Name: Ian Basques-Jellison
Date: August 20, 2025
Course: IT FDN 110 A Su 25: Foundations of Programming: Python
Assignment: Assignment06
GitHubURL: https://github.com/ianjelly/IntroToProg-Python-Mod06

Using Functions

Contents

Introduction

This document describes the steps performed in the creation of a Python program, “Assignment06.py”, that is very similar to “Assignment05.py”, except that it additionally uses functions. Most of the program’s functionality and acceptance criteria is already accomplished by the starter file, “Assignment06-Starter.py”. This document describes how the starter file is expanded upon to accomplish the remaining acceptance criteria.

Acceptance Criteria

Acceptance criteria for the program are defined in the “Mod06-Assignment.docx” file for the Introduction to Python Course IT FDN 110 A. The program must include the following components:

- File Name: Assignment06.py
- Script Header (Title, Program Description, and Change Log)
- The constants: **MENU** and **FILE_NAME**
- The variables: **menu_choice** and **students**
- The classes: FileProcessor and IO
- The functions:
    - output_error_messages(message: str, error: Exception = None)
    - output_menu(menu: str)
    - input_menu_choice()
    - input_student_data(student_data: list)
    - output_student_courses(student_data: list)
    - read_data_from_file(file_name:str, student_data: list)
    - write_data_to_file(file_name:str, student_data: list)
- Error handling

When the program starts, the contents of the “Enrollments.json” are automatically read into the **students** two-dimensional list of lists (table).

The program uses a four-option menu, and does the following for each menu choice:

Menu Choice (1)	prompts the user to enter the student’s first and last name, and the course name
Menu Choice (2)	prints a comma-separated strings composed of the first name, last name, course name for all the data in the **students** variable, including any data initially in the file
Menu Choice (3)	writes the contents of the **students** variable to the “Enrollments.json” and displays what was written to the file
Menu Choice (4)	ends the program

Much of the program’s functionality and acceptance criteria is already accomplished by the starter file, “Assignment06-Starter.py”. The acceptance criteria that are <u>not</u> accomplished by the starter file, are satisfied by the steps described in this document.

Program Construction

The program consists of a script file organized into three sections: a Header, Setup code, and the Main Body. These sections, their subcomponents (define constants, define variables, etc.), and the majority of the functionality of the program are specified by the starter file, “Assignment06-Starter.py”. The starter file is shown in Figure 1, Figure 2 and Figure 3.

```
# ------------------------------------------------------------------------------------------ #
# Title: Assignment06_Starter
# Desc: This assignment demonstrates using functions
# with structured error handling
# Change Log: (Who, When, What)
#   RRoot,1/1/2030,Created Script
#   <Your Name Here>,<Date>,<Activity>
# ------------------------------------------------------------------------------------------ #
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: str = "Enrollments.csv"
FILE_NAME: str = "Enrollments.json"

# Define the Data Variables and constants
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
file = None  # Holds a reference to an opened file.
menu_choice: str = ''  # Hold the choice made by the user.

# When the program starts, read the file data into a list of lists (table)
# Extract the data from the file
try:
    file = open(FILE_NAME, "r")

    students = json.load(file)

    file.close()
except Exception as e:
    print("Error: There was a problem with reading the file.")
    print("Please check that the file exists and that it is in a json format.")
    print("-- Technical Error Message -- ")
    print(e.__doc__)
    print(e.__str__())
finally:
    if file.closed == False:
        file.close()
```

Figure 1: Start file Assignment06-Starter.py (1 of 3)

```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!

        try:
            student_first_name = input("Enter the student's first name: ")
            if not student_first_name.isalpha():
                raise ValueError("The first 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:
            print(e)  # Prints the custom message
            print("-- Technical Error Message -- ")
            print(e.__doc__)
            print(e.__str__())
        except Exception as e:
            print("Error: There was a problem with your entered data.")
            print("-- Technical Error Message -- ")
            print(e.__doc__)
            print(e.__str__())
        continue

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

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

Figure 2: Start file Assignment06-Starter.py (continued, 2 of 3)

```
    # Save the data to a file
    elif menu_choice == "3":

        try:
            file = open(FILE_NAME, "w")
            json.dump(students, 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"]}')
        except Exception as e:
            if file.closed == False:
                file.close()
            print("Error: There was a problem with writing to the file.")
            print("Please check that the file is not open by another program.")
            print("-- Technical Error Message -- ")
            print(e.__doc__)
            print(e.__str__())
        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")
```

Figure 3: Start file Assignment06-Starter.py (continued, 3 of 3)

Header

The script header in the Python program uses the same format as defined in the acceptance criteria from the assignment document file, except with the Change Log updated with my name and date. Figure 4 shows the script header for “Assignment06.py”.

