Adnan Dawood

June 12, 2024

Foundations of Programming: Python

Assignment 08

Creating Applications

Introduction:

1. The Employee Ratings Application is a program designed to manage and track employee data, including employee names, review dates, and review ratings. This application allows users to view current employee data, input new employee data, save data to a file, and exit the program.

2. Application Components:

The application consists of several Python modules:

data_classes.py: Defines the data classes used in the application, including Person and Employee.

main.py: Contains the main program logic and user interface.

presentation_classes.py: Handles input and output tasks for the user interface.

processing_classes.py: Processes data to and from a file.

test_data_classes.py: Unit tests for the data classes.

test_presentation_classes.py: Unit tests for the presentation classes.

test_processing_classes.py: Unit tests for the processing classes.

3. Data Classes:

The data_classes.py module defines two main classes:

Person: Represents a person with attributes for first name and last name.

Employee: Inherits from Person and includes additional attributes for review date and review rating.

4. Main Program

The main.py module contains the main program logic, including the main function and menu options:

Show current employee rating data: Displays all current employee data.

Enter new employee rating data: Allows the user to input data for a new employee.

Save data to a file: Writes employee data to a JSON file.

Exit the program: Exits the application.

5. Presentation Classes

The presentation_classes.py module handles input and output tasks for the user interface:

output_menu: Displays the main menu.

input_menu_choice: Collects the user's choice from the menu.

output_employee_data: Displays all employee data.

input_employee_data: Collects data for a new employee from the user.

6. Processing Classes

The processing_classes.py module processes data to and from a file:

read_employee_data_from_file: Reads employee data from a JSON file and populates a list of Employee objects.

write_employee_data_to_file: Writes employee data to a JSON file from a list of Employee objects.

7. Testing

The application includes unit tests for each module:

test_data_classes.py: Tests for the data classes.

test_presentation_classes.py: Tests for the presentation classes.

test_processing_classes.py: Tests for the processing classes.

8. Usage

To use the application, the user is to run the main.py module, follow the prompts to view, input, and save employee data and use the menu options to navigate through the application.

This document provides an overview of the Employee Ratings Application, its components, and functionality.

Writing the program and the findings:

a- Module: data_classes.py

1. Introduction

The Person and Employee classes are part of a Python module designed to represent individuals and employees within an organization. These classes provide a structured way to manage personal and employment-related information.

2. Person Class

The Person class represents a generic individual with the following attributes:

- **first_name (str)**: The person's first name, defaults to an empty string.
- **last_name (str)**: The person's last name, defaults to an empty string.

Constructor:

```
def __init__(self, first_name: str = "", last_name: str = ""):
  self.first_name = first_name
  self.last_name = last_name
```

Properties:

- **first_name**: Retrieves the first name of the person.
- **last_name**: Retrieves the last name of the person.

Methods:

- **str**: Returns a string representation of the person's full name.

3. Employee Class

The Employee class inherits from the Person class and extends it with additional attributes related to employment:

- **review_date (date)**: The date of the employee's review, defaults to January 1, 1900.
- **review_rating (int)**: The employee's review rating, defaults to 3.

Constructor:

```
def __init__(self, first_name: str = "", last_name: str = "", review_date: str = "1900-01-01",
review_rating: int = 3):
  super().__init__(first_name, last_name)
  self._review_date = date.fromisoformat(str(review_date))

  self.review_rating = review_rating
```

Properties:

- **review_date**: Retrieves the date of the employee's review.
- **review_rating**: Retrieves the review rating of the employee.

Methods:

- **str**: Returns a string representation of the employee's full name, review date, and review rating.

4. Usage

To use the Person and Employee classes, import the module containing these classes into your Python script. Then, create instances of these classes to represent individuals and employees, setting appropriate attributes as needed.

Example usage:

```
from datetime import date

from person_employee_module import Employee


# Create an employee instance

employee = Employee("John", "Doe", "2023-06-10", 4)


# Accessing attributes

print(employee.first_name)  # Output: John

print(employee.last_name)   # Output: Doe

print(employee.review_date) # Output: 2023-06-10

print(employee.review_rating) # Output: 4


# Printing employee details

print(employee) # Output: John Doe, Review Date: 2023-06-10, Review Rating: 4
```

This provides an overview of the Person and Employee classes, their attributes, methods, and usage examples.

b- <u>Module: main.py</u>

1. Introduction

The Employee Ratings Management module is a Python script designed to facilitate the management of employee rating data. It provides functionality to display, input, and save employee ratings to a file.

2. Components

The module consists of the following components:

- **IO Class**: Handles input and output tasks for the user interface.
- **FileProcessor Class**: Processes data to and from a file.
- **Employee Class**: Represents employee data.
- **Main Function**: Contains the main program logic.

3. Main Function

The main() function serves as the entry point to the program. It orchestrates the execution flow and user interactions:

- **Initialization**: Initializes an empty list of employees.
- **File Loading**: Attempts to read employee data from a JSON file. If the file does not exist, starts with an empty dataset.
- **User Interaction Loop**: Displays a menu to the user and prompts for input until the user chooses to exit.
- **Menu Options**:
    1. **Show current employee rating data**: Displays all current employee data.
    2. **Enter new employee rating data**: Allows the user to input data for a new employee.
    3. **Save data to a file**: Writes employee data to a JSON file.
    4. **Exit the program**: Exits the application.

4. Usage

To use the Employee Ratings Management module, ensure that the required classes (IO, FileProcessor, Employee) are imported correctly. Modify the FILE_NAME variable to specify the filename for storing employee data. Run the script to interact with the program.

Example usage:

```
python main.py
```

The user is to follow the prompts in the console to navigate through the menu options and manage employee rating data.

This provides an overview of the Employee Ratings Management module, its components, functionality, and usage instructions.

c- <u>Module: presentation_classes.py</u>

1. Introduction

The User Interface Input and Output (IO) module is a Python script responsible for handling input and output tasks for the user interface of an application. It provides methods to display menus, collect user input, and present employee data to the user.

2. Class: IO

The IO class contains static methods to handle input and output operations for the application's user interface.

3. Methods:

- **output_menu(menu: str) -> None**: Displays the main menu to the user.

  Args:

  - **menu (str)**: The menu string to be displayed.
- **input_menu_choice() -> str**: Collects the user's choice from the menu.

  Returns:

  - **str**: The user's menu choice.
- **output_employee_data(employee_data: list) -> None**: Displays all employee data to the user.

