coma-separated[sic] string file.(check this in a simple text editor like notepad.)" (Root, Assignment 6, Page 4)(Figure 1-2)

```python
# ------------------------------------------------------------------------------------------ #
# Title: Assignment06_Starter
# Desc: This assignment demonstrates using functions
# with structured error handling
# Change Log: (Who, When, What)
#  Niki Way, 3/5/2025, Created Script
# ------------------------------------------------------------------------------------------ #


# Processing ---------------------------------- #
import json

# 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.
----------------------------------------
'''
# Define the Data Constants
FILE_NAME_CSV: str = "Enrollments.csv"
FILE_NAME: str = "Enrollments.json"

# Define the Data Variables
students: list = []
menu_choice: str

```

Figure 1: Displays: Header, Import, Constants, and Variables

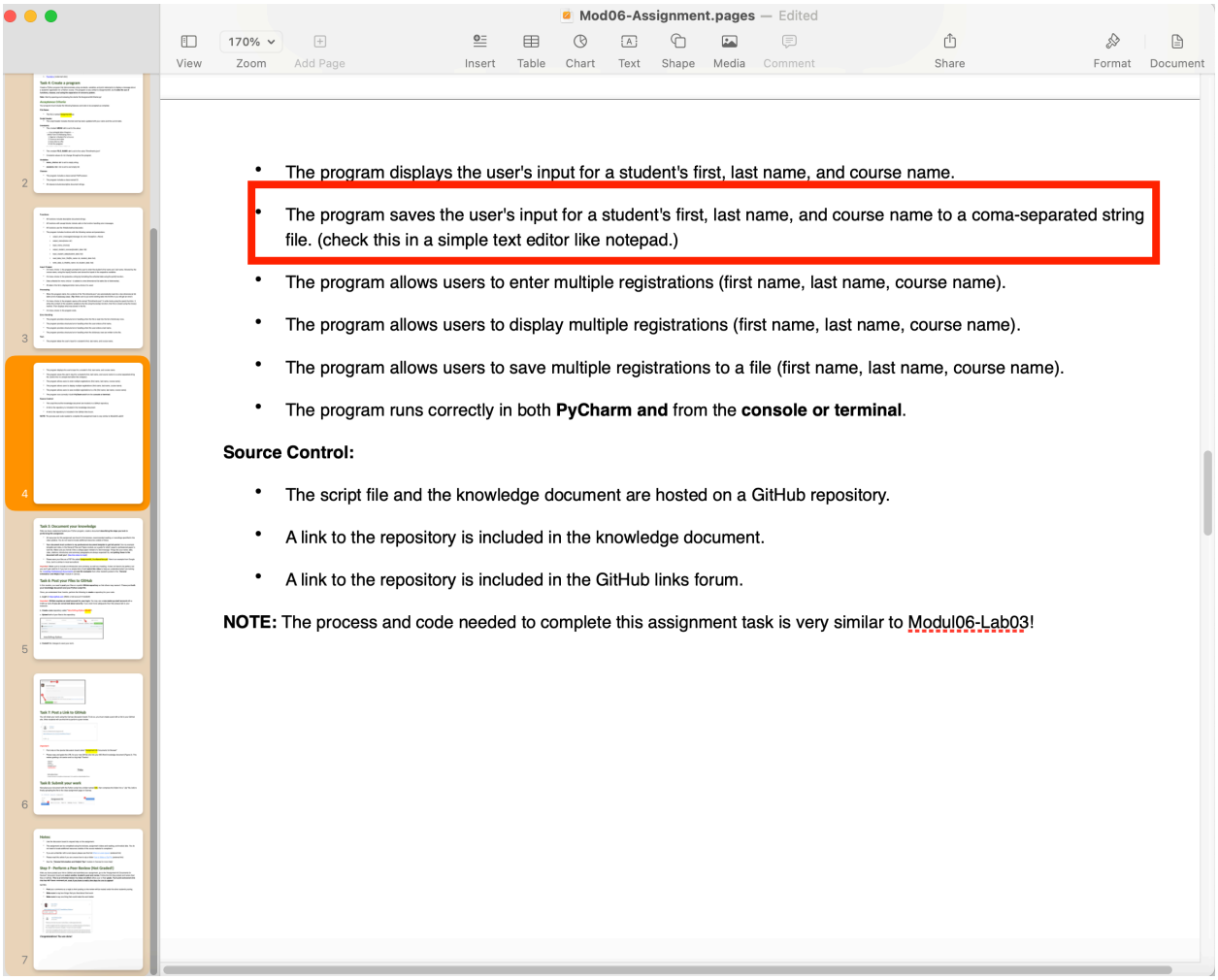

Figure 2: Reference and citation of Assignment 6 request

Creating the Class

This program uses two different classes. One named FileProcessor that houses all the file processing functions like opening, reading, writing, and closing the files.(Figure 3-4) The other class is the IO for all the input and output functions, which include the input and output from the user and also the error messages which are outputs themselves.(Figure 5-9) All of the classes have a descriptive document string.