```
# ---------------------------------------------------------------------------------- #
# Title: Assignment06
# Desc: This assignment demonstrates using functions
# with structured error handling
# Change Log: (Who, When, What)
#   Ian Basques-Jellison, 8/17/2025, Created Script
#   <Your Name Here>, <Date>, <Activity>
# ---------------------------------------------------------------------------------- #
```

Figure 4: Header from Assignment06.py

Setup Code and Main Body

The starter file, “Assignment06-Starter.py”, defines all the necessary constants and variables, except that most of these variables will not be needed in the final version of the code, since it will use functions and parameters instead of relying on the global variables. This approach takes advantage of encapsulation and flexibility, while ensuring predictable behavior (Root R., Module 06 - Functions, 2025). However, before commenting out all the unnecessary global variables, the necessary functions need to be implemented first.

FileProcessor and IO Classes

Since the acceptance criteria specifies that the program includes a class named FileProcessor and a class named IO, I added these two classes to the code first. Figure 5 shows the new class named FileProcessor and IO, with the word “pass” as a place holder for future code.

```
# Processing ------------------------------------------------------------------------- #
class FileProcessor:
    """
    A collection of processing layer functions for Json files

    ChangeLog: (Who, When, What)
    Ian Basques-Jellison, 8/17/2025, Created Class
    """
    pass

# Presentation ----------------------------------------------------------------------- #
class IO:
    """
    A collection of presentation layer functions for user input and output

    ChangeLog: (Who, When, What)
    Ian Basques-Jellison, 8/17/2025, Created Class
    """
    pass
```

Figure 5: New Class FileProcessor and IO, With Placeholder Code

output_error_messages() Function

The first function added to the IO class is a function to present error messages to the user. Figure 6 shows the output_error_messages() function.

```
@staticmethod
def output_error_messages(message: str, error: Exception = None):
    """ This function displays error messages to the user

    ChangeLog: (Who, When, What)
    Ian Basques-Jellison, 8/17/2025, Created Function

    :return: None
    """
    print(message, end="\n\n")
    if error is not None:
        print("-- Technical Error Message -- ")
        print(error, error.__doc__, type(error), sep='\n')
```

Figure 6: output_error_messages() function in IO Class

read_data_from_file() Function

The next function to be added is the read_data_from_file() function in the FileProcessor class. This is essentially the same code that is in lines 37 thru 51 of the starter file, “Assignment06-Starter.py”, except the error handling has been replaced by our new output_error_message() function. After we add the new read_data_from_file() function to the FileProcessor class, we call the function in the Main Body. Figure 1 shows the read_data_from_file() function.

```
@staticmethod
def read_data_from_file(file_name: str, student_data: list):
    """ This function reads data from a json file

    ChangeLog: (Who, When, What)
    Ian Basques-Jellison, 8/17/2025, Created Function

    :return: a table of student data
    """
    try:
        file = open(file_name, "r")
        student_data = json.load(file)
        file.close()
    except FileNotFoundError as e:
        IO.output_error_messages("Please check that the file exists " \
                                 "and that it is in a json format.", e)
    except Exception as e:
        IO.output_error_messages("There was a non-specific error " \
                                 "reading the file.", e)
    finally:
        if file.closed == False:
            file.close()
    return student_data
```

Figure 7: read_data_to_file() function definition in FileProcessor Class

Figure 8 shows the read_data_to_file() function called out in the Main Body.

