Module 8.2 JSON Practice

SPRING 2025 CSD325 ADVANCED PYTHON

Author: Brittaney Perry-Morgan
Date: Sunday, June 29th, 2025

Module 8.2 JSON Practice

main.py

```python
"""
Name: Brittaney Perry-Morgan
Date: Sunday, June 29th, 2025
Assignment: Module 8.2 JSON Practice

Purpose: Implementation of a student management system.

Imports:
    - json: Used to interact with the JSON file.
    - sys: Used to add the project root to the Python path.
    - Path: Used to work with file paths.
"""

import json
import os
import sys
import tkinter as tk
from pathlib import Path
from tkinter import messagebox

# Add project root to the Python path to
project_root = Path(__file__).resolve().parents[1]
sys.path.append(str(project_root))

# flake8: noqa: E402

from module_8.student import (  # pylint: disable=wrong-import-position
    Student,
    StudentList,
)

JSON_FILE_PATH = Path(__file__).parent / "data" / "student.json"


def load_students(file_path: Path) -> StudentList:
    """
    Loads students from JSON (PascalCase keys). Removes existing duplicates.

    Parameters:
        - file_path: The path to the JSON file.
        :type file_path: Path

    Returns:
        - A StudentList containing the loaded students.
        :rtype: StudentList
    """
    students_data = []
    if file_path.exists() and file_path.stat().st_size > 0:
        with open(file_path, encoding="utf-8", mode="r") as file:
            try:
                students_data = json.load(file)
            except json.JSONDecodeError:
                print(
                    f"Warning: JSON file at {file_path} is empty or malformed. \n"
                    f"Starting with an empty list."
                )
                students_data = []
    students = [
        Student(
            f_name=item["F_Name"],
            l_name=item["L_Name"],
            student_id=item["Student_ID"],
            email=item["Email"],
        )
        for item in students_data
        if all(k in item for k in ("F_Name", "L_Name", "Student_ID", "Email"))
    ]
    student_list = StudentList(students)  # pylint: disable=redefined-outer-name
    student_list.remove_duplicates()  # Remove any existing duplicates in the file!
    return student_list


# pylint: disable=redefined-outer-name
def save_students(file_path: Path, student_list: StudentList) -> None:
    """
    Save students to JSON with PascalCase keys.

    Parameters:
        - file_path: The path to the JSON file.
        :type file_path: Path

        - student_list: The StudentList to save.
        :type student_list: StudentList
    """
    file_path.parent.mkdir(parents=True, exist_ok=True)
    data_to_save = [
        {
            "F_Name": s.f_name,
            "L_Name": s.l_name,
            "Student_ID": s.student_id,
            "Email": s.email,
        }
        for s in student_list.students
    ]
    try:
        with open(file_path, encoding="utf-8", mode="w") as file:
            json.dump(data_to_save, file, indent=4)
    except (IOError, OSError) as e:
        print(f"Error saving students to {file_path}: {e}")
        raise


def student_notification(user_msg: str) -> str:
    """
    Returns the user message if it's not empty, otherwise returns a default message.

    Parameters:
        - user_msg: The user message to return.
        :type user_msg: str

    Returns:
        - The user message if it's not empty, otherwise returns a default message.
        :rtype: str
    """
    return user_msg or "Invalid input. Please try again..."


def get_relative_path(path: Path) -> str:
    """
    Returns a relative path string from the current working directory to the given path.

    Parameters:
        - path: The path to get the relative path for.
        :type path: Path

    Returns:
        - The relative path string.
        :rtype: str
    """
    try:
        return os.path.relpath(str(path), start=os.getcwd())
    except ValueError as e:
        print(f"Error getting relative path: {e}")
        return str(path)


def show_save_dialog(file_path: Path) -> bool:
    """
    Shows a dialog asking the user if they want to save the changes.

    Parameters:
        - file_path: The path to the file being modified.
        :type file_path: Path

    Returns:
        - True if the user clicks 'Yes', False otherwise.
