**INTERNSHIP REPORT**

**ON**

**PYTHON COMPITATIVE CODEING**

**internship Report is submitted**

**In accordance with requirement of degree of**

**BACHELOR OF TECHNOLOGY**

**IN**

**ELECTRICAL AND ELECTRONICS ENGINEERING**

Submitted by

A.GANGADHAR BABU

21KQ1A0231

Under the Mentorship of

**SRAVAN MANTRI**

**DEPARTMENT OF ELECTRICAL AND EL ECTRONICS ENGINEERING**

PACE INSTITUTE OF TECNOLOGY AND SCIENCES (AUTONOMOUS)

(Affiliated to Jawaharlal Nehru Technological University Kakinada, Kakinada &

Accredited by NAAC ‘A’ GRADE,An ISO 9001-2015 Certified Institution)

NH-16, Valluru Post , Prakasam District, A.P-523272.

(2024-25)

# 

# PROJECT TITLE

COURSE NAVIGATOR PRO

**Abstract:**

Course Navigator Pro is a comprehensive e-learning platform designed to streamline the creation, management, and delivery of online courses. It offers robust features for course development, including modules, lessons, quizzes, assignments, and interactive discussion boards. The platform supports diverse learning needs and provides tools for tracking progress and enhancing student engagement.

**Description:**

- Navigator Pro aims to facilitate educators and organizations in developing and delivering high video, audio, text, and interactive elements, making it suitable for various educational contexts. quality online courses. The platform supports a wide range of content types, including Course It features a user-friendly interface that allows instructors to design courses with ease and enables students to navigate through the content seamlessly. Additionally, the platform offers tools for progress tracking, assessment, and communication to enhance the learning experiement.

**Requirment:**

Functional Requirements

1. \*Course Development:\*

- Tools to create and organize modules and lessons.

- Support for multimedia content (videos, audios, slides, documents).

2. \*Assessment and Feedback:\*

- Features to design quizzes and assignments.

- Automated grading and feedback mechanisms.

3. \*Student Interaction:\*

- Discussion boards for student engagement.

- Real-time chat and messaging options.

4. \*Progress Tracking:\*

- Dashboard for tracking student progress and performance.

- Reporting tools for detailed analytics.

5. \*User Management:\*

- Role-based access control (e.g., administrators, instructors, students).

- Secure authentication and user profile

## 

**2.-non-Functional Requirements:**

Non-functional requirements specify the quality attributes, constraints, and performance criteria that a software system must meet. Here are some non-functional requirements for the provided course management program:

1. **Performance**:
   * The system should handle input and display results within 1 second for each operation.
   * The program should support a maximum of 1000 courses without significant degradation in performance.
2. **Usability**:
   * The system should provide clear and concise prompts for user input.
   * Error messages should be informative and guide the user to correct their input.
   * The menu options should be intuitive and easy to navigate.
3. **Reliability**:
   * The program should handle invalid input gracefully without crashing.
   * It should provide accurate and consistent results for all operations.
4. **Maintainability**:
   * The code should be modular, with each function performing a single, well-defined task.
   * The code should be well-commented to facilitate understanding and maintenance by other developers.
5. **Scalability**:
   * The design should allow for easy addition of new features or modifications to existing features without significant restructuring of the code.