```
# ------------------------------------ MAIN BODY ------------------------------------- #
# 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=FILE_NAME, student_data=students)
```

Figure 8: read_data_to_file() function called in Main Body

output_menu() and input_menu_choice() Functions

After adding the read_data_from_file() function, add the output_menu() and the input_menu_choice() functions in the IO class. and replace them in the Main Body.

Figure 9 shows the output_menu() and input_menu_choice() functions.

```
@staticmethod
def output_menu(menu: str):
    """ This function displays the menu to the user

    ChangeLog: (Who, When, What)
    Ian Basques-Jellison, 8/17/2025, Created Function

    :return: None
    """
    print(menu)

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

    ChangeLog: (Who, When, What)
    Ian Basques-Jellison, 8/17/2025, Created Function

    :return: string with the user's choice
    """
    choice = "0"
    try:
        choice = input("What would you like to do: ")
        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 the exception object
                                               # to avoid the technical message
    return choice
```

Figure 9: output_menu() and input_menu_choice() function definitions in IO Class

Figure 10 shows the output_menu() and input_menu_choice() functions called out in the Main Body.

```
# Present and Process the data
while (True):

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

Figure 10: output_menu() and input_menu_choice() functions called in Main Body

input_student_data() Function (Menu Choice 1)

Next, the input_student_data() function is added to the IO class, and replaced in the Main Body (menu choice 1).

Figure 11 shows the input_student_data() function.

```
@staticmethod
def input_student_data(student_data: list):
    """ This function gets first name, last name, and course from the user

    ChangeLog: (Who, When, What)
    Ian Basques-Jellison, 8/17/2025, Created Function

    :return: str
    """
    try:
        student_first_name = input("Enter the student's first name: ")
        if not student_first_name.isalpha():
            raise ValueError("The first 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 = {"FirstName": student_first_name,
                   "LastName": student_last_name,
                   "CourseName": course_name}
        student_data.append(student)
        print(f"You have registered {student_first_name} {student_last_name} " \
              f"for {course_name}.")
    except ValueError as e:
        IO.output_error_messages(f"{e}", e)
    except Exception as e:
        IO.output_error_messages("Error: There was a problem with " \
                                 "your entered data.", e)
    return student_data
```

Figure 11: input_student_data() function definition in IO Class

Figure 12 shows the input_student_data() function called out in the Main Body (menu choice 1).

```
# Input user data
if menu_choice == "1":  # This will not work if it is an integer!
    students = IO.input_student_data(student_data=students)
    continue
```

Figure 12: input_student_data() function called in Main Body

output_student_courses() Function (Menu Choice 2)

The output_student_courses() function is added next to the IO class, and replaced in the Main Body (menu choice 2).

Figure 13 shows the output_student_courses() function. This is also the last function needed for the IO class.

```
    @staticmethod
    def output_student_courses(student_data: list):
        """ This function displays all the registered students and their courses

        ChangeLog: (Who, When, What)
        Ian Basques-Jellison, 8/17/2025, Created Function

        :return: None
        """
        # Process the data to create and display a custom message
        print("-" * 50)
        for student in student_data:
            print(f'Student {student["FirstName"]} '
                  f'{student["LastName"]} is enrolled in {student["CourseName"]}')
        print("-" * 50)

# End of function definitions
```

Figure 13: output_student_courses() function definition in IO Class

Figure 14 shows the output_student_courses() function called out in the Main Body (menu choice 2).

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

Figure 14: output_student_courses() function called in Main Body

write_data_to_file() Function (Menu Choice 3)

Finally, the write_data_to_file() function is added to the FileProcessor class, and replaced in the Main Body (menu choice 3).

Figure 15 shows the write_data_to_file() function. This is the last function needed for the FileProcessor class.

```
@staticmethod
def write_data_to_file(file_name: str, student_data: list):
    """ This function writes data to a json file

    ChangeLog: (Who, When, What)
    Ian Basques-Jellison, 8/17/2025, Created Function

    :return: None
    """
    try:
        file = open(file_name, "w")
        json.dump(student_data, file, indent=2)

        file.close()
        print("The following data was saved to file!")
        for student in student_data:
            print(f'Student {student["FirstName"]} '
                  f'{student["LastName"]} is enrolled in {student["CourseName"]}')
    except Exception as e:
        if file.closed == False:
            file.close()
        IO.output_error_messages("Error: There was a problem with writing " /
                                 "to the file.\n" /
                                 "Please check that the file is not open " /
                                 "by another program.", e)
```

