Assignment08_KhursheedKhan

Khursheed Khan
Dec 12 2024
Introduction to Python Programming

Enter GitHub →

# Assignment 08: Employ classes to create employee registry, the program we continued from last assignments, ensure .json read in from the dictionary format as list and then list is input as dictionary into .json with date/ rating format and test modules

## Introduction

This assignment demonstrates the implementation of conditional logic and looping constructs in Python. The program creates an employee registration system that allows users to input employee information, view current data, and save the information to a .json file. It showcases fundamental programming concepts including file handling, user input processing, and menu-driven program flow. It shows us how to write multiple entries to a file as well. This implementation follows the class-OOP structure.

Here is the markdown document generated via ChatGpt:

```
# main.py Documentation

## Purpose
This Python script demonstrates how to handle employee data using object-oriented programming (OOP) principles. It makes use of `Employee` and `Person` classes to manage employee information, including their first name, last name, review date, and review rating. The script also implements functionality for reading and writing data to JSON files, as well as handling errors and user input through a menu interface.

## Overview
- **Classes**: The script defines two main classes, `Person` and `Employee`, to represent individual employee data.
- **File Operations**: The script reads from and writes to a JSON file, handling exceptions appropriately.
- **User Interface**: The script provides a basic text interface for the user to interact with the program.

## Class Definitions

### Person Class
The `Person` class manages basic information about a person, such as their first and last names.

#### Attributes:
- `first_name` (str): The first name of the person.
- `last_name` (str): The last name of the person.
```

```
#### Methods:
- **`__init__(self, user_first_name: str, user_last_name: str)`**: Initializes the first and last names of the person.
- **`first_name`** (getter and setter): Ensures the first name is alphanumeric and capitalized.
- **`last_name`** (getter and setter): Ensures the last name is alphanumeric and capitalized.
- **`__str__(self)`**: Returns a string representation of the person's name in the format "First Name, Last Name".

### Employee Class
The `Employee` class inherits from `Person` and adds additional attributes related to the employee's review information.

#### Attributes:
- `review_date` (date): The date of the employee's review.
- `review_rating` (int): The rating given to the employee (between 1 and 5).

#### Methods:
- **`__init__(self, employee_first_name: str, employee_last_name: str, employee_review_date: date | str, employee_review_rating: int)`**: Initializes employee data, including the first name, last name, review date, and review rating.
- **`review_date`** (getter and setter): Ensures that the review date is a valid date or a string in the ISO 8601 format.
- **`review_rating`** (getter and setter): Ensures that the review rating is an integer between 1 and 5.
- **`__str__(self)`**: Returns a string representation of the employee, including their name, review date, and review rating.

## Functions

### FileProcessor Class
The `FileProcessor` class provides static methods to read and write employee data to JSON files.

#### Methods:
- **`read_employee_data_from_file(file_name: str, employee_data: list)`**: Reads employee data from a JSON file and converts it into a list of `Employee` objects.
  - If the file doesn't exist, it creates a new one.
- **`write_employee_data_to_file(file_name: str, employee_data: list)`**: Writes employee data to a JSON file, formatting it as a list of dictionaries.

### IO Class
The `IO` class handles user input and output, allowing for interaction through the terminal.

#### Methods:
- **`output_error_messages(message: str, error: Exception)`**: Displays custom error messages along with the technical error details.
- **`output_menu(menu: str)`**: Displays a formatted menu to the user.
- **`input_menu_choice()`**: Prompts the user to select a menu option.
- **`input_employee_data(employee_data: list)`**: Collects employee information from the user and adds it to the `employee_data` list.
- **`output_employee_data(employee_data: list)`**: Displays the list of employees with their review date and rating.

## Example Usage

### Running the Script:
```

Assignment08_KhursheedKhan

```
1. The user is presented with a menu where they can select various options such as:
  - Add a new employee.
  - View existing employee data.
  - Exit the program.

2. The user can input employee data, including the first name, last name, review date, and review rating.

3. The employee data is saved to a JSON file and can be reloaded when the program restarts.

### Sample Input:
```

Topic

Classes!

Code

Output from Pycharm

Output from console

Enrollments.csv

Summary

A beautiful program using/ employing essential class-structure!