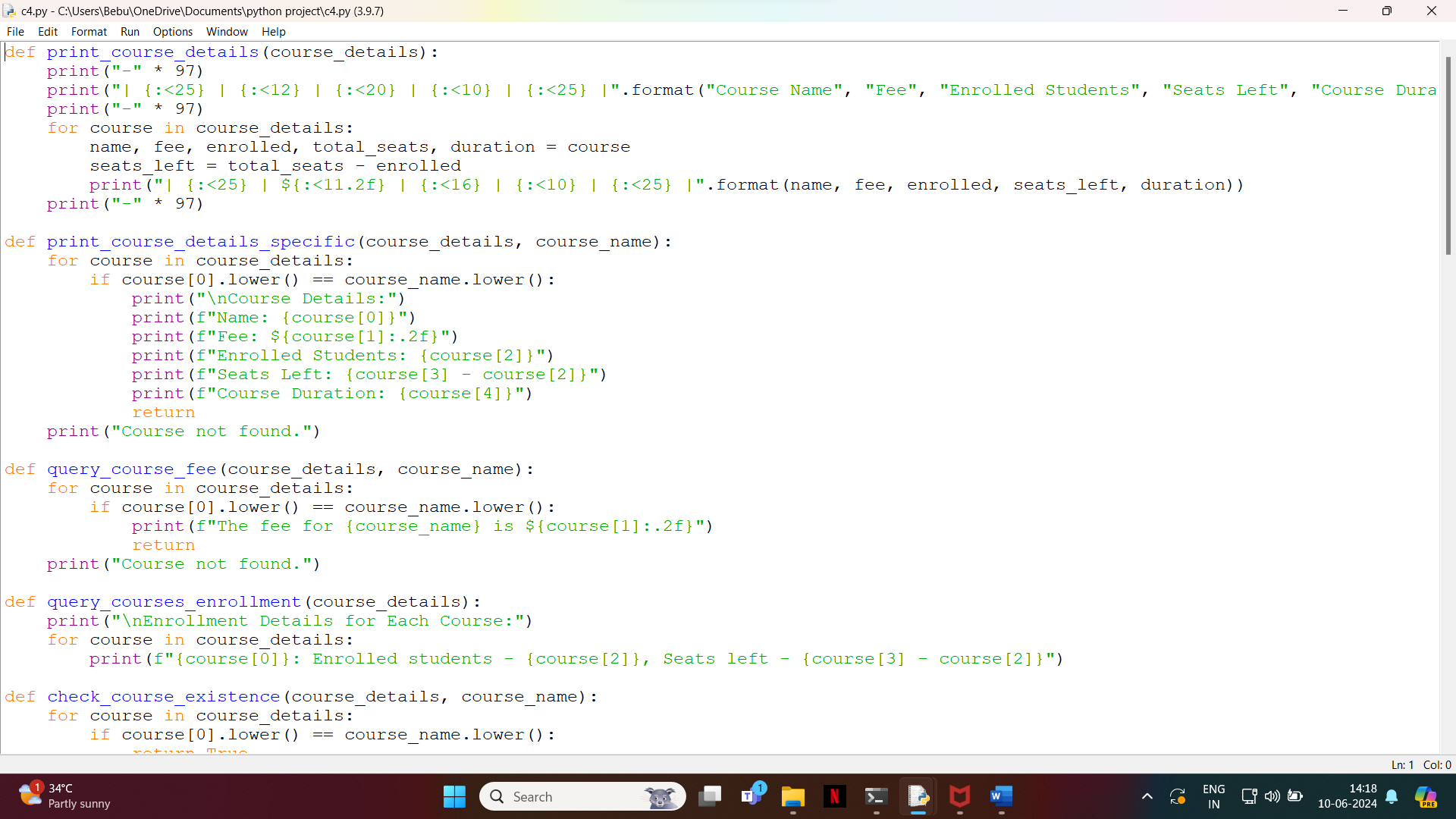
**Approach:**

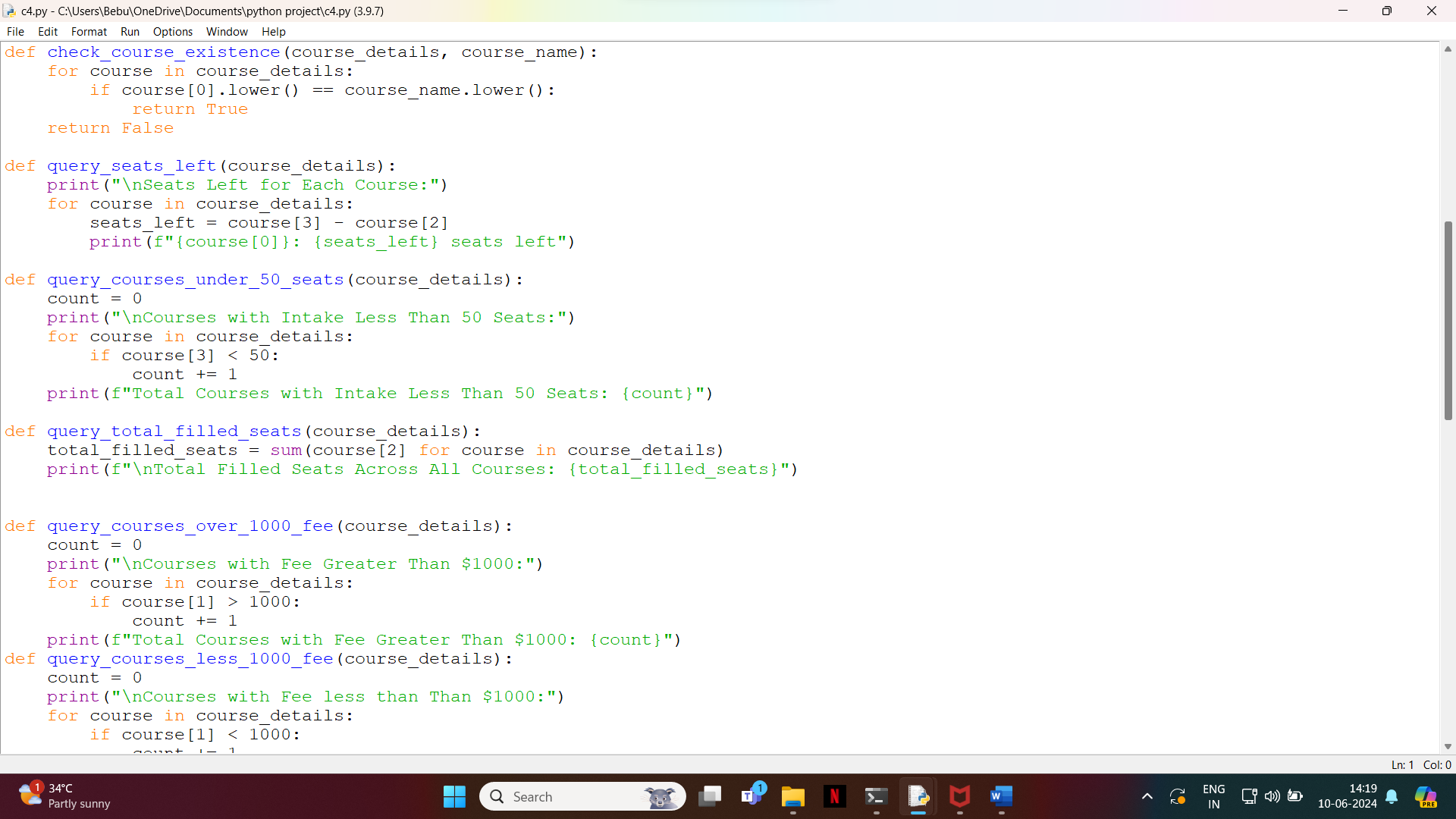
To improve the code and make it more efficient, maintainable, and user-friendly, follow these steps:

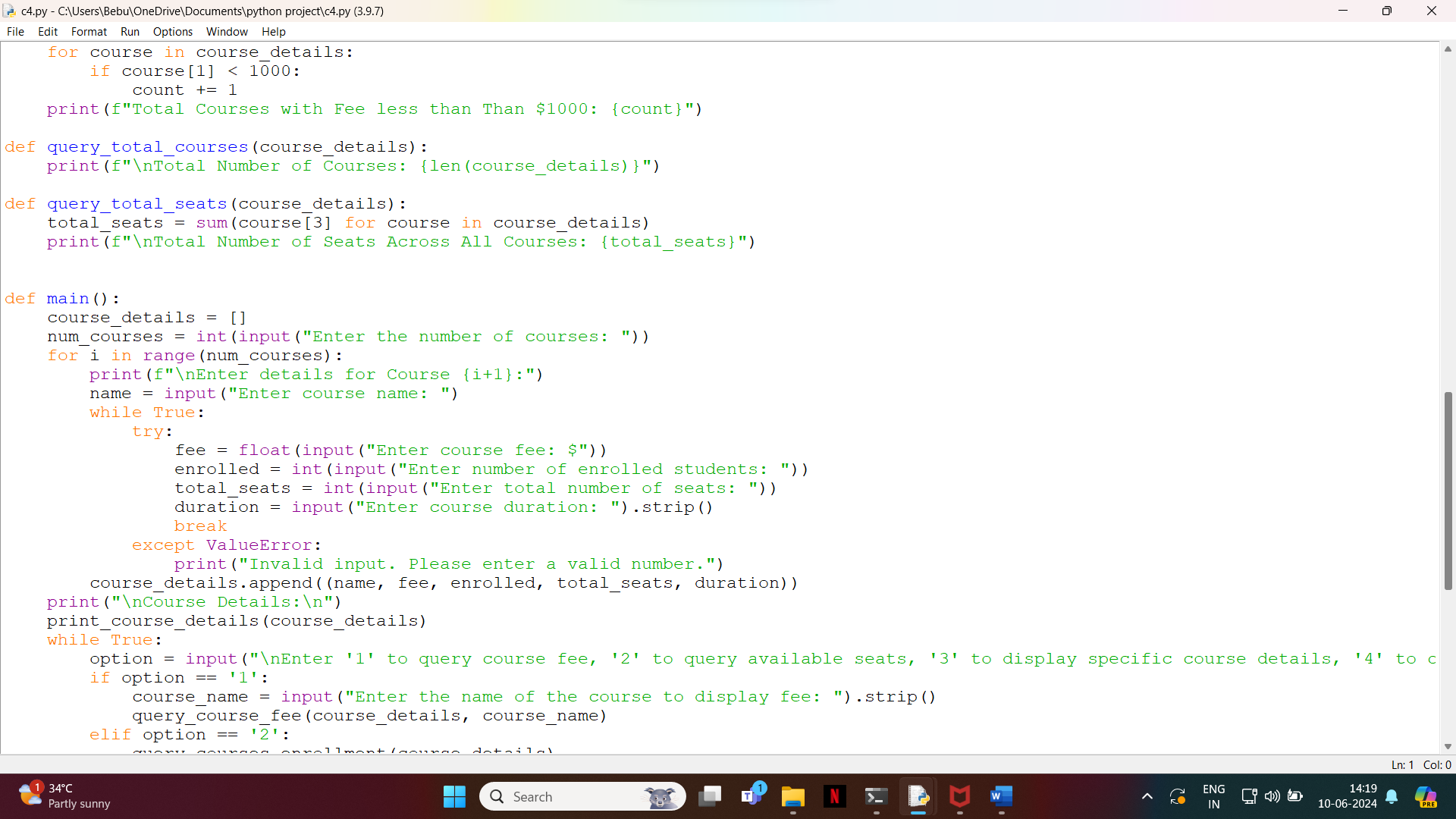
### Code Refactoring and Improvements

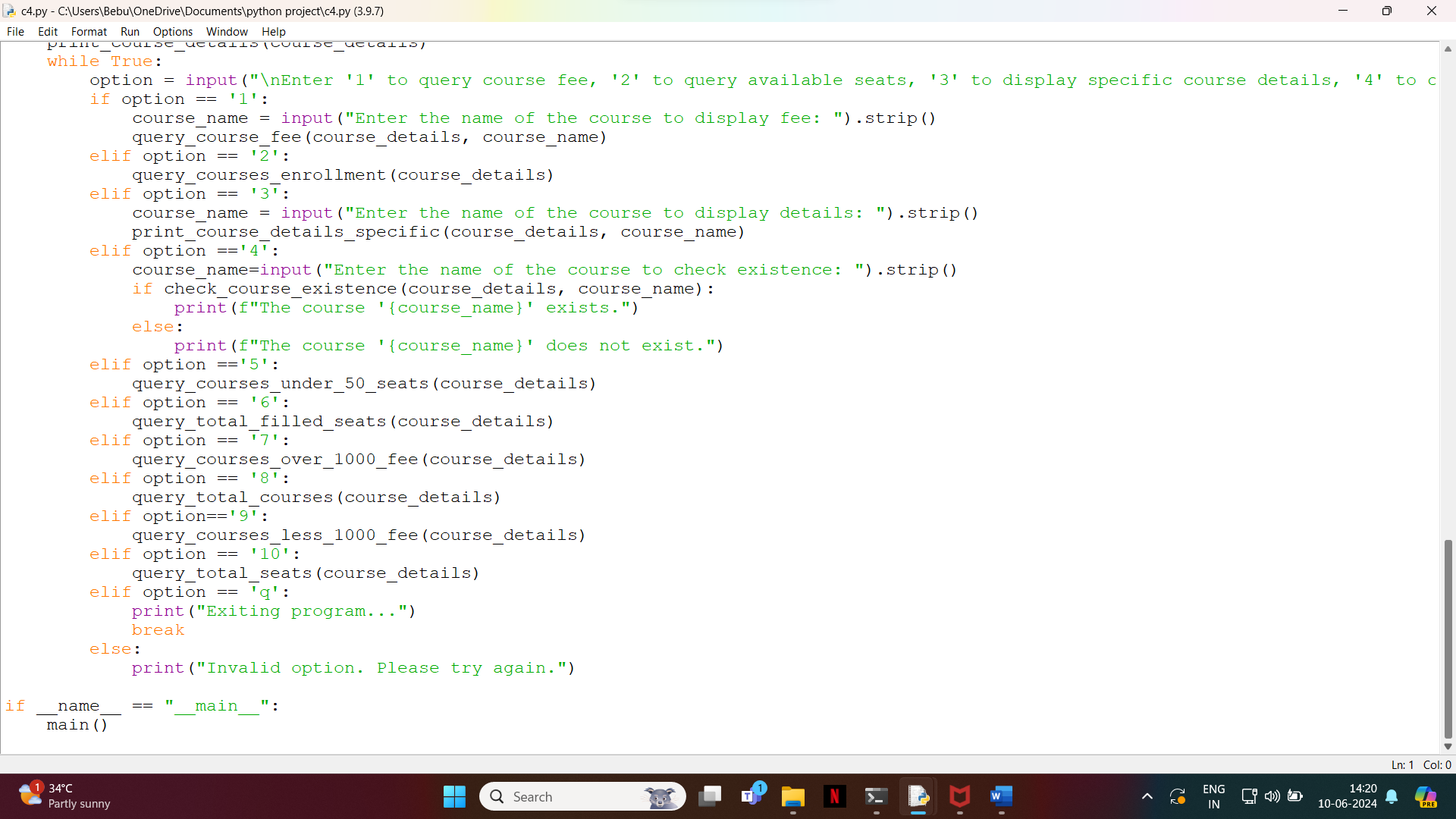
1. **Modular Design**: Ensure each function has a single responsibility, and avoid redundant functions.
2. **Error Handling**: Improve input validation and error handling to make the program more robust.
3. **User Interface**: Make the user interface more intuitive and responsive.
4. **Documentation**: Add comprehensive docstrings and comments for better understanding and maintainability.
5. **Code Cleanliness**: Remove redundant or duplicate code and ensure consistent formatting