Figure 15: write_data_to_file() function definition in FileProcessor Class

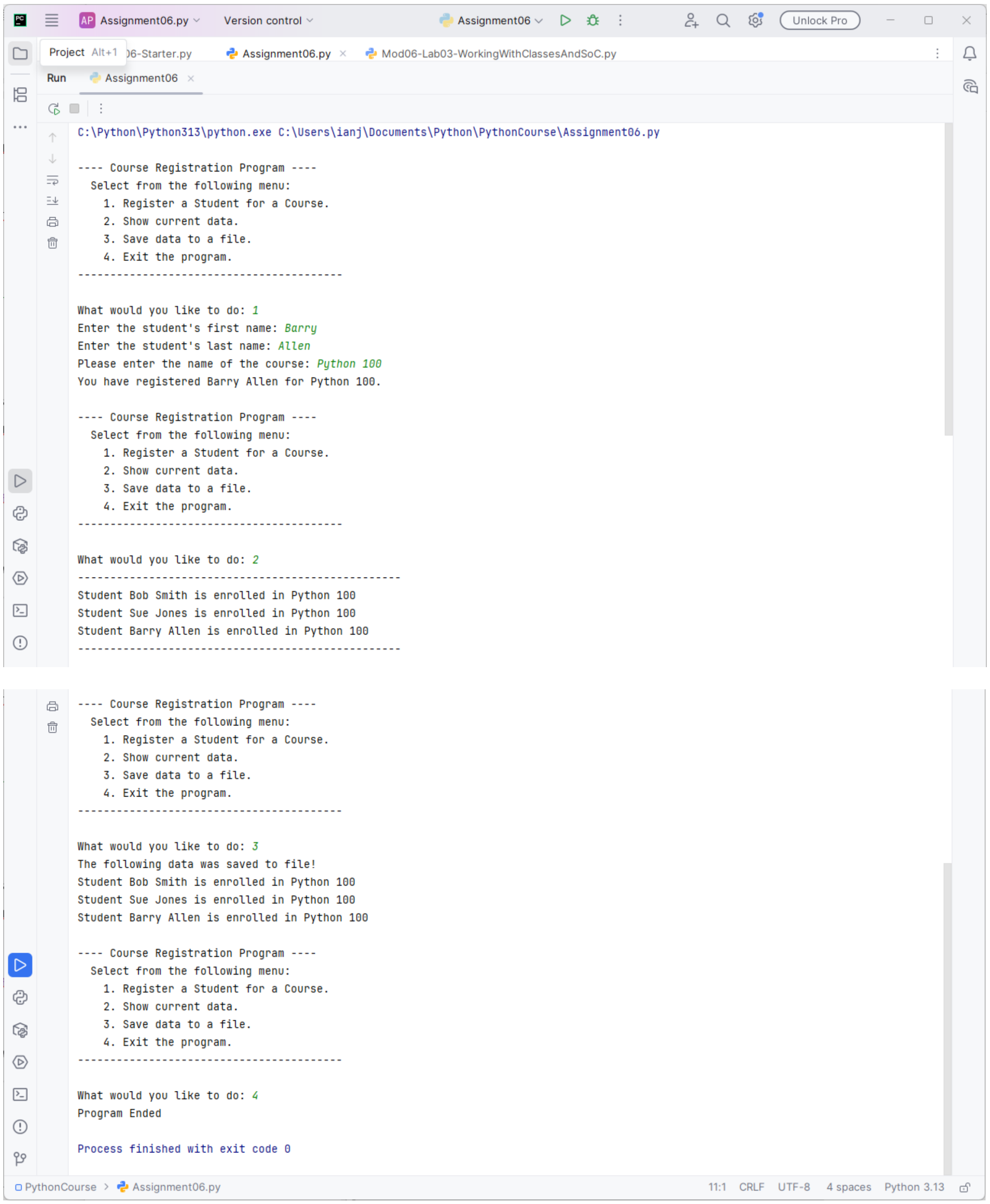
Figure 16 shows the write_data_to_file() function called out in the Main Body (menu choice 3).

```
# Save the data to a file
elif menu_choice == "3":
    FileProcessor.write_data_to_file(file_name=FILE_NAME, student_data=students)
    continue
```