Functions

The functions as previously stated fall into one of the two class types above. However there are some overlapping features of all of the functions. Such has all the functions have the following: descriptive document string (like the classes), exception blocks for handling error messages, and use the @staticmethon decorator. From on here they differ.

Under the FileProcessor class, there is the “read_data_from_file” function which opens the file, creates a two dimensional list table. It was created similar to las week’s assigned task. I opened the file via open() and used “r” to express that I want to read the file. Using the previously imported JSON file. (See Figure 1) I used the json.load(file) method to parse the JSON data from the "Enrollments.json" file into a Python list of dictionaries (student_data) This function also included two Exception blocks for error handling. One regarding the file not being found and the other for a nonspecific error. The were created with the “IO.output_error_message()” statement which then contained the message that I had typed into the parentheses. From there the file is closed, if only the file was not already closed and returns “student_data” (Figure 3)

Additionally under this class there is the “write_data_to_file” function which writes the data that is inputted by the user to both the imported JSON and a CSV file, creating a comma-separated string which was cited earlier. Using the same method from last week, once the file is open, the data is written into the open JSON file using the json.dump, the JSON file is closed. There is a print statement for the user explaining that there data has been saved. The CSV file writing works much the same. The CSV file is opened, the data is written using a “f” string and the file.write() method, and then the file is closed. All of the previously mentioned Exception blocks from are also present in this function as well. (Figure 4)

```python
class FileProcessor:  2 usages
    """
    A collection of processing layer functions that work with Json files

    ChangeLog: (Who, When, What)
    NWay,3.4.2025,Created Class
    """
    pass

    @staticmethod  1 usage
    def read_data_from_file(file_name: str, student_data: list):
        """
        This function reads data from file and convert it into a list of dictionaries

        ChangeLog: (Who, When, What)
        NWay,3.4.2025,Created function
        :param file_name: file that is being read from
        :param student_data: list of dictionary rows
        :return: list of dictionaries
        """
        try:
            file = open(file_name, "r")
            student_data = json.load(file)
            file.close()
        except FileNotFoundError as e:
            IO.output_error_messages( message: "Text file must exist before running this script!",e)
        except Exception as e:
            IO.output_error_messages( message: "There was a non-specific error!",e)
        finally:
            if file.closed == False:
                file.close()
            return student_data
```

Figure 3: "read_data_from_file" function

```python
class FileProcessor:  2 usages
    @staticmethod  1 usage
    def write_data_to_file(file_name: str, file_name_csv: str, student_data: list):
        # global file
        # global students
        """ This function writes data to a json file with data from a list of dictionary rows
        ChangeLog: (Who, When, What)
        NWay,3.4.2025,Created function
        :param file_name: JSON file that is being written in
        :param file_name_csv: CSV file that is being written in
        :param student_data: list of dictionary rows
        :return: None
        """
        try:

            # JSON
            file = open(file_name, "w")
            json.dump(student_data, file)
            file.close()

            print("The following data was saved to file!")
            for student in students:
                print(f'Student {student["FirstName"]} '
                      f'{student["LastName"]} is enrolled in {student["CourseName"]}')

            # CSV
            file = open(file_name_csv, "w")
            for student in students:
                csv_data = f'{student["FirstName"]},{student["LastName"]},{student["CourseName"]}\n'
                file.write(csv_data)
            file.close()
        except TypeError as e:
            IO.output_error_messages( message: "Please check that the data is a valid JSON or CSV format", e)
        except Exception as e:
            IO.output_error_messages( message: "There was a non-specific error!", e)
        finally:
            if file.closed == False:
                file.close()
```

Figure 4: "write_data_to_file" function

The following functions are contained within the IO class: "output_error_messages", "output_menu", "input_menu_choice", "input_student_data", and "output_student_courses".