  Args:

  - **employee_data (list)**: A list of Employee objects.
- **input_employee_data() -> 'Employee'**: Prompts the user to enter data for a new employee.

  Returns:

  - **Employee**: The new Employee object created from the user input.

4. Usage

To use the User Interface IO module, ensure that the required Employee class from the data_classes module is imported correctly. Then, call the methods of the IO class to interact with the user.

Example usage:

```
from data_classes import Employee

from datetime import datetime

from io_module import IO


# Display the main menu

IO.output_menu("Main Menu")


# Collect the user's menu choice

choice = IO.input_menu_choice()

print(f"User selected: {choice}")


# Display employee data

employees = [Employee("John", "Doe", datetime.now(), 4), Employee("Jane", "Smith",
datetime.now(), 5)]

IO.output_employee_data(employees)


# Collect data for a new employee

new_employee = IO.input_employee_data()

print(f"New employee: {new_employee}")
```

This provides an overview of the User Interface Input and Output (IO) module, its methods, and usage instructions.

d- Module: processing_classes.py

1. Introduction

The File Processor module is a Python script designed to handle the processing of data to and from a file. It provides methods to read employee data from a JSON file and write employee data to a JSON file.

2. Class: FileProcessor

The FileProcessor class contains static methods to process data to and from a file.

3. Methods:

- **read_employee_data_from_file(file_name: str, employee_data: list) -> None**: Reads employee data from a JSON file and populates a list of Employee objects.

  Args:

  - **file_name (str)**: The name of the file to read from.
  - **employee_data (list)**: The list to populate with Employee objects.

  Raises:

  - **FileNotFoundError**: If the specified file does not exist.
  - **json.JSONDecodeError**: If the file is not valid JSON.
- **write_employee_data_to_file(file_name: str, employee_data: list) -> None**: Writes employee data to a JSON file from a list of Employee objects.

  Args:

  - **file_name (str)**: The name of the file to write to.
  - **employee_data (list)**: The list of Employee objects to write.

  Raises:

  - **PermissionError**: If the file cannot be written to.

4. Usage

To use the File Processor module, ensure that the required Employee class from the data_classes module is imported correctly. Then, call the methods of the FileProcessor class to read from or write to a JSON file.

Example usage:

```
import json

from data_classes import Employee

from file_processor_module import FileProcessor


# Read employee data from a file

employees = []

FileProcessor.read_employee_data_from_file("employee_data.json", employees)

print("Employee data loaded:", employees)


# Write employee data to a file

new_employee_data = [Employee("John", "Doe", "2022-01-01", 4), Employee("Jane", "Smith", "2023-01-01", 5)]

FileProcessor.write_employee_data_to_file("new_employee_data.json", new_employee_data)

print("New employee data written to file.")
```

This provides an overview of the File Processor module, its methods, and usage instructions.

e- Module: test_data_classes.py

1. Introduction

The Unit Testing module is a Python script designed to perform unit tests on the Person and Employee classes from the data_classes module. It ensures that the classes' functionalities, including object creation, inheritance, and input validation, work as expected.

2. Class: TestPerson

This class contains test cases to verify the functionalities of the Person class.

- **test_person_creation**: Tests whether a Person object can be created successfully with valid input parameters (first_name and last_name).
- **test_person_name_validation**: Tests whether the Person class correctly raises a ValueError when invalid characters are used in the first_name parameter during object creation.

3. Class: TestEmployee

This class contains test cases to verify the functionalities of the Employee class, which inherits from the Person class.

- **test_employee_inheritance**: Tests whether an Employee object can be created successfully with valid input parameters (first_name, last_name, review_date, and review_rating). It also checks if the Employee object inherits the attributes from the Person class correctly.
- **test_review_date_validation**: Tests whether the Employee class correctly raises a ValueError when an invalid date format is used in the review_date parameter during object creation.
- **test_review_rating_validation**: Tests whether the Employee class correctly raises a ValueError when an invalid value is used in the review_rating parameter during object creation.

4. Usage

To use the Unit Testing module, ensure that the required classes (Person, Employee) from the data_classes module are imported correctly. Run the script to execute the unit tests.

Example usage:

```
python test_module.py
```

The output will indicate whether the tests passed or failed, along with any relevant error messages.

This provides an overview of the Unit Testing module, its test cases, and usage instructions.

f- Module: test_presentation_classes.py

1. Introduction

The Unit Testing module is a Python script designed to perform unit tests on the IO class from the presentation_classes module. It ensures that the input and output functionalities of the user interface work as expected.

2. Class: TestIO

The module contains a test class, TestIO, responsible for testing the functionalities of the IO class.

3. Test Cases:

- **test_output_menu**: Tests the output_menu method of the IO class. It verifies whether the method correctly displays the menu to the user.
- **test_input_menu_choice**: Tests the input_menu_choice method of the IO class. It verifies whether the method correctly collects the user's choice from the menu.
- **test_output_employee_data**: Tests the output_employee_data method of the IO class. It verifies whether the method correctly displays employee data to the user.
- **test_input_employee_data**: Tests the input_employee_data method of the IO class. It verifies whether the method correctly collects data for a new employee from the user.

4. Usage

To use the Unit Testing module, ensure that the required classes (IO, Employee) from the respective modules are imported correctly. Run the script to execute the unit tests.

Example usage:

```
python test_module.py
```

The output will indicate whether the tests passed or failed, along with any relevant error messages.

This provides an overview of the Unit Testing module, its test cases, and usage instructions. For more details on individual test cases and their implementations, refer to the class definition within the module.

g- Module: test_processing_classes.py

1. Introduction

The Unit Testing module is a Python script designed to perform unit tests on the FileProcessor class from the processing_classes module. It ensures that the methods for reading and writing employee data to a file work as expected.

2. Class: TestFileProcessor

The module contains a test class, TestFileProcessor, responsible for testing the functionalities of the FileProcessor class.

3. Test Cases:

- **test_read_employee_data_from_file**: Tests the read_employee_data_from_file method of the FileProcessor class. It verifies whether the method correctly reads employee data from a JSON file and populates a list of Employee objects.
- **test_write_employee_data_to_file**: Tests the write_employee_data_to_file method of the FileProcessor class. It verifies whether the method correctly writes employee data to a JSON file from a list of Employee objects.

4. Usage

To use the Unit Testing module, ensure that the required classes (FileProcessor, Employee) from the respective modules are imported correctly. Run the script to execute the unit tests.