Figure 16: output_student_courses() function called in Main Body

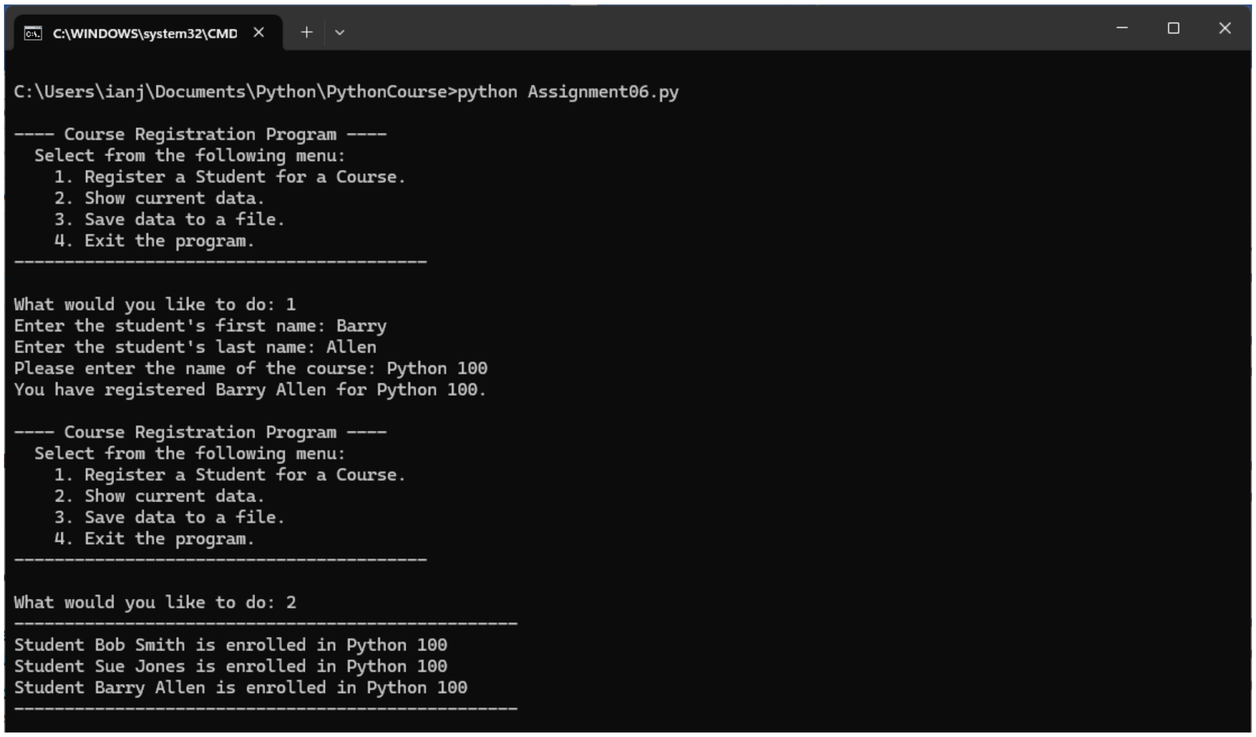
Testing

Once the script is written, the program is tested in PyCharm and from the Windows console. Test results are shown in Figure 17 for PyCharm, and Figure 18 for Windows console. Figure 19 shows the user’s input saved to the JSON file.

