coursera

# Functional Programming Principles in Scala

École Polytechnique Fédérale de Lausanne

About this Course
Functional programming is becoming increasingly widespread in industry. This trend is driven by the adoption of Scala as the main programming language for many applications. Scala fuses functional and object-oriented programming in a practical package. It interoperates seamlessly with both Java and Javascript. Scala is the implementation language of many important frameworks, including Apache Spark, Kafka, and Akka. It provides the core infrastructure for sites such as Twitter, Tumblr and also Coursera.

In this course you will discover the elements of the functional programming style and learn how to apply them usefully in your daily programming tasks. You will also develop a solid foundation for reasoning about functional programs, by touching upon proofs of invariants and the tracing of execution symbolically.

The course is hands on; most units introduce short programs that serve as illustrations of important concepts and invite you to play with them, modifying and improving them. The course is complemented by a series programming projects as homework assignments.

Learning Outcomes. By the end of this course you will be able to:

- understand the principles of functional programming,
- write purely functional programs, using recursion, pattern matching, and higher-order functions,
- combine functional programming with objects and classes,
- design immutable data structures,
- reason about properties of functions,
- understand generic types for functional programs

Recommended background: You should have at least one year programming experience. Proficiency with Java or C# is ideal, but experience with other languages such as C/C++, Python, Javascript or Ruby is also sufficient. You should have some familiarity using the command line.

Show less

Taught by: Martin Odersky, Professor
Computer Science

Help Center

| | |
|---|---|
| **Basic Info** | Course 1 of 5 in the Functional Programming in Scala Specialization . |
| **Level** | Intermediate |
| **Language** | English, **Subtitles:** Serbian Volunteer to translate subtitles for this course |
| **How To Pass** | Pass all graded assignments to complete the course. |
| **User Ratings** | ★★★★⯪ 4.8 stars |

## Syllabus

WEEK 1

### Getting Started + Functions & Evaluation

Get up and running with Scala on your computer. Complete an example assignment to familiarize yourself with our unique way of submitting assignments. In this week, we'll learn the difference between functional imperative programming. We step through the basics of Scala; covering expressions, evaluation, conditionals, functions, and recursion Less

11 videos, 8 readings

1. **Video:** Course Introduction
2. **Video:** Tools Setup for Linux
3. **Video:** Tools Setup for Mac OS X
4. **Video:** Tools Setup for Windows
5. **Ungraded Programming:** Example Assignment
6. **Reading:** Tools Setup (Please read)
7. **Reading:** Cheat Sheet
8. **Reading:** IntelliJ IDEA Tutorial
9. **Reading:** Eclipse Tutorial
10. **Reading:** SBT tutorial and Submission of Assignments (Please read)

Help Center

11. **Reading:** Learning Resources
12. **Reading:** Scala Tutorial
13. **Reading:** Scala Style Guide
14. **Video:** Lecture 1.1 - Programming Paradigms
15. **Video:** Lecture 1.2 - Elements of Programming
16. **Video:** Lecture 1.3 - Evaluation Strategies and Termination
17. **Video:** Lecture 1.4 - Conditionals and Value Definitions
18. **Video:** Lecture 1.5 - Example: square roots with Newton's method
19. **Video:** Lecture 1.6 - Blocks and Lexical Scope
20. **Video:** Lecture 1.7 - Tail Recursion

**Graded:** Recursion

WEEK 2

## Higher Order Functions

This week, we'll learn about functions as first-class values, and higher order functions. We'll also learn about Scala's syntax and how it's formally defined. Finally, we'll learn about methods, classes, and data abstraction through the design of a data structure for rational numbers. Less

7 videos

1. **Video:** Lecture 2.1 - Higher-Order Functions
2. **Video:** Lecture 2.2 - Currying
3. **Video:** Lecture 2.3 - Example: Finding Fixed Points
4. **Video:** Lecture 2.4 - Scala Syntax Summary
5. **Video:** Lecture 2.5 - Functions and Data
6. **Video:** Lecture 2.6 - More Fun With Rationals
7. **Video:** Lecture 2.7 - Evaluation and Operators

**Graded:** Functional Sets

Help Center

WEEK 3

## Data and Abstraction

This week, we'll cover traits, and we'll learn how to organize classes into hierarchies. We'll cover the hierarchy of standard Scala types, and see how to organize classes and traits into packages. Finally, we'll touch upon the different sorts of polymorphism in Scala. Less

3 videos

1. **Video:** Lecture 3.1 - Class Hierarchies
2. **Video:** Lecture 3.2 - How Classes Are Organized
3. **Video:** Lecture 3.3 - Polymorphism

**Graded:** Object-Oriented Sets

WEEK 4

## Types and Pattern Matching

This week we'll learn about the relationship between functions and objects in Scala; functions *are* objects! We'll zoom in on Scala's type system, covering subtyping and generics, and moving on to more advanced aspects of Scala's type system like variance. Finally, we'll cover Scala's most widely used data structure, Lists, and one of Scala's most powerful tools, pattern matching. Less

7 videos

1. **Video:** Lecture 4.1 - Objects Everywhere
2. **Video:** Lecture 4.2 - Functions as Objects
3. **Video:** Lecture 4.3 - Subtyping and Generics
4. **Video:** Lecture 4.4 - Variance (Optional)
5. **Video:** Lecture 4.5 - Decomposition
6. **Video:** Lecture 4.6 - Pattern Matching
7. **Video:** Lecture 4.7 - Lists

**Graded:** Huffman Coding

Help Center

WEEK 5

### Lists

This week we dive into Lists, the most commonly-used data structure in Scala.