**Program**:

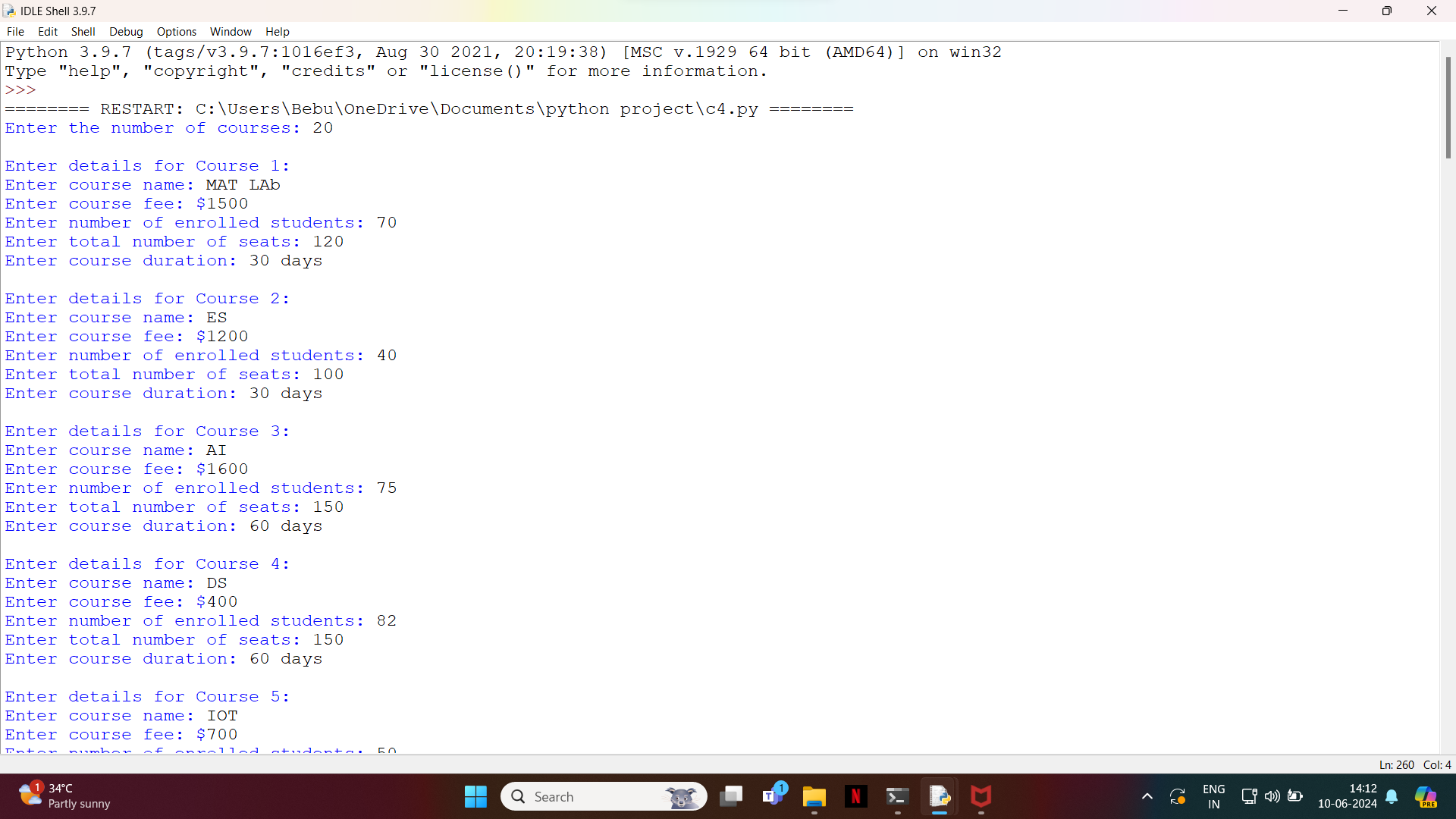