Beginning with the "output_error_messages" this will display to the user a custom error message by using the print statement if there is an error present or as the "if" statement says,"not none" (Figure 5)

For “output_menu”, this function simply prints the menu to the user by using the print statement. (Figure 6)
The “input_menu_choice” function receives the menu selection from the user with the use of a try loop and the choice equalling the input from the user. Also there is a raised exception if the user inputs a number that isn’t in the “if” loop. Additionally there are also Exception block for error message for a technical error. This function returns the choice made by the user. (Figure 7)

The “input_student_data” function gets the student’s first name, last name, and course name from the user. There is also a try loop for this is function which contains the input lines for the user. Both the student_first_name and the student_last_name have a raise exception for ValueError to flag the user should they input any other characters other than alpha characters. Once the data is collected from the user then there is a print statement letting the user know the input was successful. Additionally there are also Exception block for error messages for the ValueError and the generic “Exception as e” for a generic exception raised.(Figure 8)

The “output_student_courses” function displays the student and course names to the user. To do this, I used a print statement using a “f” string and student dictionary. (Figure 9)

```
103  class IO:
104      """
105      A collection of presentation layer functions that manage user input and output
106
107      ChangeLog: (Who, When, What)
108      NWay,3.4.2025,Created Class
109      """
110
111      @staticmethod  7 usages
112      def output_error_messages(message: str, error: Exception = None):
113          """
114          This function displays a custom error messages to the user
115          ChangeLog: (Who, When, What)
116          NWay,3.4.2025,Created function
117          :param message: string with message for user display
118          :param error: Exception with technical message for user display
119          :return: None
120          """
121          print(message, end="\n\n")
122          if error is not None:
123              print("-- Technical Error Message -- ")
124              print(error, error.__doc__, type(error), sep='\n')
125
```

Figure 5: “output_error_messages” function

```
@staticmethod  1 usage
def output_menu(menu: str):
    """
    This function displays the menu of choices to the user
    ChangeLog: (Who, When, What)
    NWay,3.4.2025,Created function
    :return: None
    """
    print()  # Adding extra space to make it look nicer.
    print(menu)
    print()  # Adding extra space to make it look nicer.
```

Figure 6: “output_menu” function

```
@staticmethod  1 usage
def input_menu_choice():
    """ This function gets a menu choice from the user

    ChangeLog: (Who, When, What)
    NWay,3.4.2025,Created function
    :return: string with the users choice
    """
    choice = "0"
    try:
        choice = input("Enter your menu choice number: ")
        if choice not in ("1","2","3","4"):  # Note these are strings
            raise Exception("Please, choose only 1, 2, 3, or 4")
    except Exception as e:
        IO.output_error_messages(e.__str__())  # Not passing e to avoid the technical message
    return choice
```

Figure 7: "input_menu_choice" function

```
@staticmethod  1 usage
def input_student_data(student_data: list):
    """ This function gets the student's first name and last name, with a course name from the user

    ChangeLog: (Who, When, What)
    NWay, 3.5.2025,Created function
    :param student_data: list of dictionary rows that the input data will be added to
    :return: list of dictionary rows
    """
    try:
        student_first_name = input("Enter the student's first name: ")
        if not student_first_name.isalpha():
            raise ValueError("The last name should not contain numbers.")
        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)
        print(f"You have registered {student_first_name} {student_last_name} for {course_name}.")
    except ValueError as e:
        IO.output_error_messages( message: "Mismatch Values",e)
    except Exception as e:
        IO.output_error_messages( message: "Error: There was a problem with your entered data.",e)
```

Figure 8: "input_student_data" function

```
@staticmethod  1 usage
def output_student_courses(student_data: list):
    """ This function displays the student and course names to the user
            ChangeLog: (Who, When, What)
            NWay,3.5.2025,Created function
            :param student_data: list of dictionary rows
            :return: None
            """
    print("-" * 50)
    for student in students:
        print(f'Student {student["FirstName"]} '
              f'{student["LastName"]} is enrolled in {student["CourseName"]}')
    print("-" * 50)
```