7 videos

1. **Video:** Lecture 5.1 - More Functions on Lists
2. **Video:** Lecture 5.2 - Pairs and Tuples
3. **Video:** Lecture 5.3 - Implicit Parameters
4. **Video:** Lecture 5.4 - Higher-Order List Functions
5. **Video:** Lecture 5.5 - Reduction of Lists
6. **Video:** Lecture 5.6 - Reasoning About Concat
7. **Video:** Lecture 5.7 - A Larger Equational Proof on Lists

WEEK 6

### Collections

After a deep-dive into Lists, this week we'll explore other data structures; vectors, maps, ranges, arrays, and more. We'll dive into Scala's powerful and flexible for-comprehensions for querying data.

6 videos

1. **Video:** Lecture 6.1 - Other Collections
2. **Video:** Lecture 6.2 - Combinatorial Search and For-Expressions
3. **Video:** Lecture 6.3 - Combinatorial Search Example
4. **Video:** Lecture 6.4 - Maps
5. **Video:** Lecture 6.5 - Putting the Pieces Together
6. **Video:** Conclusion

**Graded:** Anagrams

View Less

Help Center

## How It Works

GENERAL

**1. How do I pass the course?**

To earn your Course Certificate, you'll need to earn a passing grade on each of the required assignments—these can be quizzes, peer-graded assignments, or programming assignments. Videos, readings, and practice exercises are there to help you prepare for the graded assignments.

**2. What do start dates and end dates mean?**

Most courses have sessions that run multiple times a year — each with a specific start and end date. Once you enroll for a Certificate, you'll have access to all videos, readings, quizzes, and programming assignments (if applicable). Peer-graded assignments can only be submitted and reviewed once your session has begun. If you choose to explore the course without purchasing, you may not be able to access certain assignments. If you don't finish all graded assignments before the end of the session, you can enroll in the next session. Your progress will be saved and you'll be able to pick up where you left off when the next session begins.

**3. What are due dates? Is there a penalty for submitting my work after a due date?**

Within each session there are suggested due dates to help you manage your schedule and keep coursework from piling up. Quizzes and programming assignments can be submitted late without consequence. However, it is possible that you won't receive a grade if you submit your peer-graded assignment too late because classmates usually review assignment within three days of the assignment deadline.

**4. Can I re-attempt an assignment?**

Yes. If you want to improve your grade, you can always try again. If you're re-attempting a peer-graded assignment, re-submit your work as soon as you can to make sure there's enough time for your classmates to review your work. In some cases you may need to wait before re-submitting a programming assignment or quiz. We encourage you to review course material during this delay.

Show less

PROGRAMMING ASSIGNMENTS

**Programming assignments require you to write and run a computer program to solve a problem.**

Help Center

**1. What are programming assignments?**

Programming assignments include both assignment instructions and assignment parts. Instructions may include a link to a downloadable starter package that includes starter code, detailed guidelines, and other resources. Assignment parts are similar to individual quiz questions. Each part is a single coding task that can be completed one at a time.

**2. How are programming assignments graded?**

Programming assignments are graded automatically. If they use a built-in-algorithm you'll see your grade within seconds. If they use a custom grader, you may need to wait up to an hour.

**3. Can I resubmit a programming assignment?**

You can resubmit all programming assignments to improve your grade. Follow the same steps as submitting a new assignment.

**4. What do I do if I have trouble submitting my assignment?**

If you have trouble submitting your assignment, we encourage you to visit your course Discussion Forums as many of your peers are likely to have had similar problems and have found a solution. Each programming assignment has its own sub-forum to discuss with peers.

Show less

## Course 1 of Specialization

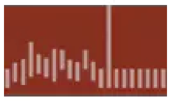
**Program on a Higher Level.** Write elegant functional code to analyze data that's big or small

Functional Programming in Scala

École Polytechnique Fédérale de Lausanne

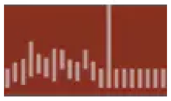
View the course in catalog

## Related Courses



Algorithms, Part II

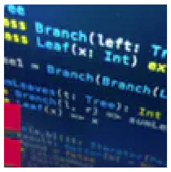
Princeton University



Algorithms, Part I

Princeton University

Help Center

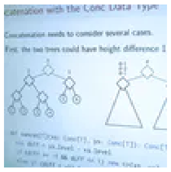


Functional Program Design in Scala

École Polytechnique Fédérale de Lausanne

Machine Learning

Stanford University



Parallel programming

École Polytechnique Fédérale de Lausanne

Help Center