Example usage:

```
python test_module.py
```

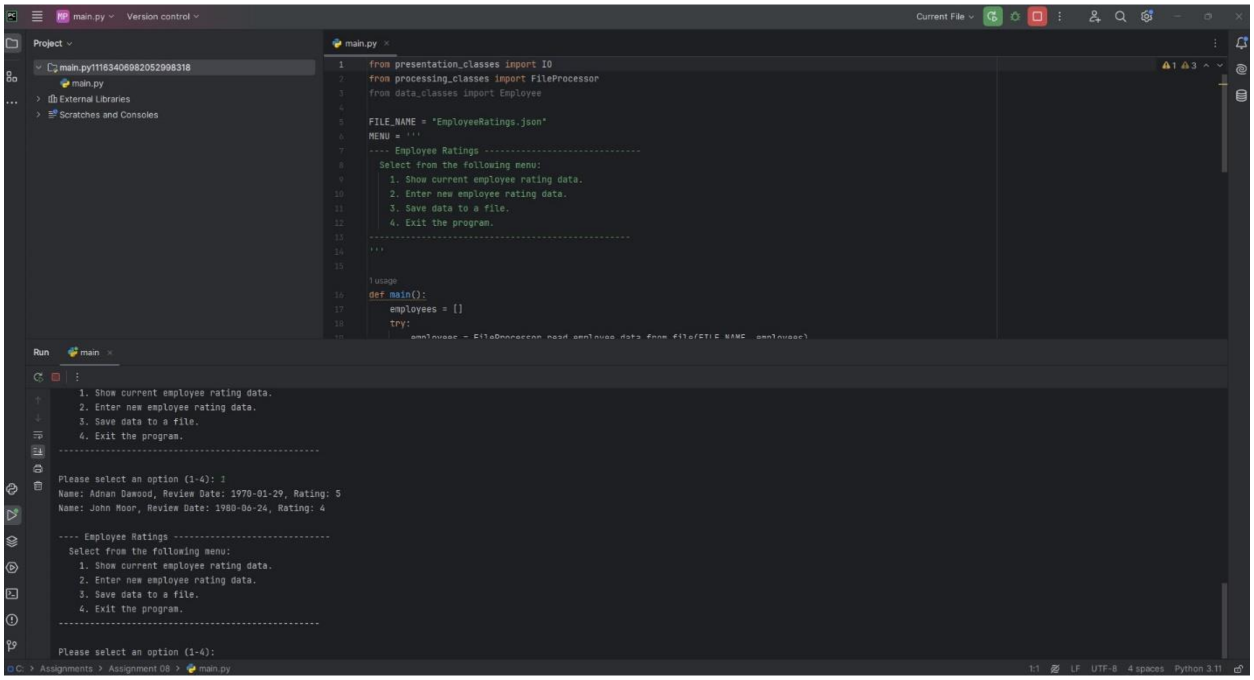
The output will indicate whether the tests passed or failed, along with any relevant error messages.

This provides an overview of the Unit Testing module, its test cases, and usage instructions.

I used PyCharm to write my script and used PyCharm and Terminal to execute it.

a- Here is how my first script looks like and its output in PyCharm:

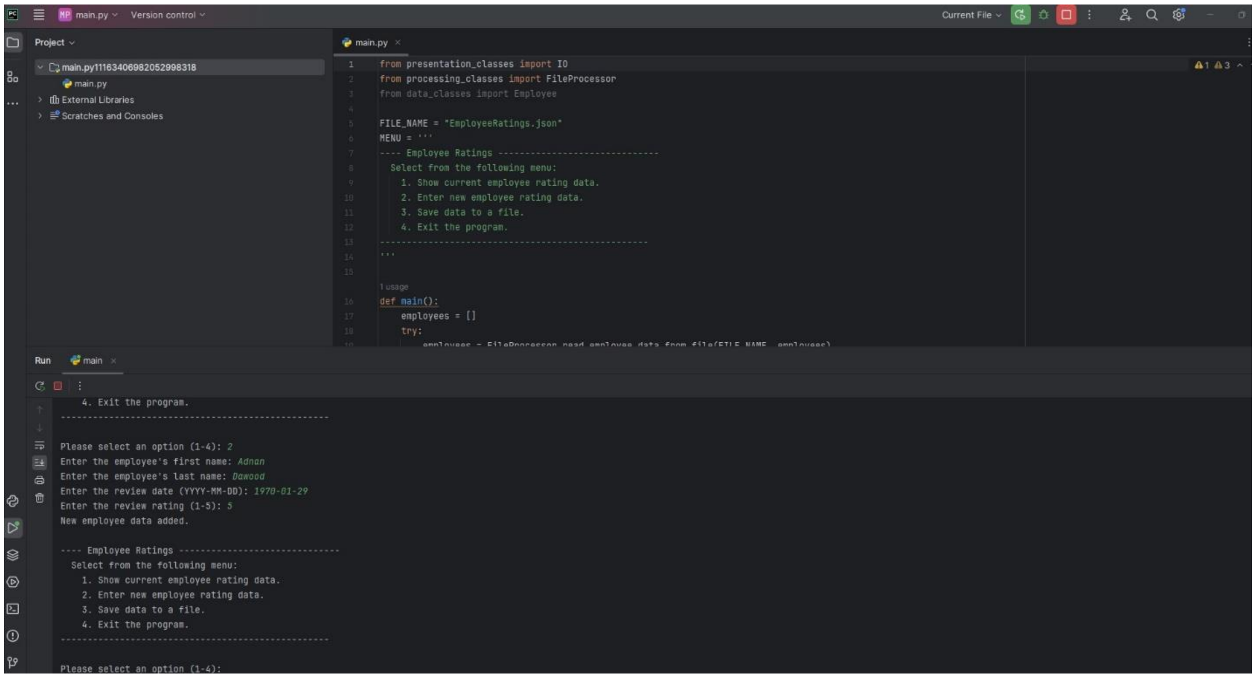
1- Menu option 1 will show the current employee rating data as shown below:



```python
from presentation_classes import IO
from processing_classes import FileProcessor
from data_classes import Employee

FILE_NAME = "EmployeeRatings.json"
MENU = '''
---- Employee Ratings ------------------------------
  Select from the following menu:
    1. Show current employee rating data.
    2. Enter new employee rating data.
    3. Save data to a file.
    4. Exit the program.
--------------------------------------------------
'''

def main():
    employees = []
    try:
        employees = FileProcessor.read_employee_data_from_file(FILE_NAME, employees)
```

