Introduction to Computational Science in Julia

Tianzhang Cai, Dominic Orchard

UNIVERSITY OF CAMBRIDGE

Institute of Computing for Climate Science

ICCS Summer School 2024, Friday 12th July

https://github.com/Cambridge-ICCS/Summer-school-Julia-tutorial

- Est. 2009, rising in popularity in the sciences, particular **climate science** (see Clima)
- **Numerical** focus
- Python-like in various ways (syntax+semantics), but stronger emphasis on **speed**

Our goal: give you a crash-course intro today

- **Multi-paradigm language**: imperative, functional

Programs comprise sequence of (side-effectful) **statements** and definitions

```
x = 2
x = x + 1
```

Functions are first-class: treated as data

∴ can be returned

```
function mkConstantFunction(x)
  function inner(_y)
    x
  end
end
```

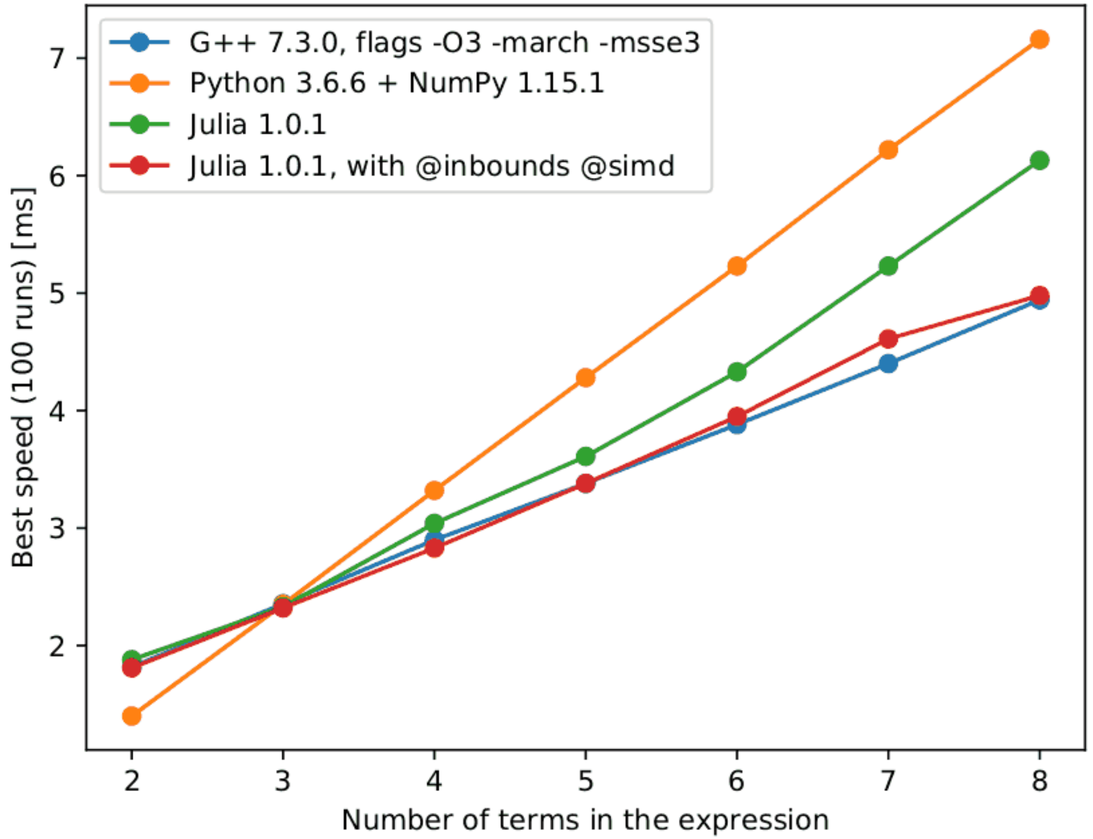
∴ can be passed as arguments

```
function applyTwice(f, x)
   f(f(x))
end
```

- **High performance** via JIT compilation to LLVM

```
@code_llvm 1+2
```

```
;  @ int.jl:87 within `+`
define i64 @"julia_+_2667"(i64 signext %0, i64 signext %1) #0 {
top:
  %2 = add i64 %1, %0
  ret i64 %2
}
```



https://discourse.julialang.org/t/comparing-python-julia-and-c/17019/9

- **Dynamically typed** with explicit optional type signatures (used by JIT compilation)

```
function plusInt(x :: Int)
  x + 1
end
```

```
function plusGen(x)
  x + 1
end
```

- Open name spaces, **multiple dispatch**

```
function scale(x :: Int, y :: Int) :: Int
  x * y
end

function scale(x :: Int, y :: String) :: String
  join([y for i in 1:x])
end
```

- …with operator overloading leads to highly **generalisable programming**

Learning objectives for today

- Understand the core ideas of Julia programming
- Use Julia to solve simple numerical programming tasks
- Working with Julia's type systems and data structures
- Leverage multiple dispatch, including for overloading programs, e.g., to symbolic form
- Put these ideas together into a **simple energy balance model**

Style: code along + exercises

Pluto.jl

Result shown above

Basic Calculation

```
27
(1 + 2) * 3 ^ 2
```

Cell contains a single expression

Evaluate cell in focus via

Reactive: if you change a cell all the cells depending on it will get run again