Cloud Contraptions
TRAINING & CONSULTING

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

Short Course on Rust for Python Programmers

Class Duration

- 21 hours of live online training
- Delivered over 3 days

Target Audience

- Proficiency in Python programming
- Basic understanding of programming concepts such as variables, expressions, functions, and control flow

Description

This comprehensive course is designed for programming professionals who want to delve into the world of Rust. It begins with an introduction to Rust’s philosophy and community, and a comparison with Python. You’ll learn how to install Rust and set up your development environment, write your first Rust program, and understand the role of Cargo, Rust’s package manager. The course then dives deep into the differences between Rust and Python, Rust’s scalar types, data structures, code logic, functions, modules, and built-in macros. You’ll also explore Rust’s memory management, strings, tuples, enums, structs, vectors, collections, iterators, traits, and generics. Finally, you’ll learn about pattern matching and concurrent programming in Rust. This course is a must for anyone looking to expand their programming skills and knowledge.

Learning Objectives

- Understand the philosophy, history, and community behind Rust, and how it compares to Python.
- Learn how to install Rust and set up a Rust development environment.
- Write your first Rust program and understand the role of Cargo, Rust’s package manager.
- Explore the differences between Rust and Python, including static vs dynamic typing, memory management, and error handling.
- Learn about Rust’s scalar types, data structures, and code logic.
- Understand how to define and use functions, modules, and built-in macros in Rust.

- Dive deep into Rust’s memory management, strings, tuples, enums, structs, vectors, collections, iterators, traits, and generics.
- Learn about pattern matching and concurrent programming in Rust.

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.

Software Requirements

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

Training Topics

Introduction

- What is Rust?
- Rust’s Philosophy and Goals
- History and motivation
- Rust vs Python
- Rust Community
- The Rust Playground

Install Rust

- RustUp Script
- macOS Homebrew
- Platform Installers

Rust Editors

- VSCode with Extensions
- Rust Rover & Zed
- Debug Rust in VSCode
- GitHub Copilot

Hello World

- Create a new Project
- Main Function
- Print to the Console
- Comments

Cargo

- What is Cargo?
- How does Cargo compare to Pip and Conda?
- Rust Crates compared to Python Packages
- Run Command
- Build Command
- Build Release Command
- Install Third-Party Crates

Popular Cargo Crates

- Serde
- Tokio
- Reqwest
- SQLx
- Anyhow

Rust and Python Differences

- Static Typing vs Dynamic Typing
- Memory Management
- Error Handling
- Sequence, Selection, and Iteration
- Structs vs Classes
- Traits vs Protocols
- Generics
- Concurrency

Scalar Types and Data

- Rust Types vs Python 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
- Rust vs Python
- References
- Immutable vs Mutable
- Lifetimes
- Heap Allocation with Box and Rc
- Dynamic Dispatch
- Drop Trait

Strings

- 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
- Struct Methods

- Constructor Pattern

Vectors

- What is a Vector?
- Create a Vector
- Add and Remove Elements
- Access Elements
- Iterate over Elements
- Slicing, Length, and Capacity
- Common Vector Operations
- Understand Memory Management
- Ownership and Borrowing Rules

Collections and Iterators

- Vectors, arrays, and slices
- HashMaps and hash sets
- Iteration and iterators

Traits

- What is a trait?
- How does a trait related to traditional OOP interfaces?
- Defining a trait
- Implementing a trait
- Default implementations
- Traits as parameters
- Traits as return types
- Traits as bounds

Generics

- What is a generic?
- How does a generic related to traditional OOP generics?
- Defining a generic
- Implementing a generic
- Generic bounds
- Multiple generic types
- Where clauses

Pattern Matching

- What is Pattern Matching?
- Match Statement
- If Let Statement
- While Let Statement
- Destructuring Structs and Tuples
- Pattern Matching with Enums
- Pattern Matching with Functions
- Pattern Matching and Ownership
- Refutability and Irrefutability

Concurrent Programming

- What is Concurrent Programming?
- Using Multiple Threads
- Mutex, RwLock, and Arc
- Message Passing with Channels
- Sync and Send Traits
- Futures and Async/Await

Conclusion

- Course wrap-up and next steps