        :rtype: bool
    """
    root = tk.Tk()
    root.withdraw()  # Hide the main window

    relative_path = get_relative_path(file_path)
    message = f"This file has been modified outside. Do you want to reload it?\n\n{relative_path}"
    return messagebox.askyesno("File Modified", message)


if __name__ == "__main__":
    JSON_FILE_PATH.parent.mkdir(parents=True, exist_ok=True)

    student_list = load_students(JSON_FILE_PATH)
    print(student_notification(f"\n{'=' * 50}\nOriginal Student List:\n{'=' * 50}\n"))
    student_list.print_students()
    print("\n")

    new_student = Student(
        f_name="Brittaney",
        l_name="Perry-Morgan",
        student_id=12345,
        email="bperrymorgan@me.com",
    )

    if student_list.contains_student(new_student):
        print(
            student_notification(
                f"\n***** DUPLICATE STUDENT DETECTED: ({new_student}) will not be added.*****\n\n"
            )
        )
    else:
        student_list.add_student(new_student)
        print(
            student_notification(
                f"\n***** STUDENT ADDED: ({new_student}) has been added.*****\n\n"
            )
        )
        if show_save_dialog(JSON_FILE_PATH):
            save_students(JSON_FILE_PATH, student_list)
            print(student_notification("Changes saved."))
        else:
            print(student_notification("Changes not saved."))

    print(student_notification(f"\n{'=' * 50}\nUpdated Student List:\n{'=' * 50}\n"))
    student_list.print_students()
    print("\n")

    student_list = load_students(JSON_FILE_PATH)
    print(
        student_notification(
            f"\n{'=' * 50}\nUpdated Student List from JSON:\n{'=' * 50}\n"
        )
    )
    student_list.print_students()
    print("\n")
```

data/student.json (original)

```
[
    {
        "F_Name": "Ellen",
        "L_Name": "Ripley",
        "Student_ID": 45604,
        "Email": "eripley@gmail.com"
    },
    {
        "F_Name": "Arthur",
        "L_Name": "Dallas",
        "Student_ID": 45605,
        "Email": "adallas@gmail.com"
    },
    {
        "F_Name": "Joan",
        "L_Name": "Lambert",
        "Student_ID": 45714,
        "Email": "jlambert@gmail.com"
    },
    {
        "F_Name": "Thomas",
        "L_Name": "Kane",
        "Student_ID": 68554,
        "Email": "tkane@gmail.com"
    }
]
```

data/student.json (updated)