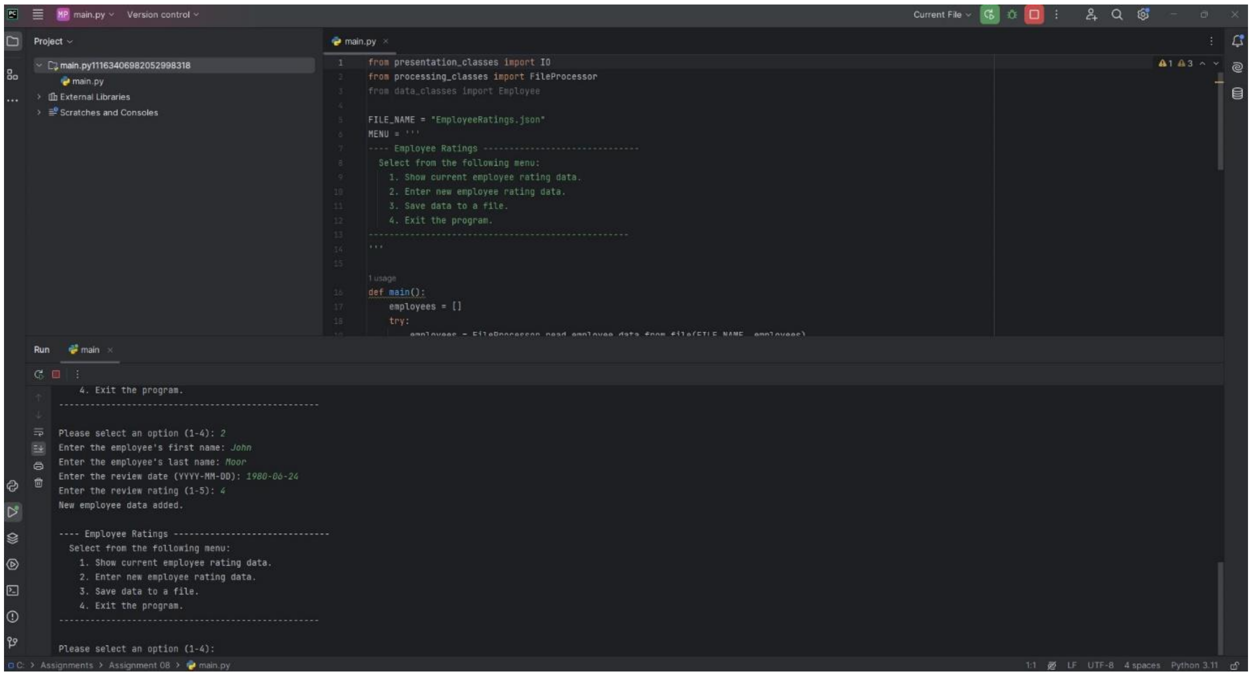
```
    1. Show current employee rating data.
    2. Enter new employee rating data.
    3. Save data to a file.
    4. Exit the program.
--------------------------------------------------

Please select an option (1-4): 1
Name: Adnan Dawood, Review Date: 1970-01-29, Rating: 5
Name: John Moor, Review Date: 1980-06-24, Rating: 4

---- Employee Ratings ------------------------------
  Select from the following menu:
    1. Show current employee rating data.
    2. Enter new employee rating data.
    3. Save data to a file.
    4. Exit the program.
--------------------------------------------------

Please select an option (1-4):
```

2- Menu option 2 will enter new employee rating data as shown below:



```python
from presentation_classes import IO
from processing_classes import FileProcessor
from data_classes import Employee

FILE_NAME = "EmployeeRatings.json"
MENU = '''
---- Employee Ratings ------------------------------
  Select from the following menu:
    1. Show current employee rating data.
    2. Enter new employee rating data.
    3. Save data to a file.
    4. Exit the program.
--------------------------------------------------
'''

def main():
    employees = []
    try:
        employees = FileProcessor.read_employee_data_from_file(FILE_NAME, employees)
```

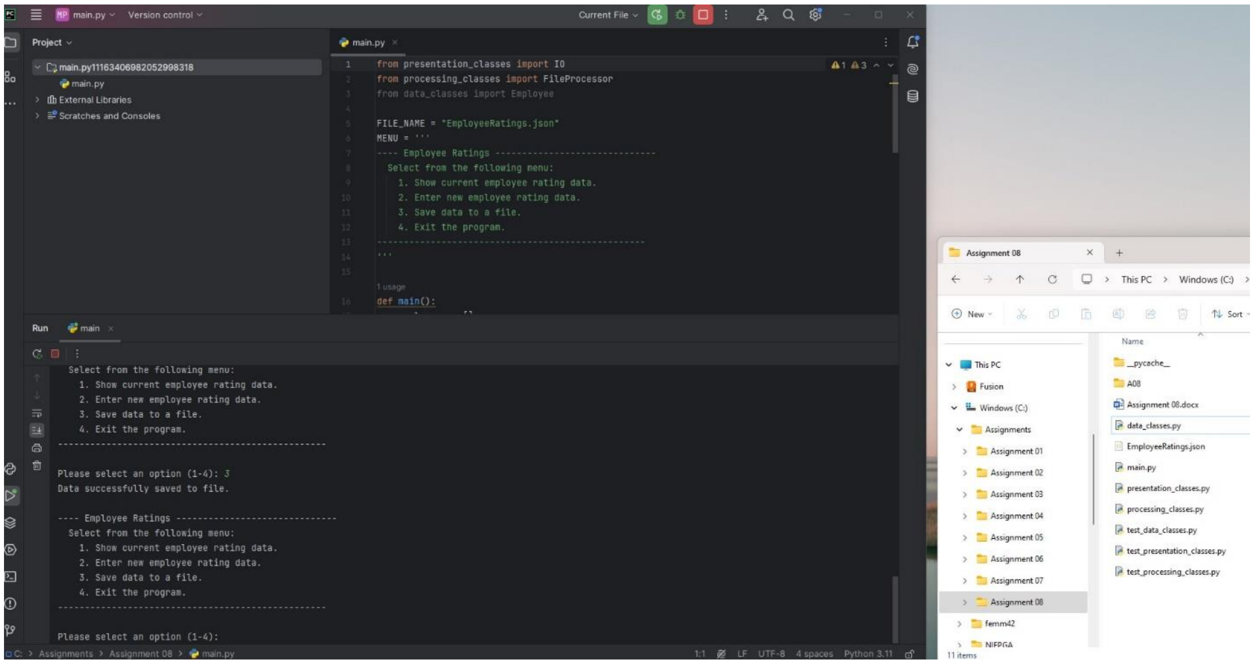
```
    4. Exit the program.
--------------------------------------------------

Please select an option (1-4): 2
Enter the employee's first name: Adnan
Enter the employee's last name: Dawood
Enter the review date (YYYY-MM-DD): 1970-01-29
Enter the review rating (1-5): 5
New employee data added.

---- Employee Ratings ------------------------------
  Select from the following menu:
    1. Show current employee rating data.
    2. Enter new employee rating data.
    3. Save data to a file.
    4. Exit the program.
--------------------------------------------------

Please select an option (1-4):
```



```python
from presentation_classes import IO
from processing_classes import FileProcessor
from data_classes import Employee

FILE_NAME = "EmployeeRatings.json"
MENU = '''
---- Employee Ratings ------------------------------
  Select from the following menu:
    1. Show current employee rating data.
    2. Enter new employee rating data.
    3. Save data to a file.
    4. Exit the program.
--------------------------------------------------
'''

def main():
    employees = []
    try:
        employees = FileProcessor.read_employee_data_from_file(FILE_NAME, employees)
```

```
    4. Exit the program.
--------------------------------------------------

Please select an option (1-4): 2
Enter the employee's first name: John
Enter the employee's last name: Moor
Enter the review date (YYYY-MM-DD): 1980-06-24
Enter the review rating (1-5): 4
New employee data added.

---- Employee Ratings ------------------------------
  Select from the following menu:
    1. Show current employee rating data.
    2. Enter new employee rating data.
    3. Save data to a file.
    4. Exit the program.
--------------------------------------------------

Please select an option (1-4):
```