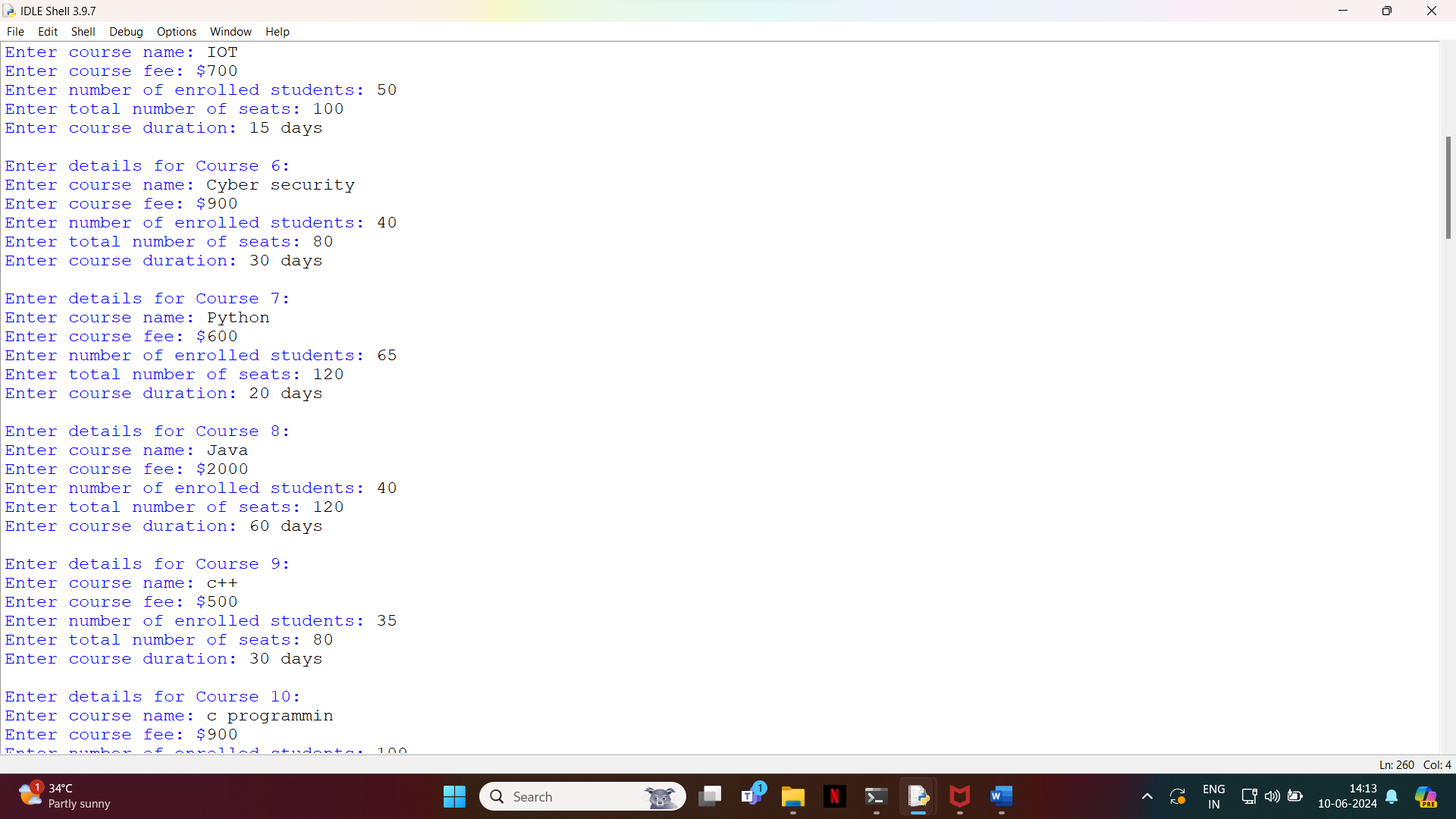