Figure 9: "output_student_courses" function

The Main Body

The program starts off by reading the file data into a list of lists and then extracting that data. To do this, I used the function the FileProcessor class for read_data_from_file. (Figure 10) From there I started the While loop, like has been previously used then I present the menu to the user by with the IO class function IO.outout_menu and then stating that the menu_choice equals the user's input by using the IO class function IO.input_menu_choice(). (Figure 10) Then we start collecting data from the user with the if/elif loop. With menu option one, allows for user input with the IO class function IO.input_student_data(students). Menu option two presents the current data back to the user by using IO class function IO.output_student_courses(students)(Figure 10) Menu option three saves the data to both a JSON file and a CSV file by using the FileProcessor class function FileProcessor.write_data_to_file(FILE_NAME, FILE_NAME_CSV, students). (Figure 10) Finally menu option four stops the loop and prints "Program Ended" Should the user select anything other than numbers one through four then the loop will uses the print statement "Please only choose option 1,2,3, or 4". (Figure 10)

```
# When the program starts, read the file data into a list of lists (table)
# Extract the data from the file
students = FileProcessor.read_data_from_file(FILE_NAME, students)

# Present and Process the data
while (True):

    # Present the menu of choices
    IO.output_menu(MENU)
    menu_choice = IO.input_menu_choice()

    # Input user data
    if menu_choice == "1":
        IO.input_student_data(students)

    # Present the current data
    elif menu_choice == "2":
        IO.output_student_courses(students)

    # Save the data to a file
    elif menu_choice == "3":
        FileProcessor.write_data_to_file(FILE_NAME,FILE_NAME_CSV,students)
        #Data was saved to a CSV file also as requested in bullet point 3 under Test on the assignment

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

print("Program Ended")
```

Figure 10: Main body of the script

Testing

For testing I was able to to obtain the following outcomes: the script takes the user's input for a student's first name, last name, and course name, displays the above mentioned user's input, saves the user's previously mentioned input to both a JSON file and CSV file, allows for users to enter multiple registrations, displays those multiple registrations, save those multiple registrations and run correctly in both PyCharm and terminal. (Figure 11-15)