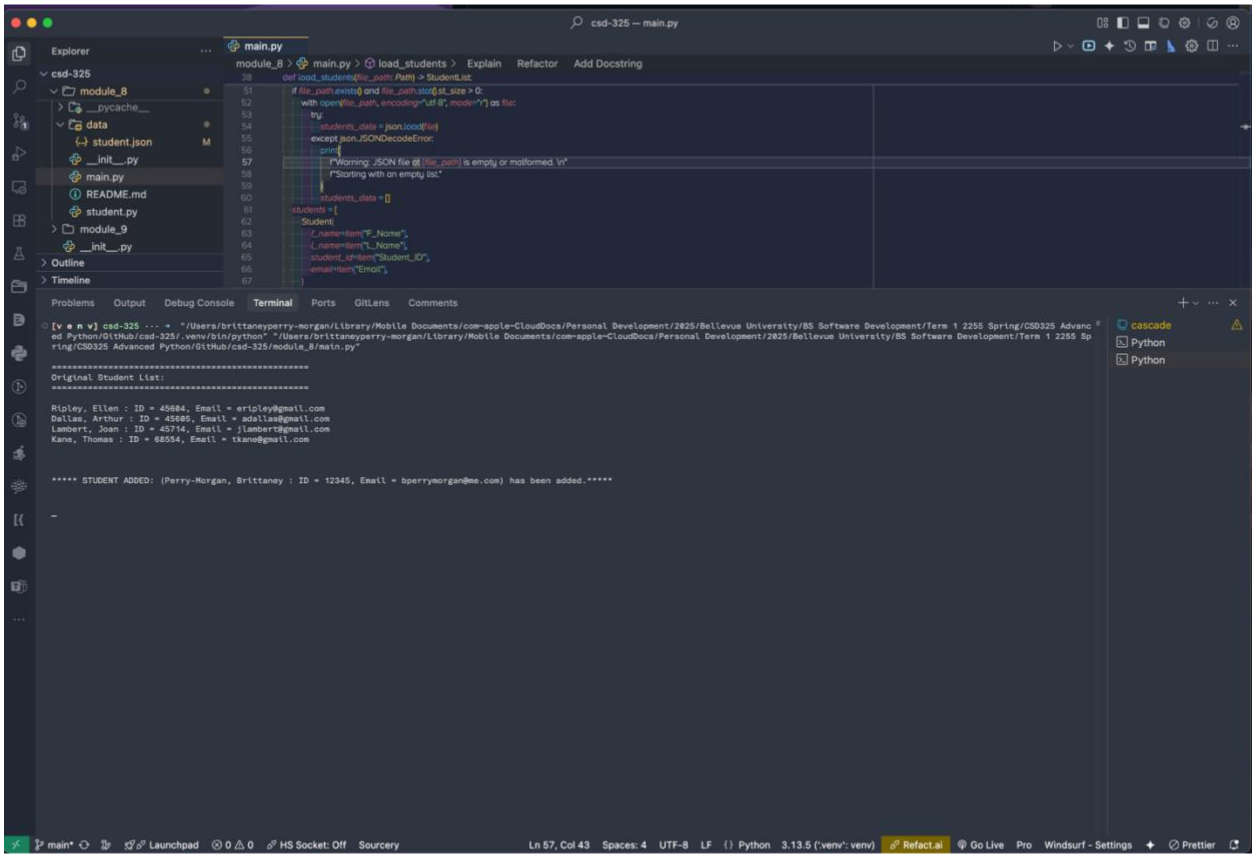
```
[
    {
        "F_Name": "Ellen",
        "L_Name": "Ripley",
        "Student_ID": 45604,
        "Email": "eripley@gmail.com"
    },
    {
        "F_Name": "Arthur",
        "L_Name": "Dallas",
        "Student_ID": 45605,
        "Email": "adallas@gmail.com"
    },
    {
        "F_Name": "Joan",
        "L_Name": "Lambert",
        "Student_ID": 45714,
        "Email": "jlambert@gmail.com"
    },
    {
        "F_Name": "Thomas",
        "L_Name": "Kane",
        "Student_ID": 68554,
        "Email": "tkane@gmail.com"
    },
    {
        "F_Name": "Brittaney",
        "L_Name": "Perry-Morgan",
        "Student_ID": 12345,
        "Email": "bperrymorgan@me.com"
    }
]
```

student.py

```
"""
Name: Brittaney Perry-Morgan
Date: Sunday, June 1st, 2025
Assignment: Module 8.2 JSON

Purpose: Holds the Student and StudentList dataclasses.
```

```python
Imports:
    - dataclass: Used to create dataclasses.
    - field: Used to create default factory for the students list.
    - List: Used to type hint the students list.
"""

from dataclasses import dataclass, field
from typing import List


@dataclass
class Student:
    """
    Representation of a Student.

    Fields:
        - f_name: The first name of the student.
        :type f_name: str

        - l_name: The last name of the student.
        :type l_name: str

        - student_id: The student's unique ID.
        :type student_id: int

        - email: The student's email address.
        :type email: str
    """

    f_name: str
    l_name: str
    student_id: int
    email: str

    def __str__(self) -> str:
        """String representation of a student."""
        return f"{self.l_name}, {self.f_name} : ID = {self.student_id}, Email = {self.email}"


@dataclass
class StudentList:
    """
    Representation of a list of students.

    Fields:
        - students: The list of students.
        :type students: List[Student]
    """

    students: List[Student] = field(default_factory=list)

    def __iter__(self):
        """Iterator for the StudentList."""
        return iter(self.students)

    def print_students(self) -> None:
        """Print all students in the list."""
        for student in self.students:
            print(student)

    def add_student(self, student: Student) -> None:
        """
        Add a student to the list.