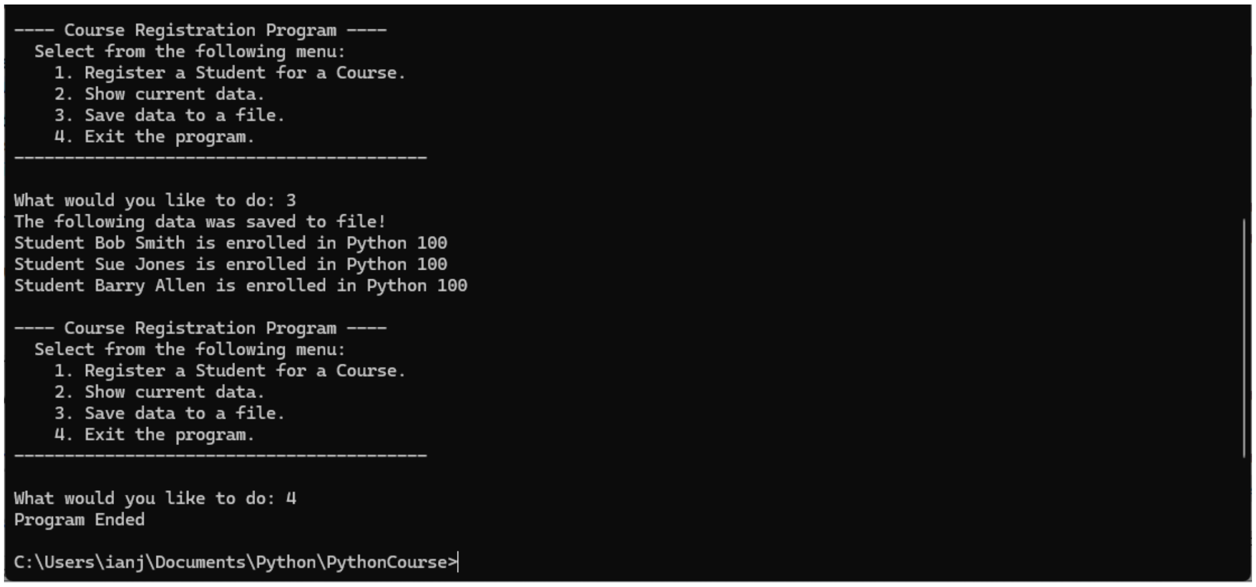


```
C:\Python\Python313\python.exe C:\Users\ianj\Documents\Python\PythonCourse\Assignment06.py

---- 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.
-----------------------------------------

What would you like to do: 1
Enter the student's first name: Barry
Enter the student's last name: Allen
Please enter the name of the course: Python 100
You have registered Barry Allen 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: 2
--------------------------------------------------
Student Bob Smith is enrolled in Python 100
Student Sue Jones is enrolled in Python 100
Student Barry Allen 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.
-----------------------------------------

What would you like to do: 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 Barry Allen 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.
-----------------------------------------

What would you like to do: 4
Program Ended

Process finished with exit code 0
```

