### Lesson 8: Asynchronous Programming Basics

* **Reading**: Official Rust documentation and “The Rust Programming Language,” focusing on asynchronous programming concepts introduced in newer editions.
* **Assignments**: Create an asynchronous web scraper that concurrently downloads and processes web pages.
* **Preflight**: Study examples of asynchronous code in Rust, noting the use of async and await.
* **Lesson Goals**:
  + Grasp the basics of asynchronous programming in Rust.
  + Understand how async and await improve code readability and maintainability.
  + Experience the performance benefits of non-blocking IO operations.
* **Motivation**: Asynchronous programming is crucial for building scalable, high-performance applications, particularly in IO-bound and web-based services.
* **Lecture**:
  + Introduction to asynchronous execution in Rust and its advantages over synchronous code.
  + The event loop and executor: how async tasks are scheduled and run.
  + Error handling in async Rust.
* **Lab**: Implement the web scraper, focusing on handling concurrent downloads without blocking execution. The lab will reinforce async/await syntax and effective use of Rust’s futures.