        Parameters:
            - student: The Student to add to the list.
            :type student: Student
        """
        self.students.append(student)

    def contains_student(self, student: "Student") -> bool:
        """
        Check for duplicate by student_id OR email.

        Parameters:
            - student: The student to check for duplicates.
            :type student: Student

        Returns:
            - True if the student is a duplicate, False otherwise.
            :rtype: bool
        """
        return any(
            s.student_id == student.student_id or s.email == student.email
            for s in self.students
        )

    def remove_duplicates(self) -> None:
        """
        Removes duplicate students by student_id or email, keeping the first occurrence.
        """
        seen_ids = set()
        seen_emails = set()
        unique_students = []
        for s in self.students:
            if s.student_id not in seen_ids and s.email not in seen_emails:
                unique_students.append(s)
                seen_ids.add(s.student_id)
                seen_emails.add(s.email)
        self.students = unique_students
```

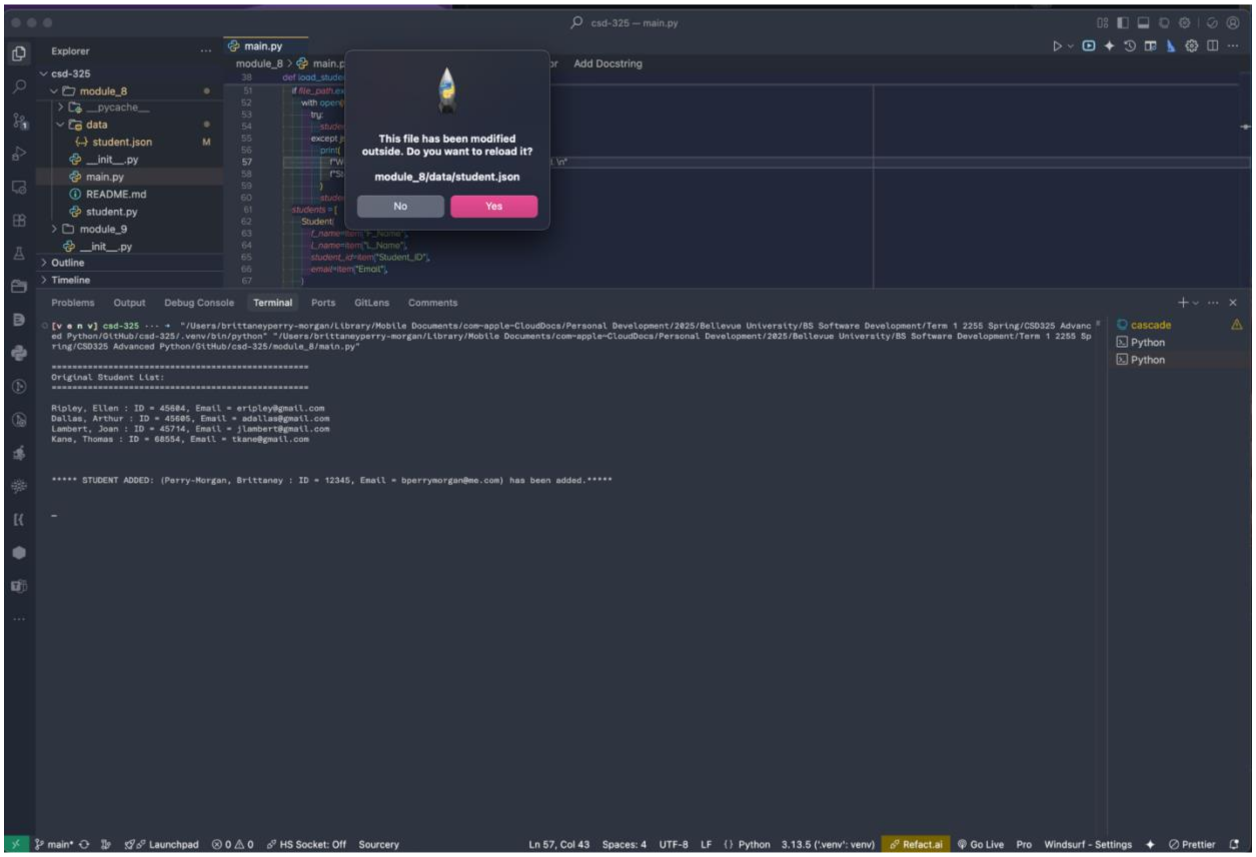
deliverables



```python
def load_students(file_path: Path) -> StudentList:
    if file_path.exists() and file_path.stat().st_size > 0:
        with open(file_path, encoding="utf-8", mode="r") as file:
            try:
                students_data = json.load(file)
            except json.JSONDecodeError:
                print(
                    f"Warning: JSON file at {file_path} is empty or malformed. \n"
                    f"Starting with an empty list."