Figure 17: Test in PyCharm

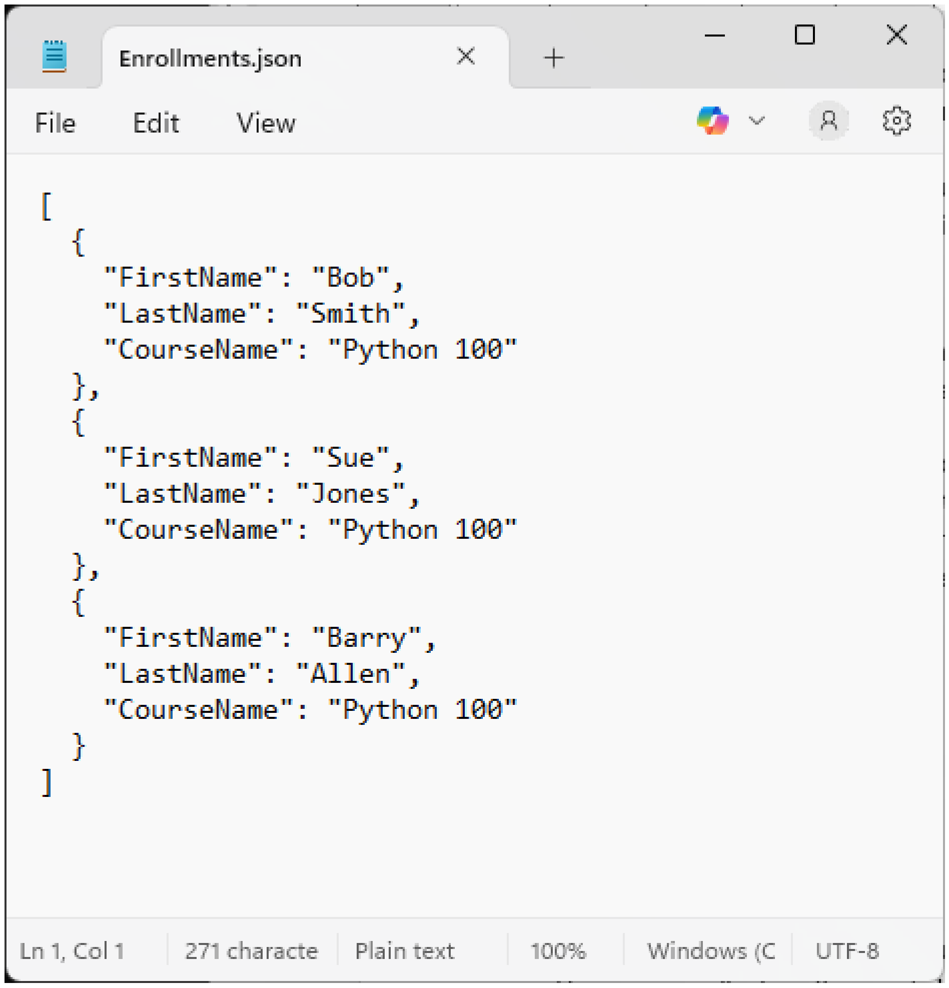


```
C:\Users\ianj\Documents\Python\PythonCourse>python Assignment06.py

---- 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.
-----------------------------------------

What would you like to do: 1
Enter the student's first name: Barry
Enter the student's last name: Allen
Please enter the name of the course: Python 100
You have registered Barry Allen 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: 2
--------------------------------------------------
Student Bob Smith is enrolled in Python 100
Student Sue Jones is enrolled in Python 100
Student Barry Allen 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.
-----------------------------------------

What would you like to do: 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 Barry Allen 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.
-----------------------------------------

What would you like to do: 4
Program Ended

C:\Users\ianj\Documents\Python\PythonCourse>
```

Figure 18: Test in Windows Console



```
[
  {
    "FirstName": "Bob",
    "LastName": "Smith",
    "CourseName": "Python 100"
  },
  {
    "FirstName": "Sue",
    "LastName": "Jones",
    "CourseName": "Python 100"
  },
  {
    "FirstName": "Barry",
    "LastName": "Allen",
    "CourseName": "Python 100"
  }
]
```

Figure 19: Test of Saving User's Input to JSON file

Summary

This document describes the steps performed in the creation of a Python program, “Assignment06.py”, that is similar to “Assignment05.py”, except that it additionally uses functions. Much of the program’s functionality and acceptance criteria is already accomplished by the starter file, “Assignment06-Starter.py”. The program satisfies the acceptance criteria by expanding upon the starter file to satisfy the remaining acceptance criteria.

