Introduction to Julia Programming for Developers

Course Summary

Description

Julia was developed by researchers at MIT and other universities who explained why they started the Julia Project in 2009:

“We are greedy: we want more. We want a language that’s open source, with a liberal license. We want the speed of C with the dynamism of Ruby. We want a language that’s homoiconic, with true macros like Lisp, but with obvious, familiar mathematical notation like Matlab. We want something as usable for general programming as Python, as easy for statistics as R, as natural for string processing as Perl, as powerful for linear algebra as Matlab, as good at gluing programs together as the shell. Something that is dirt simple to learn, yet keeps the most serious hackers happy. We want it interactive and we want it compiled. Did we mention it should be as fast as C?”

Since then, Julia – maintained primarily through the Julia lab at MIT – has accomplished that goal and is now making significant inroads in the advanced computing community. From artificial intelligence, big data analytics and heavy duty numerical processing, Julia represents a modern high performance replacement for traditional number crunching languages like Fortran but also for modern slower performance data analytic languages like R, Python, Octave and a range of other specialized languages. Julia also is designed to integrate with code written in other languages including legacy C code, Fortran code and even Python code.

Jumping into Julia can be intimidating because of the advanced nature of the language and its sheer scope and size. This course is designed to help programmers develop a solid foundation in the syntax, semantics and architecture of Julia necessary for mastery of the more advanced features of the language (like the powerful data analytic packages).

The course covers the Julia core basics with an overview into the larger Julia ecosystem with the goal of helping programmers leverage their existing programming skills and knowledge seamlessly and painlessly into the Julia environment. The course also provides a basic overview of the advanced features for high performance computing and the libraries available to the Julia developer as a preparation for more advanced work in Julia.

The class is designed to be about 50% hands on labs and exercises, about 25% theory and 25% instructor led hands on learning where students code along with the instructor.

Topics

- The design goals and overall structure of the Julia language.
- Variable and data types – elementary operations and regular expressions.
- Defining and using functions in Julia – functions as first class objects.
- Anonymous functions and closures, using recursion.
- Maps, filters and list comprehensions.
- Generic functions and multiple dispatch.
- Basic control structures, error and exception handling.
- Collections – matrices, tuples, dictionaries, and sets.
- Types – defining types, hierarchies, user defined types, parametric types and methods.
- Concurrent and parallel computing.
- I/O – files, sockets, using data frames and databases.
- Calling C, Fortran, Python and shell scripts.
- Overview of the standard libraries and package management.
- Meta-programming with macros.
- Speed, performance, bench marking and profiling in Julia.
- Overview of scientific computing, working with data and graphics.
- Overview of high performance Julia – optimizations and parallelism.
- Overview of advanced networking and advanced Julia features.
Introduction to Julia Programming for Developers

Course Summary (cont’d)

Audience

This course is designed for programmers who want to get started programming in Julia.

Prerequisites

Before taking this course, an intermediate level of skill in and a solid knowledge of a high level programming language like Java, Python, or C is essential. Students who do not have this prerequisite may struggle with the content and pace of the course.

Duration

Three days