

**Introduction to Julia for Data Scientists**

**Course Number:** JUL-102
**Duration:** 3 days

**Overview**

This Introduction to Julia for Data Scientists training course teaches attendees how to perform daily data science tasks with the Julia programming language, including setting up the environment, working with all popular data formats and databases, and calling packages from other languages. In addition, participants explore data frames, data visualization, machine learning, and optimization.

**Prerequisites**

All students must have previous hands-on knowledge of any data science-oriented programming language (e.g., Python, R, or MATLAB). However, it is possible to adjust the course for attendees without programming or data science experience.

**Materials**

All Julia programming training students receive comprehensive courseware.

**Software Needed on Each Student PC**

Students must have a modern web browser and Internet access. Accelebrate will provide installation instructions (for Windows, Linux, and macOS).

**Objectives**

* Use Julia for their daily data science workflows
* Work with all popular data formats
* Pre- and post-process data with Julia
* Visualize data
* Build machine learning and optimization models

**Outline**

* Introduction
* Installing and Running Julia
	+ Installation worth-to-know issues
	+ Julia with Jupyter notebook
	+ Julia with Juno IDE
* Julia Type System
	+ Tuples, arrays, dictionaries, and complex data structures
	+ Functions, optional typing, and multiple dispatch
* File IO
	+ CSV
	+ JSON
	+ Excel
	+ Web scraping
* Julia Integration with Other Programming Languages
	+ Python
	+ R
	+ Matlab (optional for teams running legacy Matlab codes)
* Visualizing Data
	+ PyPlot
	+ Plots.jl
* Data and Machine Learning
	+ Working with databases (MySQL/MariaDB/PostgreSQL/MS SQL/Oracle/Elasticsearch – choose one)
	+ Manipulating data frames DataFrames.jl
	+ Machine learning ScikitLearn.jl
* Optimization with JuMP.jl
	+ Building optimization models with JuMP
	+ Linear, MIP and nonlinear models with JuMP
* Conclusion