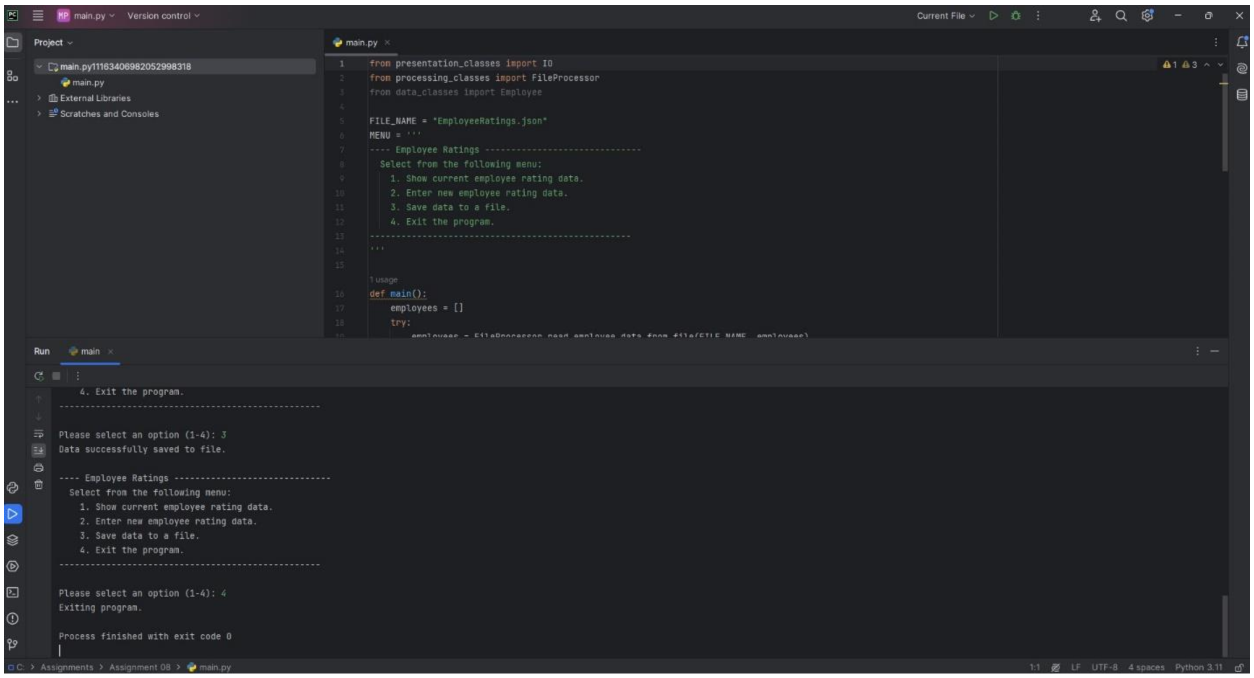
3- Menu option 3 will save data to a file as shown below:



```python
from presentation_classes import IO
from processing_classes import FileProcessor
from data_classes import Employee

FILE_NAME = "EmployeeRatings.json"
MENU = '''
---- Employee Ratings ------------------------------
  Select from the following menu:
    1. Show current employee rating data.
    2. Enter new employee rating data.
    3. Save data to a file.
    4. Exit the program.
--------------------------------------------------
'''

def main():
```

```
  Select from the following menu:
    1. Show current employee rating data.
    2. Enter new employee rating data.
    3. Save data to a file.
    4. Exit the program.
--------------------------------------------------

Please select an option (1-4): 3
Data successfully saved to file.

---- Employee Ratings ------------------------------
  Select from the following menu:
    1. Show current employee rating data.
    2. Enter new employee rating data.
    3. Save data to a file.
    4. Exit the program.
--------------------------------------------------

Please select an option (1-4):
```

4- Menu option 4 will exit the program as shown below:

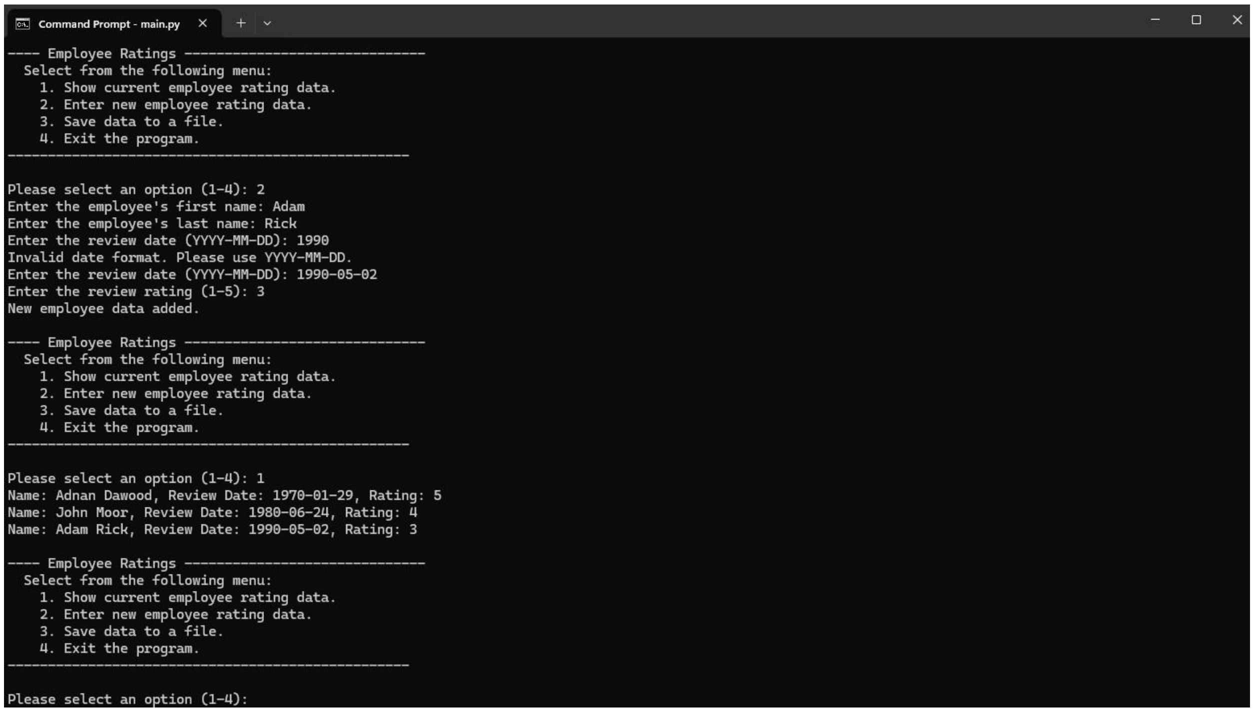


```python
from presentation_classes import IO
from processing_classes import FileProcessor
from data_classes import Employee

FILE_NAME = "EmployeeRatings.json"
MENU = '''
---- Employee Ratings ------------------------------
  Select from the following menu:
    1. Show current employee rating data.
    2. Enter new employee rating data.
    3. Save data to a file.
    4. Exit the program.
--------------------------------------------------
'''

def main():
    employees = []
    try:
```

```
    4. Exit the program.
--------------------------------------------------

Please select an option (1-4): 3
Data successfully saved to file.

---- Employee Ratings ------------------------------
  Select from the following menu:
    1. Show current employee rating data.
    2. Enter new employee rating data.
    3. Save data to a file.
    4. Exit the program.
--------------------------------------------------

Please select an option (1-4): 4
Exiting program.

Process finished with exit code 0
```

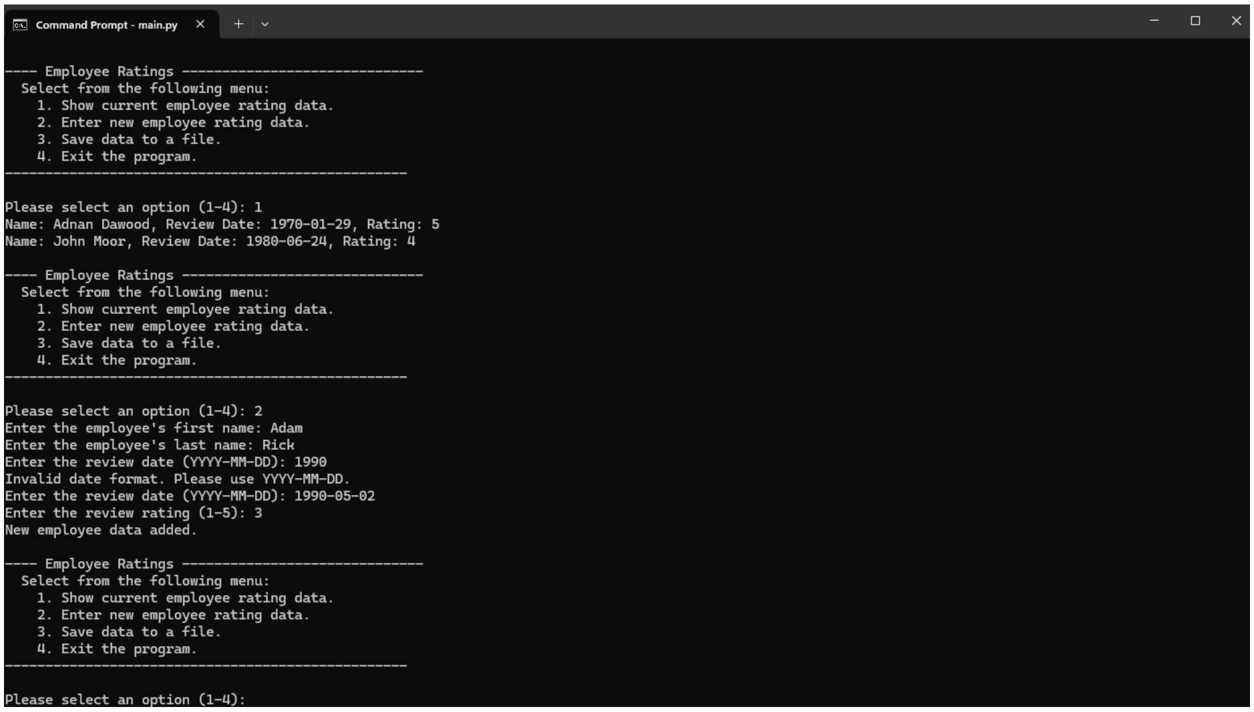
b- Here is how my first script looks like and its output in terminal:

1- Menu option 1 will show the current employee rating data as shown below:



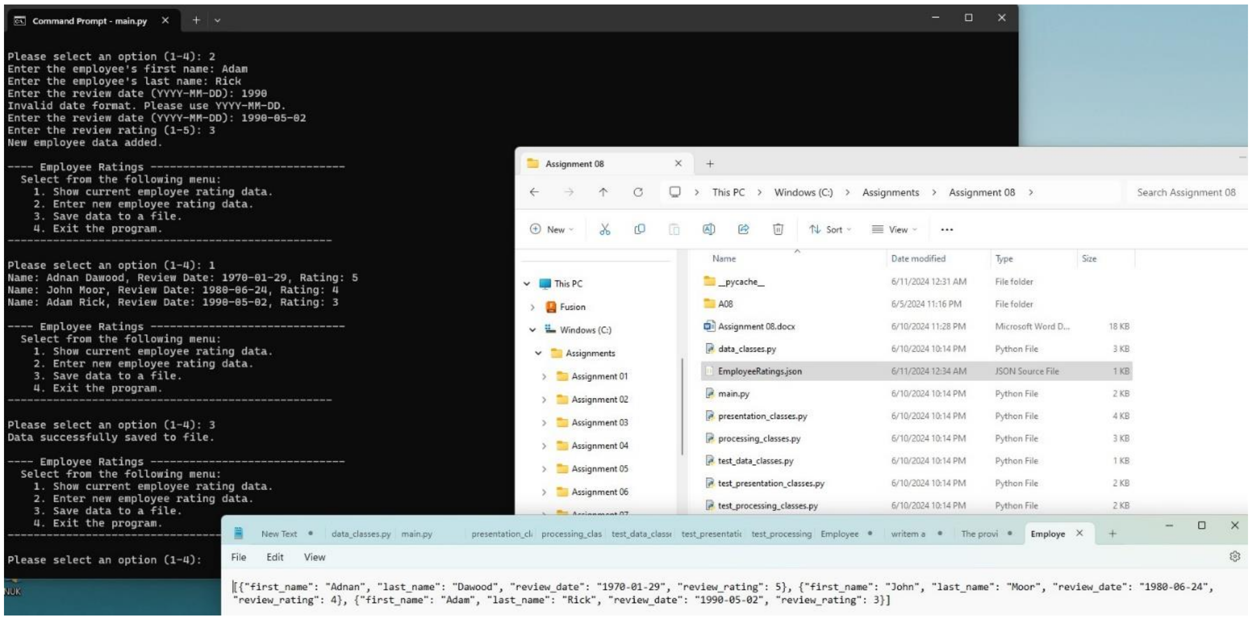
```
---- Employee Ratings ------------------------------
  Select from the following menu:
    1. Show current employee rating data.
    2. Enter new employee rating data.
    3. Save data to a file.
    4. Exit the program.
--------------------------------------------------

Please select an option (1-4): 2
Enter the employee's first name: Adam
Enter the employee's last name: Rick
Enter the review date (YYYY-MM-DD): 1990
Invalid date format. Please use YYYY-MM-DD.
Enter the review date (YYYY-MM-DD): 1990-05-02
Enter the review rating (1-5): 3
New employee data added.

---- Employee Ratings ------------------------------
  Select from the following menu:
    1. Show current employee rating data.
    2. Enter new employee rating data.
    3. Save data to a file.
    4. Exit the program.
--------------------------------------------------

Please select an option (1-4): 1
Name: Adnan Dawood, Review Date: 1970-01-29, Rating: 5
Name: John Moor, Review Date: 1980-06-24, Rating: 4
Name: Adam Rick, Review Date: 1990-05-02, Rating: 3

---- Employee Ratings ------------------------------
  Select from the following menu:
    1. Show current employee rating data.
    2. Enter new employee rating data.
    3. Save data to a file.
    4. Exit the program.
--------------------------------------------------

Please select an option (1-4):
```

2- Menu option 2 will enter new employee rating data as shown below:



```
---- Employee Ratings ------------------------------
  Select from the following menu:
    1. Show current employee rating data.
    2. Enter new employee rating data.
    3. Save data to a file.
    4. Exit the program.
--------------------------------------------------

Please select an option (1-4): 1
Name: Adnan Dawood, Review Date: 1970-01-29, Rating: 5
Name: John Moor, Review Date: 1980-06-24, Rating: 4

---- Employee Ratings ------------------------------
  Select from the following menu:
    1. Show current employee rating data.
    2. Enter new employee rating data.
    3. Save data to a file.
    4. Exit the program.
--------------------------------------------------

Please select an option (1-4): 2
Enter the employee's first name: Adam
Enter the employee's last name: Rick
Enter the review date (YYYY-MM-DD): 1990
Invalid date format. Please use YYYY-MM-DD.
Enter the review date (YYYY-MM-DD): 1990-05-02
Enter the review rating (1-5): 3
New employee data added.

---- Employee Ratings ------------------------------
  Select from the following menu:
    1. Show current employee rating data.
    2. Enter new employee rating data.
    3. Save data to a file.
    4. Exit the program.
--------------------------------------------------

Please select an option (1-4):
```

3- Menu option 3 will save data to a file as shown below:



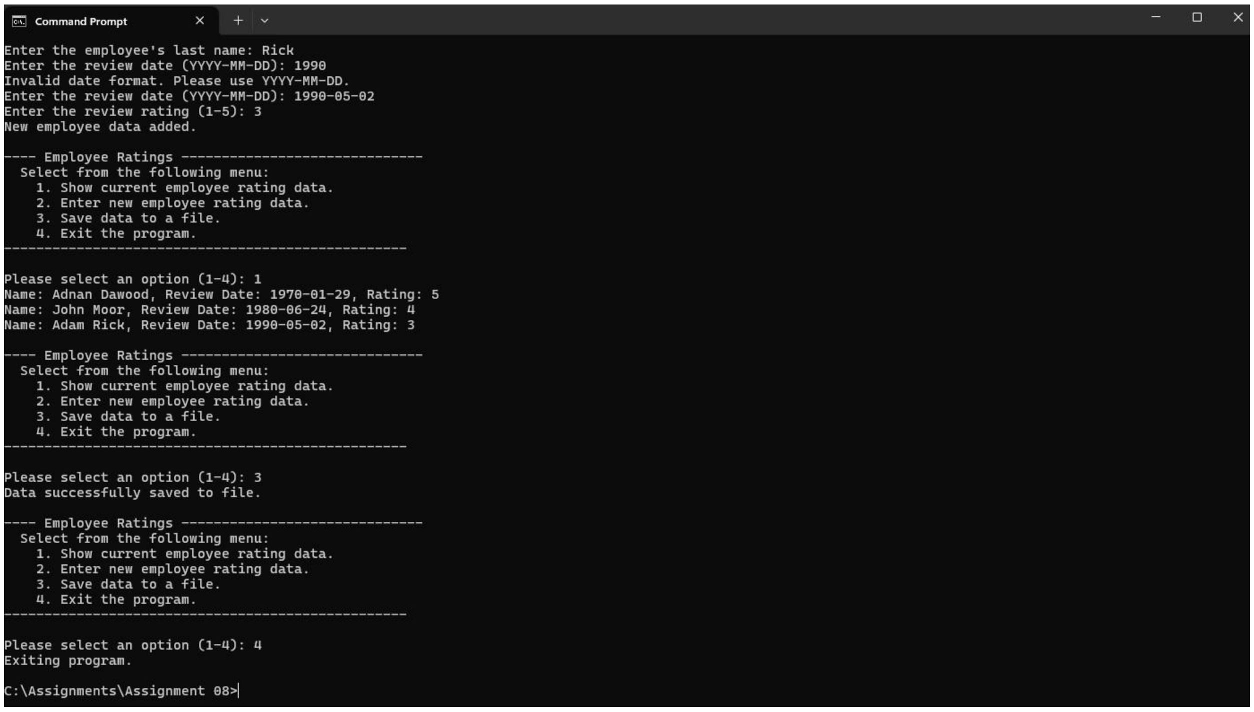
```
Please select an option (1-4): 2
Enter the employee's first name: Adam
Enter the employee's last name: Rick
Enter the review date (YYYY-MM-DD): 1990
Invalid date format. Please use YYYY-MM-DD.
Enter the review date (YYYY-MM-DD): 1990-05-02
Enter the review rating (1-5): 3
New employee data added.

---- Employee Ratings ------------------------------
  Select from the following menu:
    1. Show current employee rating data.
    2. Enter new employee rating data.
    3. Save data to a file.
    4. Exit the program.
--------------------------------------------------

Please select an option (1-4): 1
Name: Adnan Dawood, Review Date: 1970-01-29, Rating: 5
Name: John Moor, Review Date: 1980-06-24, Rating: 4
Name: Adam Rick, Review Date: 1990-05-02, Rating: 3

---- Employee Ratings ------------------------------
  Select from the following menu:
    1. Show current employee rating data.
    2. Enter new employee rating data.
    3. Save data to a file.
    4. Exit the program.
--------------------------------------------------

Please select an option (1-4): 3
Data successfully saved to file.

---- Employee Ratings ------------------------------
  Select from the following menu:
    1. Show current employee rating data.
    2. Enter new employee rating data.
    3. Save data to a file.
    4. Exit the program.
--------------------------------------------------

Please select an option (1-4):
```

This PC > Windows (C:) > Assignments > Assignment 08 >

Name	Date modified	Type	Size
__pycache__	6/11/2024 12:31 AM	File folder	
A08	6/5/2024 11:16 PM	File folder	
Assignment 08.docx	6/10/2024 11:28 PM	Microsoft Word D...	18 KB
data_classes.py	6/10/2024 10:14 PM	Python File	3 KB
EmployeeRatings.json	6/11/2024 12:34 AM	JSON Source File	1 KB
main.py	6/10/2024 10:14 PM	Python File	2 KB
presentation_classes.py	6/10/2024 10:14 PM	Python File	4 KB
processing_classes.py	6/10/2024 10:14 PM	Python File	3 KB
test_data_classes.py	6/10/2024 10:14 PM	Python File	1 KB
test_presentation_classes.py	6/10/2024 10:14 PM	Python File	2 KB
test_processing_classes.py	6/10/2024 10:14 PM	Python File	2 KB

```
[{"first_name": "Adnan", "last_name": "Dawood", "review_date": "1970-01-29", "review_rating": 5}, {"first_name": "John", "last_name": "Moor", "review_date": "1980-06-24",
"review_rating": 4}, {"first_name": "Adam", "last_name": "Rick", "review_date": "1990-05-02", "review_rating": 3}]
```

4- Menu option 4 will exit the program as shown below:



```
Enter the employee's last name: Rick
Enter the review date (YYYY-MM-DD): 1990
Invalid date format. Please use YYYY-MM-DD.
Enter the review date (YYYY-MM-DD): 1990-05-02
Enter the review rating (1-5): 3
New employee data added.

---- Employee Ratings ------------------------------
  Select from the following menu:
    1. Show current employee rating data.
    2. Enter new employee rating data.
    3. Save data to a file.
    4. Exit the program.
--------------------------------------------------

Please select an option (1-4): 1
Name: Adnan Dawood, Review Date: 1970-01-29, Rating: 5
Name: John Moor, Review Date: 1980-06-24, Rating: 4
Name: Adam Rick, Review Date: 1990-05-02, Rating: 3

---- Employee Ratings ------------------------------
  Select from the following menu:
    1. Show current employee rating data.
    2. Enter new employee rating data.
    3. Save data to a file.
    4. Exit the program.
--------------------------------------------------

Please select an option (1-4): 3
Data successfully saved to file.

---- Employee Ratings ------------------------------
  Select from the following menu:
    1. Show current employee rating data.
    2. Enter new employee rating data.
    3. Save data to a file.
    4. Exit the program.
--------------------------------------------------

Please select an option (1-4): 4
Exiting program.

C:\Assignments\Assignment 08>
```

About the scrip and the associated modules:

1- Special attention was paid to the user input error handling. The script should be able to check whether there is a json file within its directory and create one if it does not exist. Furthermore, it will print “No employee data available.” If the json file is empty. Also, it will check if there are unnecessary spaces added during the input data, will check for date format if it is valid or not and will display the following message if the format is not correct “Invalid date format. Please use YYYY-MM-DD.” In addition to checking the review rating whether the input is within the expected range of 1 to 5 and will display the following message if the input number is out of range “Review rating must be between 1 and 5.”. etc.

Summary

The Python program consists of several modules:

data_classes.py: Defines two classes, Person and Employee, representing a basic person and an employee respectively. These classes encapsulate attributes such as first name, last name, review date, and review rating, with appropriate validation.

main.py: Contains the main program logic. It imports classes from other modules (IO from presentation_classes, FileProcessor from processing_classes, and Employee from data_classes). The main function allows users to interact with the program through a menu-driven interface, enabling them to view, add, and save employee data.

presentation_classes.py: Provides presentation-related functionality. It defines a class IO with methods for displaying menus, collecting user input, and showing employee data.

processing_classes.py: Contains classes for processing data to and from files. The FileProcessor class has methods for reading employee data from a JSON file and writing employee data to a JSON file.

test_data_classes.py, test_presentation_classes.py, and test_processing_classes.py: These modules contain unit tests for the classes defined in data_classes.py, presentation_classes.py, and processing_classes.py, respectively.

The program allows users to manage employee data, including viewing existing data, adding new employees, and saving data to a file. It incorporates error handling and validation throughout, ensuring data integrity and robustness. Additionally, unit tests are provided to verify the correctness of the program's functionality.