                )
                students_data = []
    students = [
        Student(
            f_name=item["F_Name"],
            l_name=item["L_Name"],
            student_id=item["Student_ID"],
            email=item["Email"],
        )
```

```
[v e n v] csd-325 ··· →  "/Users/brittaneyperry-morgan/Library/Mobile Documents/com-apple-CloudDocs/Personal Development/2025/Bellevue University/BS Software Development/Term 1 2255 Spring/CSD325 Advanced Python/GitHub/csd-325/.venv/bin/python" "/Users/brittaneyperry-morgan/Library/Mobile Documents/com-apple-CloudDocs/Personal Development/2025/Bellevue University/BS Software Development/Term 1 2255 Spring/CSD325 Advanced Python/GitHub/csd-325/module_8/main.py"

==================================================
Original Student List:
==================================================

Ripley, Ellen : ID = 45604, Email = eripley@gmail.com
Dallas, Arthur : ID = 45605, Email = adallas@gmail.com
Lambert, Joan : ID = 45714, Email = jlambert@gmail.com
Kane, Thomas : ID = 68554, Email = tkane@gmail.com


***** STUDENT ADDED: (Perry-Morgan, Brittaney : ID = 12345, Email = bperrymorgan@me.com) has been added.*****
```

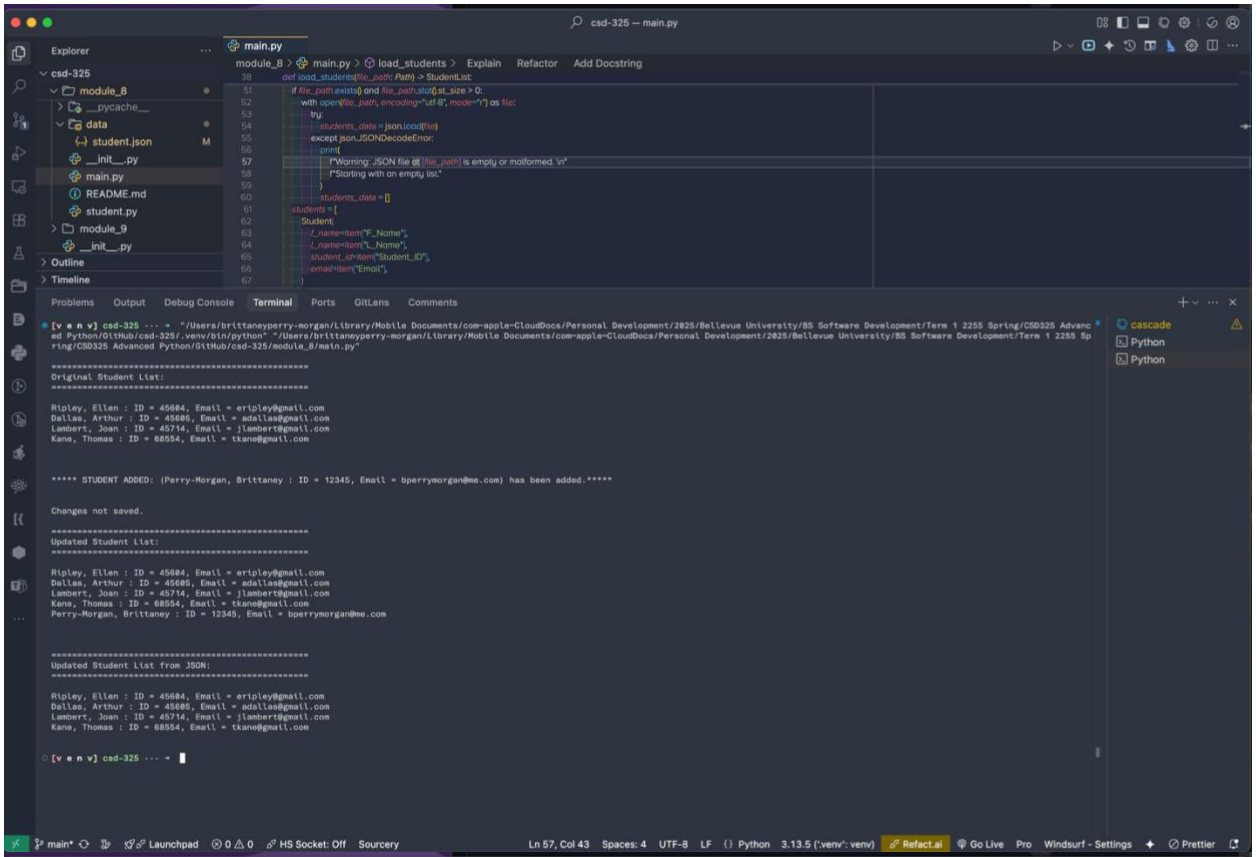
csd-325 — main.py
Explorer
csd-325
module_8
__pycache__
data
student.json
__init__.py
main.py
README.md
student.py
module_9
__init__.py
Outline
Timeline
module_8 > main.py > load_students > Explain Refactor Add Docstring