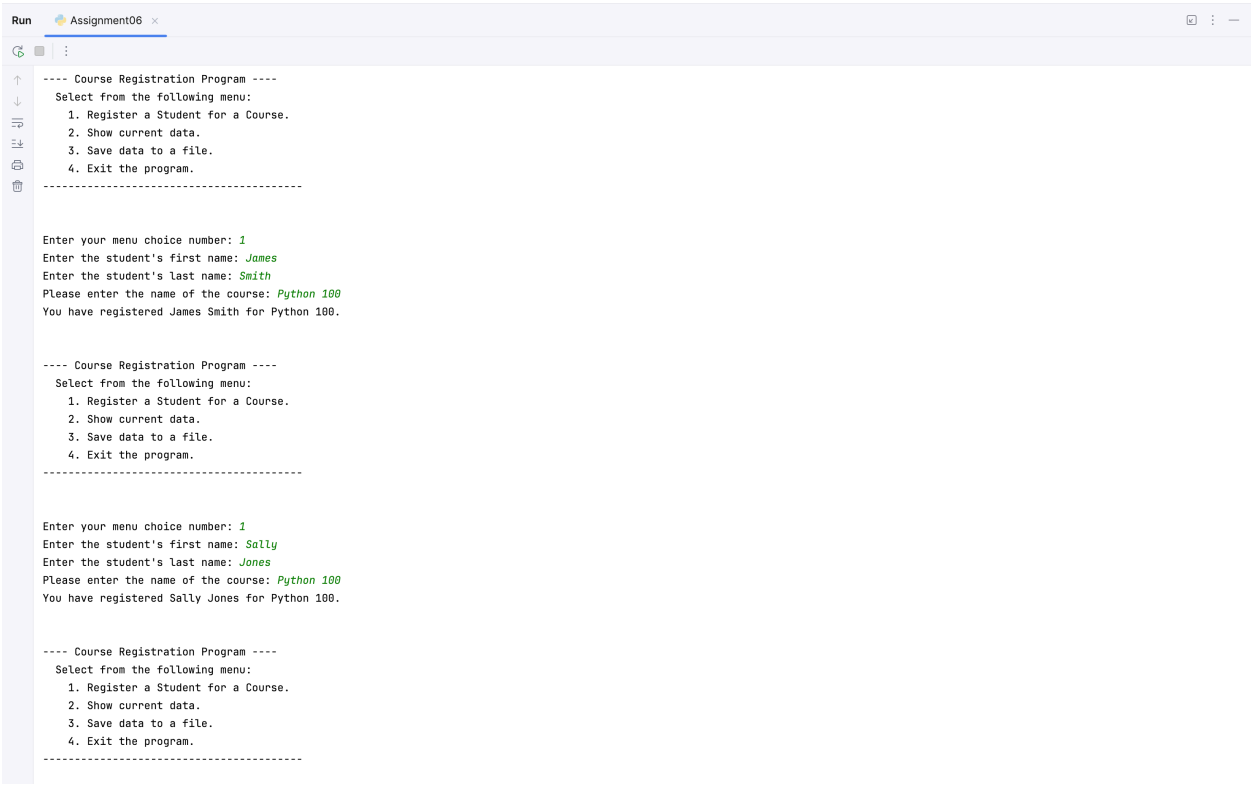

Figure 11: Menu Option 1 outcomes

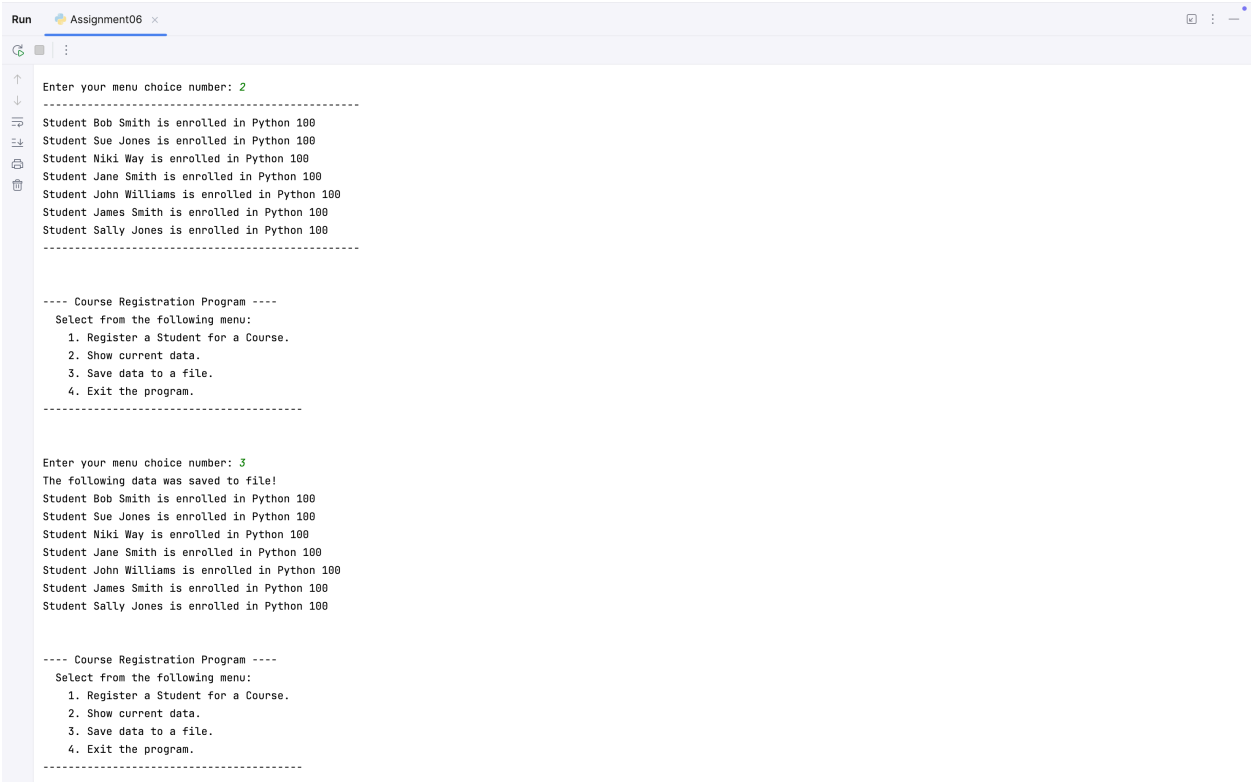

Figure 12: Menu Options 2 and 3 outcomes

```
Run  Assignment06

Student John Williams is enrolled in Python 100
Student James Smith is enrolled in Python 100
Student Sally Jones is 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 your menu choice number: 3
The following data was saved to file!
Student Bob Smith is enrolled in Python 100
Student Sue Jones is enrolled in Python 100
Student Niki Way is enrolled in Python 100
Student Jane Smith is enrolled in Python 100
Student John Williams is enrolled in Python 100
Student James Smith is enrolled in Python 100
Student Sally Jones is 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 your menu choice number: 4
Program Ended

Process finished with exit code 0
```