Assignment08_KhursheedKhan

**ChatGPT to write the document with my code as input, this is a markdown doc**

```
# User Documentation for Employee Management System

## Table of Contents
1. [Introduction](#introduction)
2. [Classes and Their Purpose](#classes-and-their-purpose)
   - [Person Class](#person-class)
   - [Employee Class](#employee-class)
   - [FileProcessor Class](#fileprocessor-class)
   - [IO Class](#io-class)
3. [Usage Instructions](#usage-instructions)
   - [Running the Script](#running-the-script)
   - [Menu Options](#menu-options)
4. [Error Handling](#error-handling)
5. [Example](#example)
6. [Conclusion](#conclusion)

## Introduction
The Employee Management System allows users to manage employee data through a simple interface. The script is built using Python and demonstrates how to use classes and objects to store and manage employee information, such as name, review date, and review rating. Additionally, it offers functionality to save and load employee data from a JSON file and interact with the user through a terminal-based menu.

The program is divided into four main components:
1. **Person Class**: Handles basic personal information.
2. **Employee Class**: Inherits from `Person` and adds additional employee-related information.
3. **FileProcessor Class**: Manages reading and writing employee data to a JSON file.
4. **IO Class**: Provides functions for user input and output, including menu management.

## Classes and Their Purpose

### Person Class
The `Person` class is designed to store basic personal information about a person, specifically their first and last names.

#### Attributes:
- `first_name` (str): The first name of the person.
- `last_name` (str): The last name of the person.

#### Methods:
- **`__init__(self, user_first_name: str, user_last_name: str)`**: Initializes a new `Person` instance with a first name and last name. The names are capitalized to maintain consistency.
- **`first_name`** (getter and setter): Ensures that the first name is alphabetic and capitalized.
- **`last_name`** (getter and setter): Ensures that the last name is alphabetic and capitalized.
- **`__str__(self)`**: Returns a string representation of the person's name in the format "First Name, Last Name".

### Employee Class
The `Employee` class inherits from `Person` and adds employee-specific information, such as the review date and review rating.

#### Attributes:
- `review_date` (date): The date of the employee's review.
```

```
- `review_rating` (int): The rating given to the employee (between 1 and 5).

#### Methods:
- **`__init__(self, employee_first_name: str, employee_last_name: str, employee_review_date: date | str, employee_review_rating: int)`**: Initializes an employee's details, including their first name, last name, review date, and review rating.
- **`review_date`** (getter and setter): Ensures that the review date is in a valid format (either a `date` object or an ISO 8601 string).
- **`review_rating`** (getter and setter): Ensures that the review rating is an integer between 1 and 5.
- **`__str__(self)`**: Returns a string representation of the employee, including their name, review date, and review rating.

### FileProcessor Class
The `FileProcessor` class contains static methods for reading from and writing to JSON files. It allows for the storage and retrieval of employee data.

#### Methods:
- **`read_employee_data_from_file(file_name: str, employee_data: list)`**: Reads employee data from a specified JSON file and converts it into `Employee` objects.
  - If the file doesn't exist, it creates a new file.
- **`write_employee_data_to_file(file_name: str, employee_data: list)`**: Writes a list of `Employee` objects to a JSON file, storing them as dictionaries.

### IO Class
The `IO` class provides user interface functions for handling input and output through the terminal.

#### Methods:
- **`output_error_messages(message: str, error: Exception)`**: Displays custom error messages along with technical error details, if any.
- **`output_menu(menu: str)`**: Displays a formatted menu to the user, presenting available options.
- **`input_menu_choice()`**: Prompts the user to select a menu option. It validates the choice to ensure it is a valid option.
- **`input_employee_data(employee_data: list)`**: Collects employee data (first name, last name, review date, and review rating) from the user and adds it to the employee data list.
- **`output_employee_data(employee_data: list)`**: Displays the list of employees, showing their first name, last name, review date, and review rating.

## Usage Instructions

### Running the Script
1. Upon running the script, the user is presented with a text-based menu interface that allows them to choose from various options (e.g., add an employee, view employee data, or exit).
2. The script will guide the user to input employee details such as:
   - First Name
   - Last Name
   - Review Date (in the format YYYY-MM-DD)
   - Review Rating (an integer between 1 and 5)

3. The data is stored in memory and can be saved to a JSON file. The script will handle reading from and writing to the file as necessary.

### Menu Options
- **Option 1**: Add a new employee
```

```
  - Prompts the user to enter the employee's first name, last name, review date, and review rating.
- **Option 2**: View existing employee data
  - Displays a list of all employees along with their review dates and ratings.
- **Option 3**: Exit the program
  - Exits the program and saves the data to the JSON file if applicable.

## Error Handling
The script is designed to handle various errors:
- **Invalid Names**: If the user enters a name with non-alphabetic characters, a `ValueError` will be raised.
- **Invalid Review Dates**: If the review date is not in the correct format (e.g., `YYYY-MM-DD`), a `ValueError` will be raised.
- **Invalid Review Ratings**: If the review rating is outside the range of 1 to 5, a `ValueError` will be raised.
- **File Errors**: If the JSON file does not exist or is corrupted, the script will display an error message and create a new file.

## Example

### Running the Script:
Upon running the script, the user is presented with the following menu:
```