Rust Learning Syllabus - 60 Days (1 Hour/Day)

Week 1-2: Foundations & Setup

Day 1-2: Getting Started

- Install Rust & Cargo
- Hello World program
- Understanding `cargo new`, `cargo build`, `cargo run`
- Basic println! macro

Day 3-4: Variables & Data Types

- Immutable vs mutable variables (`let` vs `let mut`)
- Scalar types: integers, floats, booleans, characters
- Compound types: tuples, arrays
- Type annotations

Day 5-7: Functions & Control Flow

- Function syntax and parameters
- Return values and expressions vs statements
- if/else conditions
- Loops: `loop`, `while`, `for`

Day 8-10: Ownership (First Encounter)

- What is ownership?
- Stack vs heap memory
- Move semantics

- Copy vs Clone

Day 11-14: References & Borrowing

- Immutable references (`&T`)
- Mutable references (`&mut T`)
- Borrowing rules (one mutable OR multiple immutable)
- Dangling references

Week 3-4: Core Concepts

Day 15-17: Slices

- String slices (`&str`)
- Array slices
- Understanding slice syntax

Day 18-21: Structs

- Defining structs
- Method syntax (`impl` blocks)
- Associated functions
- Tuple structs and unit structs

Day 22-25: Enums & Pattern Matching

- Defining enums
- `Option<T>` enum
- `match` expressions
- `if let` syntax

Day 26-28: Collections

- Vectors (`Vec<T>`)
- Strings (`String` vs `&str`)
- Hash maps (`HashMap<K, V>`)

Week 5-6: Error Handling & Modules

Day 29-32: Error Handling

- `Result<T, E>` enum
- `panic!` macro
- `unwrap()` and `expect()`
- `?` operator for error propagation

Day 33-35: Modules & Packages

- Module system (`mod`, `pub`)
- `use` keyword
- Creating libraries with `lib.rs`
- Cargo workspaces

Day 36-42: Advanced Ownership

- Lifetimes (`'a` syntax)
- Lifetime annotations in functions
- Lifetime elision rules
- `'static` lifetime

Week 7-8: Traits & Generics

Day 43-46: Generics

- Generic functions
- Generic structs
- Generic enums
- Type parameters

Day 47-50: Traits

- Defining traits
- Implementing traits
- Trait bounds
- Default implementations
- Derive macros

Day 51-53: Advanced Traits

- Associated types
- Trait objects (`dyn`)
- Operator overloading

Week 9: Practical Applications

Day 54-56: File I/O & CLI

- Reading/writing files
- Command line arguments
- Error handling in practice
- Building a CLI tool

Day 57-59: Concurrency Basics

- Threads (`std::thread`)

- Message passing (`mpsc`)
- Shared state (`Arc<Mutex<T>>`)

Day 60: Integration & Next Steps

- Review and consolidate
- Plan advanced topics
- Identify areas for deeper study

Daily Structure (1 Hour)

- **20 minutes:** Read/watch concept
- **30 minutes:** Code examples and exercises
- **10 minutes:** Write notes and reflect

Key Resources

- **The Rust Book** (official, free online)
- **Rustlings** (interactive exercises)
- **Rust by Example** (code-focused learning)

Important Notes

- **Expect frustration weeks 3-6** - this is normal!
- **The borrow checker will be your enemy then best friend**
- **Don't skip the ownership chapters** - everything builds on this
- **Write code every day** - reading isn't enough
- **Embrace compiler errors** - they're teaching you

Weekly Milestones

- **Week 2:** Can write basic programs without compiler errors

- **Week 4:** Understand ownership and borrowing concepts
- **Week 6:** Can handle errors properly and organize code
- **Week 8:** Can write generic, reusable code
- **Week 9:** Can build practical applications

After 60 Days, You'll Be Ready For:

- Web development with Actix/Warp
- System programming projects
- Contributing to open source Rust projects
- Advanced topics: async/await, macros, unsafe code