Figure 13: Menu Options 3 and 4 outcomes

```
Last login: Wed Feb 26 18:49:18 on ttys000
nikiway@Nikis-MacBook-Pro ~ % cd documents/python/PythonCourse/A06
nikiway@Nikis-MacBook-Pro A06 % python3 Assignment06.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 your menu choice number: 1
Enter the student's first name: Matt
Enter the student's last name: Smith
Please enter the name of the course: Python
You have registered Matt Smith for Python.


---- 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 your menu choice number: 1
Enter the student's first name: David
Enter the student's last name: Johnson
Please enter the name of the course: Python 100
You have registered David Johnson 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.
-----------------------------------------


Enter your menu choice number: 2
--------------------------------------------------
Student Bob Smith is enrolled in Python 100
Student Sue Jones is enrolled in Python 100
Student Niki Way is enrolled in Python 100
Student Jane Smith is enrolled in Python 100
Student John Williams is enrolled in Python 100
```

Figure 14: Terminal Outcomes for Menu Options 1 and 2

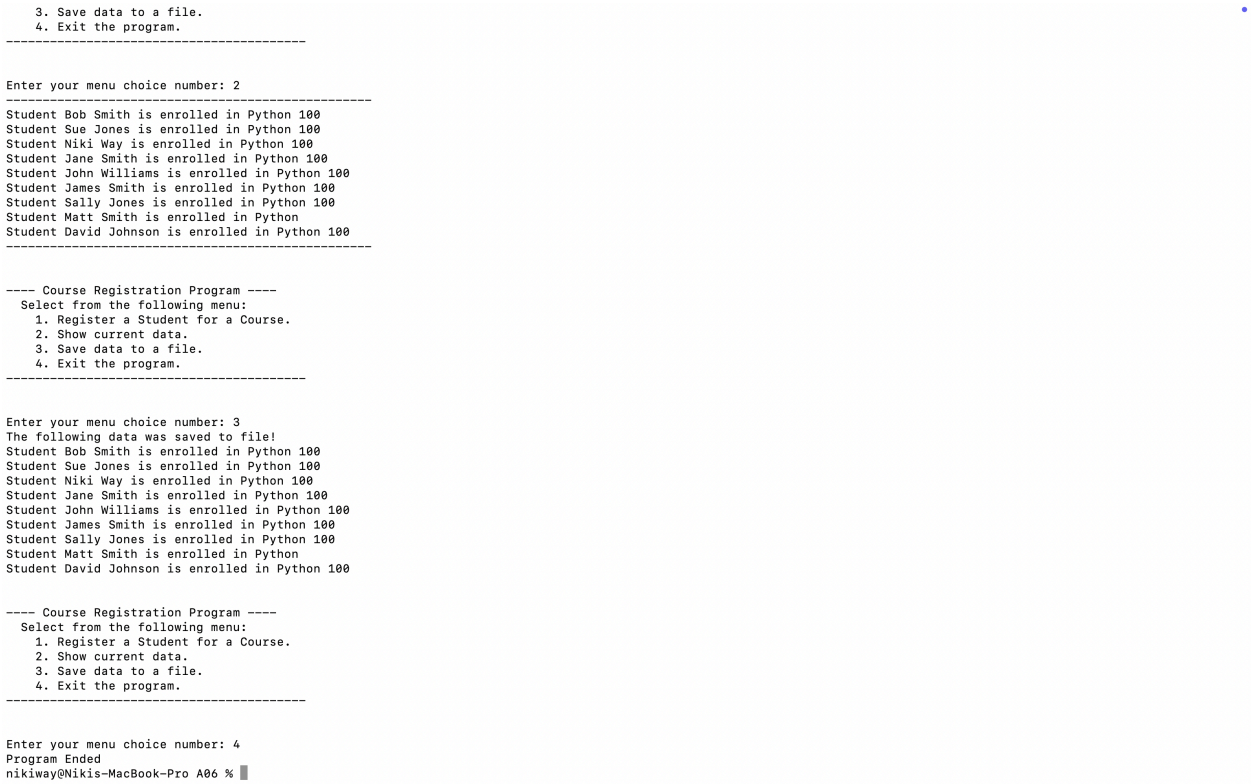

Figure 15: Terminal Outcomes for Menu Options 2-4

Summary

This week's assignment seemed to be a bit easier for me to understand. Last week I struggled with Exception blocks and where to place them. The classes and the functions seem to help brake things up and helped me with the blocking. Also the functions really seem to streamline the way in script things are written