CLOUD CONTRAPTIONS
COMPUTER PROGRAMMING & TRAINING

To discuss this course and customizations:
Call: 434-509-5680 or Email: sales@cloudcontraptions.com

Rust Essentials

Duration

3 days

Description

The Rust Essentials course is designed to provide participants with a foundational understanding of the Rust programming language. This hands-on, beginner-friendly course covers key concepts such as ownership, borrowing, and lifetimes, enabling students to write safe and efficient code. Participants will learn how to create custom data structures, work with pattern matching, and harness the power of Rust's modern features. For experienced software developers looking to add Rust to your skillset, this course equips you with the essential knowledge needed to become proficient in one of the industry's most innovative and sought-after languages.

Objectives

- Understand the Rust Philosophy
- Set Up and Navigate the Rust Environment
- Grasp Basic Rust Syntax and Semantics
- Learn Control Flow and Logic
- Learn Ownership and Borrowing Concepts
- Utilize Tuples, Enums, Structs, and Vectors
- Employ Pattern Matching

Prerequisites

- Software development experience, this is not a general introduction to programming course.
- Basic understanding of programming concepts such as variables, expressions, functions, and control flow.

Training Materials

All students receive comprehensive courseware covering all topics in the course. Courseware is distributed via GitHub in the form of documentation and extensive code samples. Students practice the topics covered through challenging hands-on lab exercises.

CLOUD CONTRAPTIONS
COMPUTER PROGRAMMING & TRAINING

To discuss this course and customizations:
Call: 434-509-5680 or Email: sales@cloudcontraptions.com

Software Requirements

Students will need a free, personal GitHub account to access the courseware. Student will need permission to install Rust and Visual Studio Code on their computers. Also, students will need permission to install Rust Crates and Visual Studio Extensions. If students are unable to configure a local environment, a cloud-based environment can be provided.

Outline

- Introduction
- What is Rust?
  - Rust's Philosophy and Goals
  - History and motivation
  - Rust Community
  - The Rust Playground
- Install Rust
  - Script
  - macOS Homebrew
  - Platform Installers
- Rust Editors
  - VSCode with Extensions
  - Rust Rover
  - Debug Rust in VSCode
  - GitHub Copilot
- Hello World
  - Create a new Project
  - Main Function
  - Print to the Console
  - Comments
- Cargo
  - What is Cargo?
  - Run Command
  - Build Command
  - Build Release Command
  - Install Third-Party Crates
- Scalar Types and Data
  - Rust Types

  - Constants
  - Immutable Variables
  - Mutable Variables
- Code Logic
  - If Statement
  - Loop with Break
  - While Loop
- Functions
  - Define a Function
  - Call a Function
  - Parameter Types
  - Return Types
  - Closure Functions
- Modules
  - Import Modules from Standard Library
  - Import Modules from Third-Party Crates
  - Define Custom Modules
  - Import Custom Modules
- Built-In Macros
  - print! and println!
  - format!
  - vec!
  - include_str! and include_bytes!
  - cfg! and env!
  - panic!
- Memory Management
  - Problems with Manual Management
  - Problems with Garbage Collection
  - Ownership & Borrowing
  - References
  - Lifetimes
- Strings and String Slices
  - What is a String and a String Slice?
  - String Slices
  - String Objects

  - Convert Between Slices and Strings
  - Parse Number from String
  - Trim String
  - Print Strings with Interpolation
- Tuples
  - What is a Tuple?
  - Heterogeneous Elements
  - Access Elements
  - Destructuring
  - Immutable
- Enums
  - What is an Enum?
  - Define an Enum
  - Using Enums
  - Enum Variants
  - Enum Methods
  - Enums and Pattern Matching
  - Result Enum
  - Option Enum
  - Enums vs Structs
- Structs
  - What is a Struct?
  - Create Instance
  - Field Init Shorthand
  - Struct Update Syntax
  - Tuple Structs
  - Unit-Like Structs
  - Ownership of Struct Data
  - Function Implementation
  - Associated Functions
  - Stuct Methods
  - Constructor Pattern
- Vectors
  - What is a Vector?
  - Create a Vector

  - Add and Remove Elements
  - Access Elements
  - Iteration and iterators
  - Iterate over Elements
  - Slicing, Length, and Capacity
  - Common Vector Operations
  - Understand Memory Management
  - Ownership and Borrowing Rules
- Pattern Matching
  - What is Pattern Matching?
  - Match Statement
  - If Let Statement
  - While Let Statement
  - Destructuring Stucts and Tuples
  - Pattern Matching with Enums
  - Pattern Matching with Functions
  - Pattern Matching and Ownership
  - Refutability and Irrefutability
- Conclusion