5Minds IT-SOLUTIONS

Julia

Frank von der Höh, Gelsenkirchen, 29.05.2020

Agenda

- Overview
- Variables
- Functions
- Type system
- Methods

Overview

Overview

- First release in 2012
- Intended to combine best of C, Ruby, Lisp, Matlab, Python, R, Perl
- Idea: Speed and ease of use in one language
- General purpose, but focus on scientific computing
- Interop
  - Call C and Fortran libraries directly
  - Others (e.g. C++, Java, R, MATLAB) through integration packages
- Integrated REPL
- Integrated package manager

Overview

- Parallel Computing
  - Coroutines/Tasks (not multi-threaded)
  - Multi-Threading (Experimental)
  - Distributed/Multi-core
- Metaprogramming
- IDE: Juno (based on Atom) or VS Code
- Debugger (since 03/2019)
- Linter

Variables

Variables

- Name bound to a value => Dynamically typed
- Redefinition of built-in constants and functions
    - Possible, but discouraged
    - Only valid until usage

Functions

Functions

- Value of last expression is automatically returned, but explicit return is possible
- Functions are objects => can be assigned to variables and passed around
- Most operators are functions
- Keyword arguments
- Functions can be composed using the operator “∘”
- Piping using “|>”

Functions - Dot operator

- Vectorized functions common in scientific computing
  => Any function in Julia can be applied to a vector by using dot operator
- Actually performs ‘broadcast’ which expands dimensions of arguments to match other arguments
- Can be shortened using macro ‘@.’ for multiple calls
- Can be combined with piping

Type System

Type System

- Dynamic, but supports type annotations
- Nominative
- Parametric polymorphism
- Abstract and final types only
- Primitives = concrete types whose data consists of bits
  - Only multiples of 8 allowed, e.g. Bool is an Integer 8
- Composite types (structs) are a collection of named fields, immutable by default, but can be declared to be mutable

Type System

- Union types
- Tuple types (unnamed, named and variable)
- Singleton type (needed for methods with behavior not depending on argument types alone, Trait-based dispatch)

Methods

Methods

- Methods with the same name and different argument lists make up a function
- Multiple dispatch
  - Best-fitting method is chosen at runtime
  - Different from static overloading
  - Different from OOP, because methods do not “belong” to a type
- Parametric methods

Links

- https://docs.julialang.org/en/v1/
- https://julialang.org/blog/2012/02/why-we-created-julia/
- https://julialang.org/blog/2019/03/debuggers/

5Minds IT-SOLUTIONS