The final completed Python script is shown in Figure 20.

```
# ------------------------------------------------------------------------------------------ #
# Title: Assignment06
# Desc: This assignment demonstrates using functions
# with structured error handling
# Change Log: (Who, When, What)
#   Ian Basques-Jellison, 8/17/2025, Created Script
#   <Your Name Here>, <Date>, <Activity>
# ------------------------------------------------------------------------------------------ #
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: str = "Enrollments.csv"
FILE_NAME: str = "Enrollments.json"

# Define the Data Variables and constants
# 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
# file = None                   # Holds a reference to an opened file.
menu_choice: str = ''           # Hold the choice made by the user.


# Processing ------------------------------------------------------------------------ #
class FileProcessor:
    """
    A collection of processing layer functions for Json files

    ChangeLog: (Who, When, What)
    Ian Basques-Jellison, 8/17/2025, Created Class
    """

    @staticmethod
    def read_data_from_file(file_name: str, student_data: list):
        """ This function reads data from a json file

        ChangeLog: (Who, When, What)
        Ian Basques-Jellison, 8/17/2025, Created Function

        :return: a table of student data
        """
        try:
            file = open(file_name, "r")
            student_data = json.load(file)
            file.close()
        except FileNotFoundError as e:
            IO.output_error_messages("Please check that the file exists " \
                                     "and that it is in a json format.", e)
        except Exception as e:
            IO.output_error_messages("There was a non-specific error " \
                                     "reading the file.", e)
        finally:
            if file.closed == False:
                file.close()
        return student_data

    @staticmethod
    def write_data_to_file(file_name: str, student_data: list):
        """ This function writes data to a json file

        ChangeLog: (Who, When, What)
        Ian Basques-Jellison, 8/17/2025, Created Function

        :return: None
        """
        try:
            file = open(file_name, "w")
            json.dump(student_data, file, indent=2)

            file.close()
            print("The following data was saved to file!")
            for student in student_data:
                print(f'Student {student["FirstName"]} '
                      f'{student["LastName"]} is enrolled in {student["CourseName"]}')
        except Exception as e:
            if file.closed == False:
                file.close()
            IO.output_error_messages("Error: There was a problem with writing " /
                                     "to the file.\n" /
                                     "Please check that the file is not open " /
                                     "by another program.", e)


# Presentation ----------------------------------------------------------------------- #
class IO:
    """
    A collection of presentation layer functions for user input and output

    ChangeLog: (Who, When, What)
    Ian Basques-Jellison, 8/17/2025, Created Class
    """

    @staticmethod
    def output_error_messages(message: str, error: Exception = None):
        """ This function displays error messages to the user

        ChangeLog: (Who, When, What)
        Ian Basques-Jellison, 8/17/2025, Created Function

        :return: None
        """
        print(message, end="\n\n")
        if error is not None:
            print("-- Technical Error Message -- ")
            print(error, error.__doc__, type(error), sep='\n')

    @staticmethod
    def output_menu(menu: str):
        """ This function displays the menu to the user

        ChangeLog: (Who, When, What)
        Ian Basques-Jellison, 8/17/2025, Created Function

        :return: None
        """
        print(menu)

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

        ChangeLog: (Who, When, What)
        Ian Basques-Jellison, 8/17/2025, Created Function
        :return: string with the user's choice
        """
        choice = "0"
        try:
            choice = input("What would you like to do: ")
            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 the exception object
                                                   # to avoid the technical message
        return choice

    @staticmethod
    def input_student_data(student_data: list):
        """ This function gets first name, last name, and course from the user

        ChangeLog: (Who, When, What)
        Ian Basques-Jellison, 8/17/2025, Created Function

        :return: str
        """
        try:
            student_first_name = input("Enter the student's first name: ")
            if not student_first_name.isalpha():
                raise ValueError("The first 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 = {"FirstName": student_first_name,
                       "LastName": student_last_name,
                       "CourseName": course_name}
            student_data.append(student)
            print(f"You have registered {student_first_name} {student_last_name} " \
                  f"for {course_name}.")
        except ValueError as e:
            IO.output_error_messages(f"{e}", e)
        except Exception as e:
            IO.output_error_messages("Error: There was a problem with " \
                                     "your entered data.", e)
        return student_data

    @staticmethod
    def output_student_courses(student_data: list):
        """ This function displays all the registered students and their courses

        ChangeLog: (Who, When, What)
        Ian Basques-Jellison, 8/17/2025, Created Function

        :return: None
        """
        # Process the data to create and display a custom message
        print("-" * 50)
        for student in student_data:
            print(f'Student {student["FirstName"]} '
                  f'{student["LastName"]} is enrolled in {student["CourseName"]}')
        print("-" * 50)

# End of function definitions


# ------------------------------------ MAIN BODY ------------------------------------- #
# 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=FILE_NAME, student_data=students)

# Present and Process the data
while (True):
    # Present the menu of choices
    IO.output_menu(menu=MENU)
    menu_choice = IO.input_menu_choice()

    # Input user data
    if menu_choice == "1":  # This will not work if it is an integer!
        students = IO.input_student_data(student_data=students)
        continue

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

    # Save the data to a file
    elif menu_choice == "3":
        FileProcessor.write_data_to_file(file_name=FILE_NAME, student_data=students)
        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")
```

Figure 20: Complete Python Script for Assignment06.py