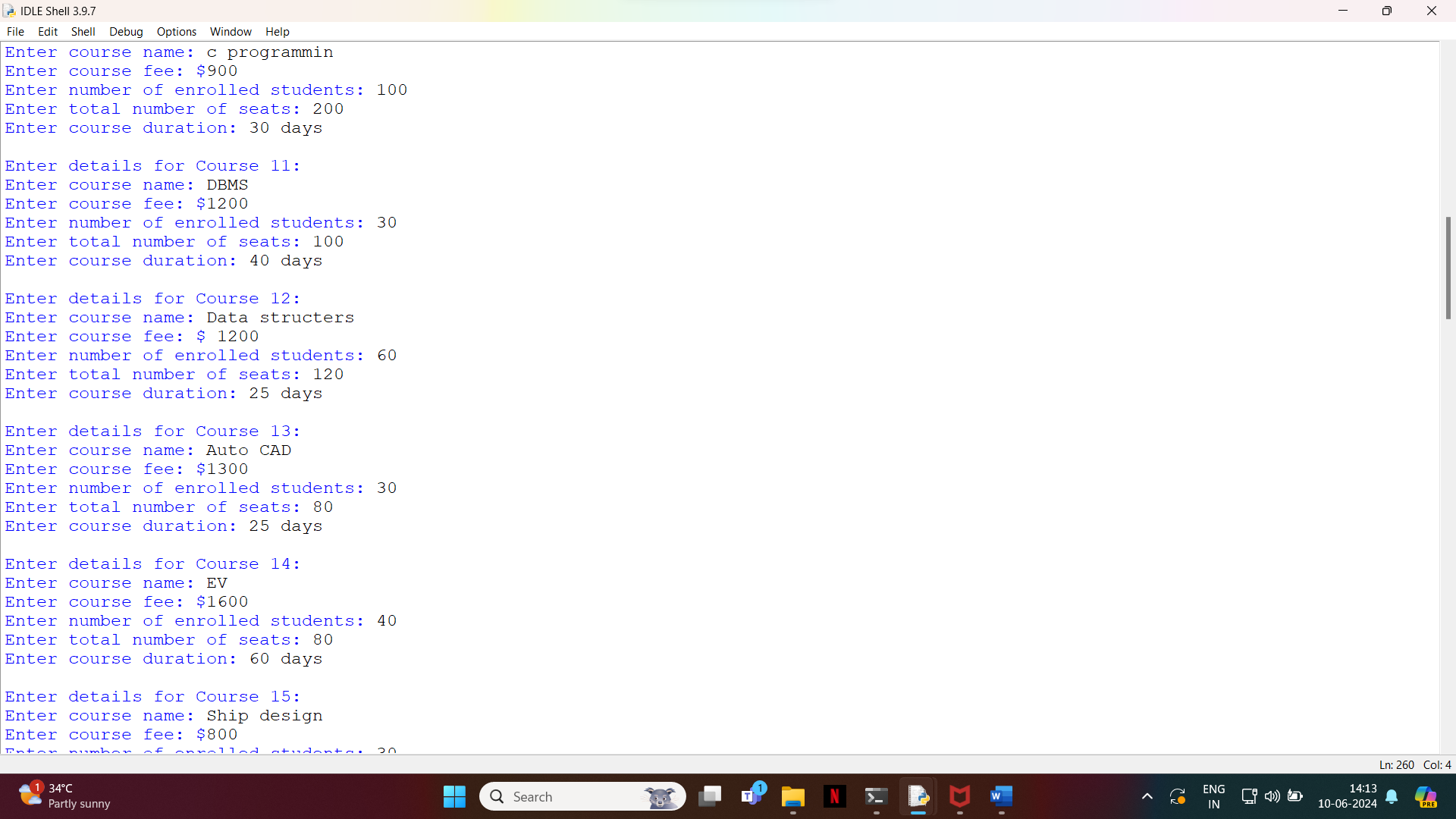


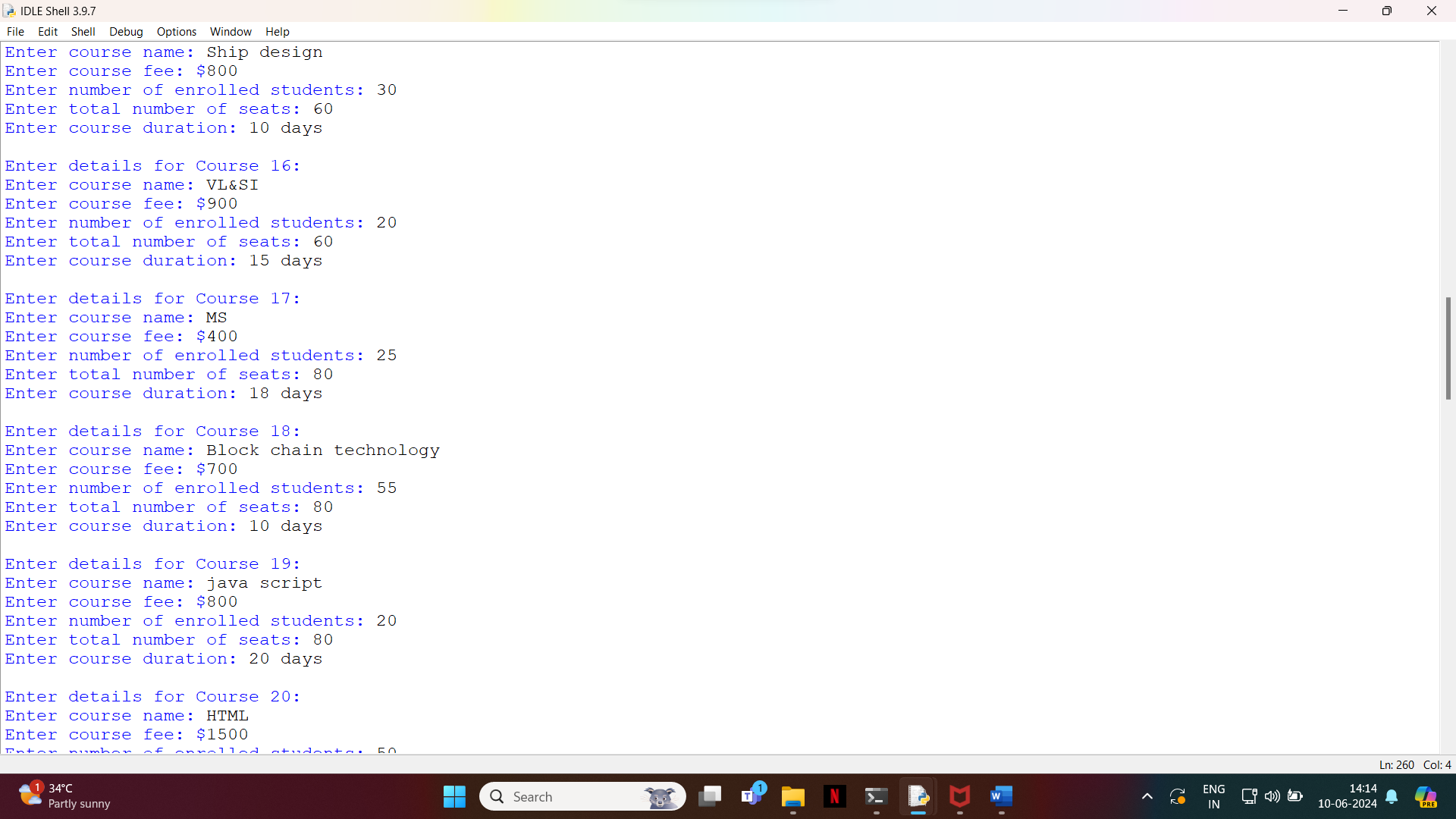