Problems Output Debug Console Terminal Ports GitLens Comments
cascade
Python
Python
main* Launchpad HS Socket: Off Sourcery Ln 57, Col 43 Spaces: 4 UTF-8 LF Python 3.13.5 ('.venv': venv) Refact.ai Go Live Pro Windsurf - Settings Prettier



This file has been modified outside. Do you want to reload it?

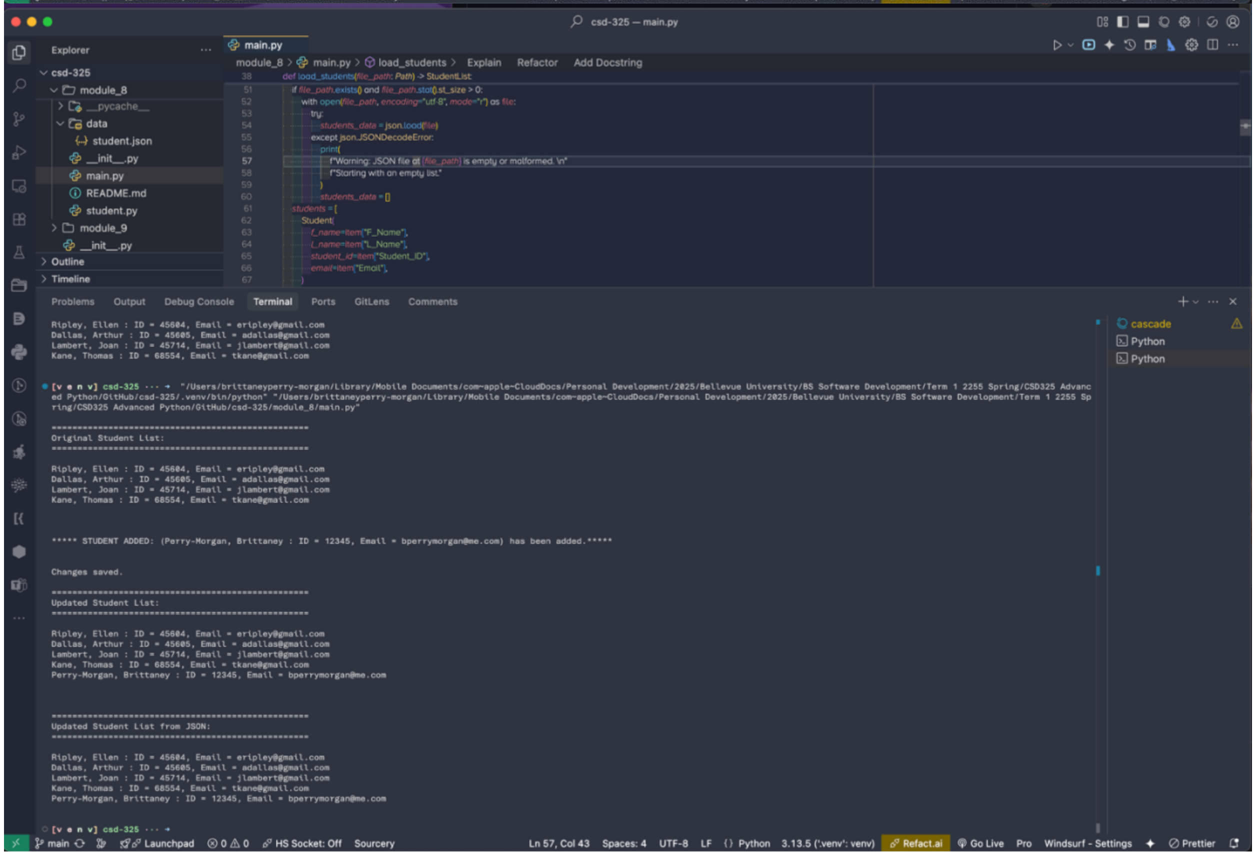
module_8/data/student.json

```
[v e n v] csd-325 ··· → "/Users/brittaneyperry-morgan/Library/Mobile Documents/com~apple~CloudDocs/Personal Development/2025/Bellevue University/BS Software Development/Term 1 2255 Spring/CSD325 Advanc
ed Python/GitHub/csd-325/.venv/bin/python" "/Users/brittaneyperry-morgan/Library/Mobile Documents/com~apple~CloudDocs/Personal Development/2025/Bellevue University/BS Software Development/Term 1 2255 Sp
ring/CSD325 Advanced Python/GitHub/csd-325/module_8/main.py"

==================================================
Original Student List:
==================================================

Ripley, Ellen : ID = 45604, Email = eripley@gmail.com
Dallas, Arthur : ID = 45605, Email = adallas@gmail.com
Lambert, Joan : ID = 45714, Email = jlambert@gmail.com
Kane, Thomas : ID = 68554, Email = tkane@gmail.com


***** STUDENT ADDED: (Perry-Morgan, Brittaney : ID = 12345, Email = bperrymorgan@me.com) has been added.*****
```



```python
def load_students(file_path: Path) -> StudentList:
    if file_path.exists() and file_path.stat().st_size > 0:
        with open(file_path, encoding="utf-8", mode="r") as file:
            try:
                students_data = json.load(file)
            except json.JSONDecodeError:
                print(
                    f"Warning: JSON file at {file_path} is empty or malformed. \n"
                    f"Starting with an empty list."
                )
                students_data = []
    students = [
        Student(
            f_name=item["F_Name"],
            l_name=item["L_Name"],
            student_id=item["Student_ID"],
            email=item["Email"],
        )
```

```
[v e n v] csd-325 ··· → "/Users/brittaneyperry-morgan/Library/Mobile Documents/com-apple-CloudDocs/Personal Development/2025/Bellevue University/BS Software Development/Term 1 2255 Spring/CSD325 Advanced Python/GitHub/csd-325/.venv/bin/python" "/Users/brittaneyperry-morgan/Library/Mobile Documents/com-apple-CloudDocs/Personal Development/2025/Bellevue University/BS Software Development/Term 1 2255 Spring/CSD325 Advanced Python/GitHub/csd-325/module_8/main.py"

==================================================
Original Student List:
==================================================

Ripley, Ellen : ID = 45604, Email = eripley@gmail.com
Dallas, Arthur : ID = 45605, Email = adallas@gmail.com
Lambert, Joan : ID = 45714, Email = jlambert@gmail.com
Kane, Thomas : ID = 68554, Email = tkane@gmail.com


***** STUDENT ADDED: (Perry-Morgan, Brittaney : ID = 12345, Email = bperrymorgan@me.com) has been added.*****


Changes not saved.

