Julia
Programming Language

January 2023

Stefan Penzinger

Overview

- numerical/technical computing
- general purpose
- web development
- About as fast as C → but JIT

https://julialang.org/downloads/
Current stable release: v1.8.5 (January 8, 2023)

- Julia has a set of supported Platforms categorized in different tiers, which indicate if Julia compiles to a 100% or not given the specific resources of the destination computer

General

- Started in 2009 by Jeff Bezanson, Stefan Karpinski, Viral B. Shah, and Alan Edelman
- First blog post in 2012 - where he stated why it is called Julia
  - in 2012 also the launch of pre-1.0 Julia happened
- In the next 10 years the community grew
- 2019 the inventors claimed the " James H. Wilkinson Prize for Numerical Software" award for (Julia 1.1 at that time)
  - "for the creation of Julia, an innovative environment for the creation of high-performance tools that enable the analysis and solution of computational science problems."
- January 2023 - Julia 1.8.5

Language Overview (1)

- high-level, dynamic programming lanugage
- parametric polymorphism → multiple dispatch (core programming paradigm)
- Uses the JIT compiler, or also called JAOT in the community
  - Julia compiles code by default to machine code before running it.
- supports concurrent, (composable) parallel and distributed computing
- direct calling of C and Fortran libraries without glue code

Language Overview (2)

- garbage-collected
- uses eager evaluation
- includes efficient libraries for
  - floating-point calculations
  - linear algebra
  - random number generation
  - regular expression matching

Features

- Dynamically typed
  - assigns a type to all the variables at run-time by its value
- Build-in package manager
- Metaprogramming
  - code represented as data structure
- Interface with other languages like C, R, Scala/Java and Python
- **Multiple dispatch**

Single/Dynamic Dispatch

- Choice of which method version is called is defined by the object it executes
- Supported by common object-oriented languages

- typed language
  - the dispatch mechanism will be performed based on the type of the arguments (mostly type of the receiver of a message).
- weak / non typed language
  - carry a dispatch table as part of the object (for each object)
  - instance behaviour

Multiple Dispatch

- generalization of single-dispatch polymorphism
- dispatch based on combination of multiple arguments

```
abstract type Person end
abstract type Tourist <:Person end
abstract type Deer end

encounter(a::Deer, b::Tourist) = "bows politely"
encounter(a::Tourist, b::Deer) = "feeds"
encounter(a::Person, b::Deer) = "beckons"
encounter(a::Deer, b::Person) = "ignores"
encounter(a::Tourist, b::Deer, foo::String) = "feeds deer $foo"
```

Syntax(1)

if - else if - else

```
if condition
    Statement
else
    Statement
end
```

```
if A[1,2] == 5
 println("This true")
else
 println("This is false")
end
```

Syntax(2)

for loop

```
for i in 0: 10: 100
    Print(i)
end
```

Output:
0 10 20 30 40 50 60 70 80 90 100

```
for a in [“red”, “green”, “yellow”]
    Print(a, “ “)
end
```

red green yellow

Syntax(3)

Generator expression

```
sum(x^2 for x in 1:10)
```

Output:
385

Syntax(4)

Matrix

```
A = [1 2 3; 4 5 6; 7 8 9]
println(A)
A[1,2] = 5
println(A[1,2])

println(transpose(A))
```

```
[1 2 3; 4 5 6; 7 8 9]

5

[1 4 7; 5 5 8; 3 6 9]
```

Areas of Application

Data science

Companies developing in Julia.

- Intel
- Disney
- Amazon
- Google
- Microsoft
- NASA
- IBM

Examples(1)

```
using DataFrames
using CSV
using Plots

df = CSV.read("trees.csv", DataFrame)
print(describe(df))
```

```
4×7 DataFrame
 Row │ variable      mean     min    median   max   nmissing  eltype
     │ Symbol        Float64  Real   Float64  Real  Int64     DataType
─────┼──────────────────────────────────────────────────────────────────
   1 │ Index         16.0      1        16.0  31           0  Int64
   2 │ Girth (in)    13.2484   8.3      12.9  20.6         0  Float64
   3 │ Height (ft)   76.0     63        76.0  87           0  Int64
   4 │ Volume(ft^3)  30.171   10.2      24.2  77.0         0  Float64
```

Examples(2)

Distribution of tree height

```
using Plots
using RDatasets, StatsPlots

@df df boxplot(:"Height (ft)")
```