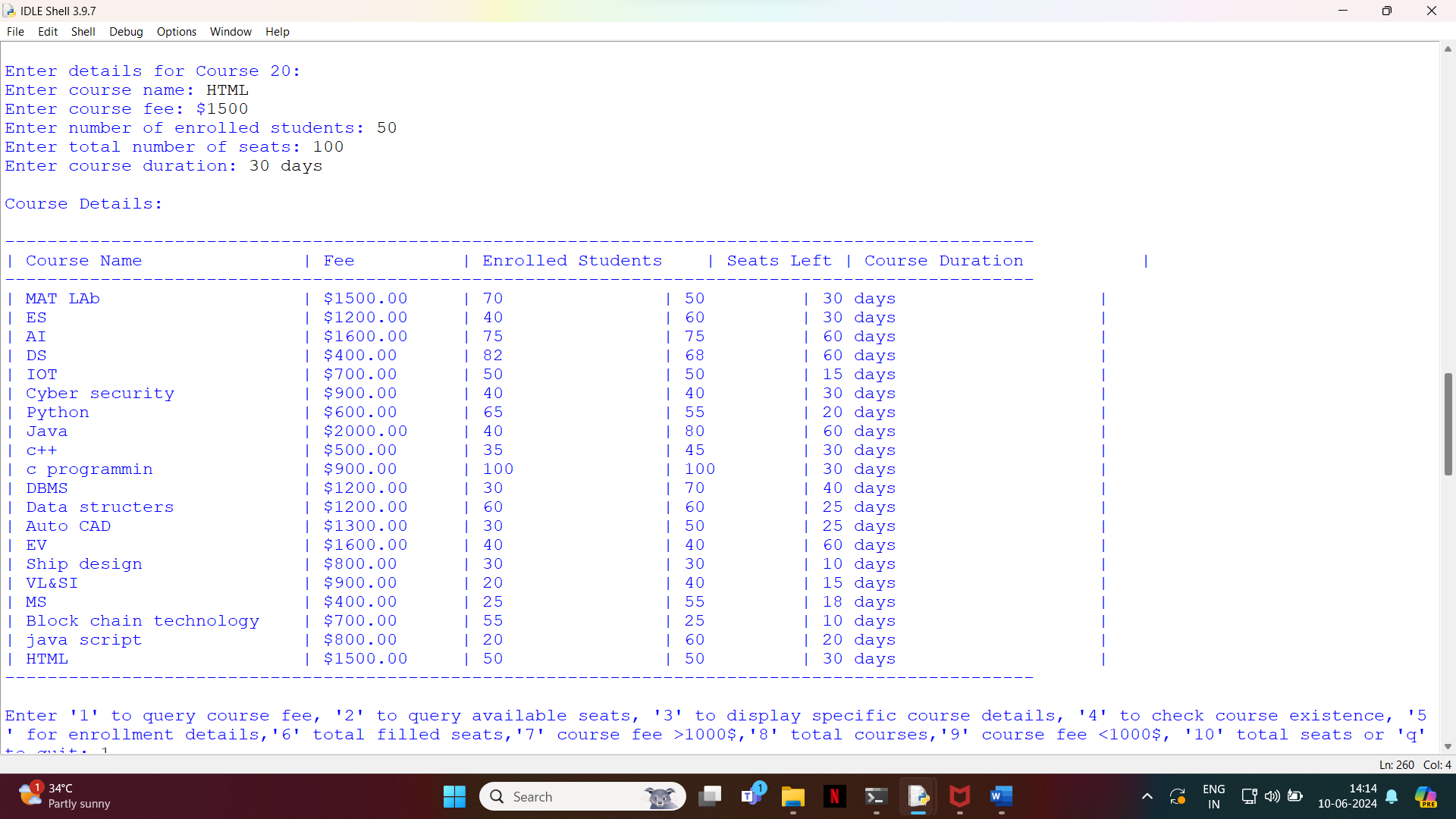
OUTPUT:

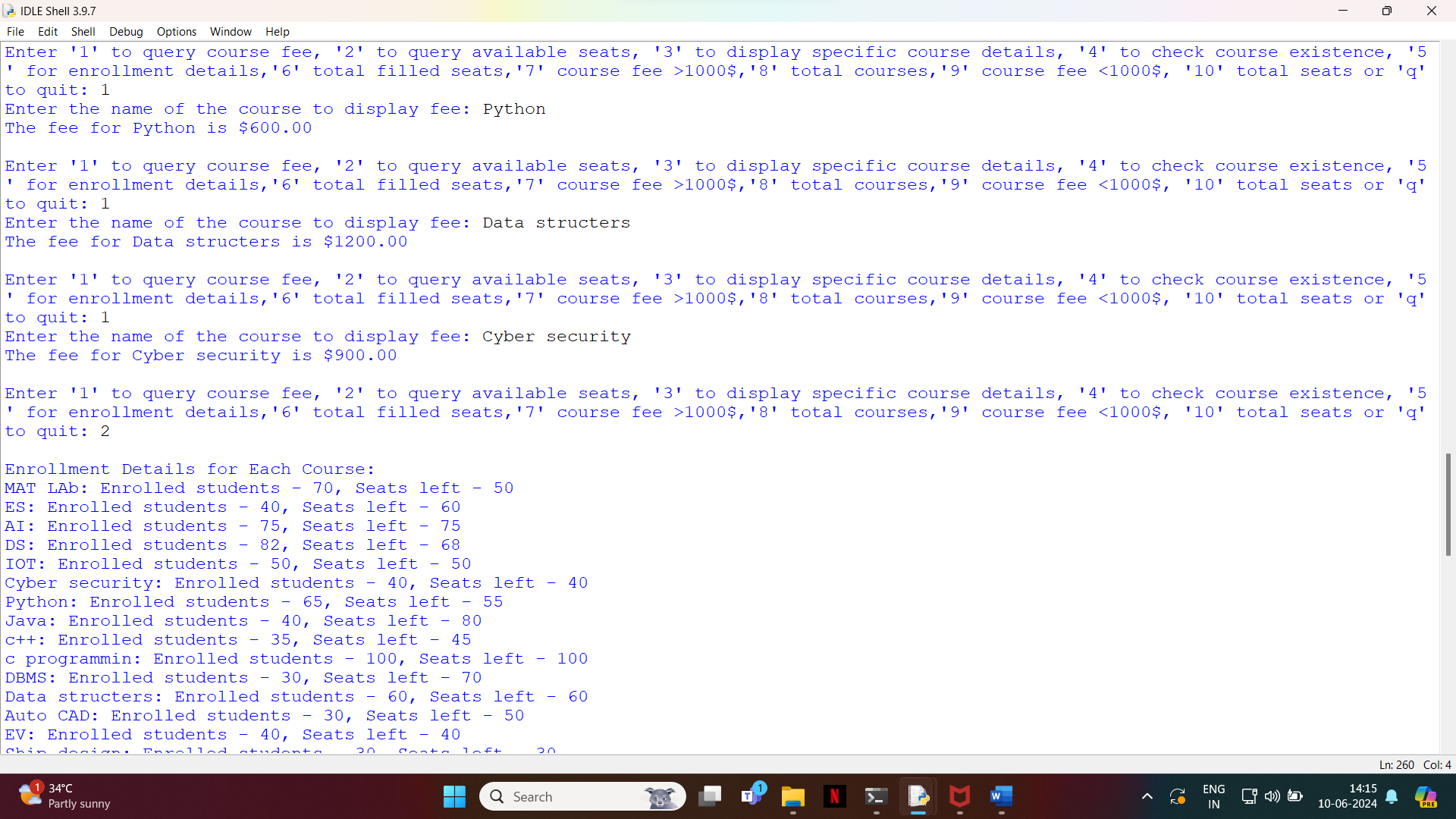












**Explanation**:

The provided code is a Python script for managing and querying course details. It allows users to input various details for multiple courses, and then perform different queries on these courses. Here is an explanation of the key components and functions of the code:

### Functions

1. **print\_course\_details(course\_details)**:
   * Prints a table with details of all the courses.
   * The table includes columns for course name, fee, enrolled students, seats left, and course duration.
2. **print\_course\_details\_specific(course\_details, course\_name)**:
   * Prints details of a specific course identified by course\_name.
   * Displays course name, fee, enrolled students, seats left, and duration if the course exists.
3. **query\_course\_fee(course\_details, course\_name)**:
   * Queries and prints the fee of a specific course identified by course\_name.
   * Displays "Course not found." if the course does not exist.
4. **query\_courses\_enrollment(course\_details)**:
   * Prints enrollment details for each course.
   * Shows the number of enrolled students and seats left for each course.
5. **check\_course\_existence(course\_details, course\_name)**:
   * Checks if a course identified by course\_name exists in the list.
   * Returns True if the course exists, False otherwise.
6. **query\_seats\_left(course\_details)**:
   * Prints the number of seats left for each course.
7. **query\_courses\_under\_50\_seats(course\_details)**:
   * Prints the count of courses with less than 50 total seats.
8. **query\_total\_filled\_seats(course\_details)**:
   * Prints the total number of filled seats across all courses.
9. **query\_courses\_over\_1000\_fee(course\_details)**:
   * Prints the count of courses with a fee greater than $1000.
10. **query\_courses\_less\_1000\_fee(course\_details)**:

* Prints the count of courses with a fee less than $1000.

1. **query\_total\_courses(course\_details)**:

* Prints the total number of courses.

1. **query\_total\_seats(course\_details)**:

* Prints the total number of seats across all courses.

**Conclusion**:

I concluded that the course navigator pro,This script provides a straightforward yet powerful way to manage and query course information. It demonstrates the practical use of Python for data management tasks and can be extended with additional features such as data persistence (saving and loading course data from a file or database), user authentication, or a graphical user interface (GUI) for enhanced usability.