==================================================
Updated Student List:
==================================================

Ripley, Ellen : ID = 45604, Email = eripley@gmail.com
Dallas, Arthur : ID = 45605, Email = adallas@gmail.com
Lambert, Joan : ID = 45714, Email = jlambert@gmail.com
Kane, Thomas : ID = 68554, Email = tkane@gmail.com
Perry-Morgan, Brittaney : ID = 12345, Email = bperrymorgan@me.com


==================================================
Updated Student List from JSON:
==================================================

Ripley, Ellen : ID = 45604, Email = eripley@gmail.com
Dallas, Arthur : ID = 45605, Email = adallas@gmail.com
Lambert, Joan : ID = 45714, Email = jlambert@gmail.com
Kane, Thomas : ID = 68554, Email = tkane@gmail.com

[v e n v] csd-325 ··· →
```



```python
def load_students(file_path: Path) -> StudentList:
    if file_path.exists() and file_path.stat().st_size > 0:
        with open(file_path, encoding="utf-8", mode="r") as file:
            try:
                students_data = json.load(file)
            except json.JSONDecodeError:
                print(
                    f"Warning: JSON file at {file_path} is empty or malformed. \n"
                    f"Starting with an empty list."
                )
                students_data = []
    students = [
        Student(
            f_name=item["F_Name"],
            l_name=item["L_Name"],
            student_id=item["Student_ID"],
            email=item["Email"],
        )
```

```
Ripley, Ellen : ID = 45604, Email = eripley@gmail.com
Dallas, Arthur : ID = 45605, Email = adallas@gmail.com
Lambert, Joan : ID = 45714, Email = jlambert@gmail.com
Kane, Thomas : ID = 68554, Email = tkane@gmail.com

[v e n v] csd-325 ··· → "/Users/brittaneyperry-morgan/Library/Mobile Documents/com-apple-CloudDocs/Personal Development/2025/Bellevue University/BS Software Development/Term 1 2255 Spring/CSD325 Advanced Python/GitHub/csd-325/.venv/bin/python" "/Users/brittaneyperry-morgan/Library/Mobile Documents/com-apple-CloudDocs/Personal Development/2025/Bellevue University/BS Software Development/Term 1 2255 Spring/CSD325 Advanced Python/GitHub/csd-325/module_8/main.py"

==================================================
Original Student List:
==================================================

Ripley, Ellen : ID = 45604, Email = eripley@gmail.com
Dallas, Arthur : ID = 45605, Email = adallas@gmail.com
Lambert, Joan : ID = 45714, Email = jlambert@gmail.com
Kane, Thomas : ID = 68554, Email = tkane@gmail.com


***** STUDENT ADDED: (Perry-Morgan, Brittaney : ID = 12345, Email = bperrymorgan@me.com) has been added.*****


Changes saved.

==================================================
Updated Student List:
==================================================

Ripley, Ellen : ID = 45604, Email = eripley@gmail.com
Dallas, Arthur : ID = 45605, Email = adallas@gmail.com
Lambert, Joan : ID = 45714, Email = jlambert@gmail.com
Kane, Thomas : ID = 68554, Email = tkane@gmail.com
Perry-Morgan, Brittaney : ID = 12345, Email = bperrymorgan@me.com


==================================================
Updated Student List from JSON:
==================================================

Ripley, Ellen : ID = 45604, Email = eripley@gmail.com
Dallas, Arthur : ID = 45605, Email = adallas@gmail.com
Lambert, Joan : ID = 45714, Email = jlambert@gmail.com
Kane, Thomas : ID = 68554, Email = tkane@gmail.com
Perry-Morgan, Brittaney : ID = 12345, Email = bperrymorgan@me.com

[v e n v] csd-325 ··· →
```