Dhirubhai Ambani Institute of Information and Communication Technology

JULIA PROGRAMMING LANGUAGE

Presented by: Bhavan Bhatt (202311021)
Pratham Patel (202311022)
Samarth Motka (202311023)
Harshneel Soni (202311024)
Aditya Patel (202311025)

Introduction to Julia Programming Language

- The Julia programming language was created to meet the demands for both high-level convenience and computational efficiency.
- It is a high-level, high-performance dynamic programming language for technical computing.
- It was developed to address the "two-language problem", which saw scientists and engineers frequently using a high-level language for performance and a low-level language for ease of creation.
- The goal of Julia is to offer a single language that combines the best aspects of each.
- Julia supports concurrent, parallel and distributed computing.
- Julia uses a just-in-time compiler that can generate highly optimized machine code, which helps achieve performance comparable to statically typed languages like C and Fortran.

History Of Julia Programming Language

- Julia's development started in 2009, and it was publicly released in 2012.
- Jeff Bezanson, Stefan Karpinski, Viral B. Shah and Alan Edelman created the language.
- Version 0.1 of Julia, the initial release for the general public, was made available in February, 2012.
- Following its original release, Julia attracted interest and a huge user and contributor community. The language underwent quick development and improvement, with frequent updates.
- In August 2018, Julia 1.0, an important milestone signifying language stability, was released. With this version, Julia enabled backward compatibility and demonstrated that it was prepared for widespread use in real-world settings.
- Currently Julia has V1.9.2 stable release.

Syntax : Variables

Example Variables

A variable
x=10

float variable
f=3.14

string variable
s="Hello World"

UTF-8 variable
σ = 1

Built-in Types

Int8,Int16,Int32,Int64,Int12, UInt32,UInt64, UInt128	103
Bool	false,true
Char	'Z'
Float16,Float32,Float64,	
complex	1 +2im
rational	5//6

Some math functions

round(Int, 76.2)

1+2
2+3/2*2

floor,ceil,sqrt
sin,cos,tan,
log,.....

Syntax : Strings

Types

A variable
x=10

float variable
f=3.14

string variable
s="Hello World"

UTF-8 variable
σ = 1

Simple Usage Examples

Using a char
a='A'

Simple String
b="For technical"

Regular Expression
match(r"(\W\w+){,2}",b)

Unicode usage
println("\u2200 x\u2203 y")

Triple Quote
```
json="""{
    "Id":10232
}"""
```
Concat
"Lets"*"code"

Repeat sentence
repeat("Julia",10)

Syntax : Functions, Control Flow

Functions

Basic function defination

```
function(x,y)
    x+y
end
```

terse function defination

```
f(x,y) = x+y
```

optional and keywords args
x:optional; a:keyword

```
f(x,y=1,a=3)=1
```

Control Flow

Compund expressions

```
z=begin(x,y)
   x+1
   x+y
end
```

Repeated eval loops

```
while x<1
for x = 1:10
```

Short-circuit evalutions
&&, ||

Conditional Evaluations

```
if x<10
  x +=1
elseif 10<=x <12
  x+=2
else
  x+=3
end
```

Exception Handling
try-catch

Tasks(coroutines)
yieldto

Major Applications

Data Science and Machine Learning

Julia has gained traction in the data science and machine learning communities due to its performance advantages. Libraries like "Flux.jl" offer efficient deep learning capabilities, and other data science libraries are catching up.

Metaprogramming support

Julia programs can generate other Julia programs, and even modify their own code, in a way that is reminiscent of languages like Lisp.

Static and Dynamic Type

Julia is both a dynamically- typed and a static-type language.

Major Applications

Parallel Computing

With built-in components for parallel programming at every level, Julia was created with parallelism in mind.Julia provides in-built multi-threading to enable multiple tasks to be executed in parallel within a single process or program, multi-processing, and distributed computing to enable computers to take on larger tasks.

Scientific Computing

Julia has a robust ecosystem of libraries focused on scientific computing and an effective inbuilt package manager to install and manage dependencies.

Robotics

Julia is suitable for robotics applications since it enables speedy algorithm testing and development.For instance, the JuliaRobotics GitHub Organization offers a selection of tools for Julia robotics development.

Major Applications

Web Development

Julia is not left out of the web development space as it has support to create web applications and APIs. The Genie.jl framework is a notable example that includes all you need to build full-stack applications and APIs (Genie).

Medicine and Pharmacy

Julia is widely used in the fields of medicine and pharmacy research. Researchers use Julia to analyze large datasets to determine the effectiveness of drugs, understand the long-term effects of treatments, and simulate and develop new treatments.

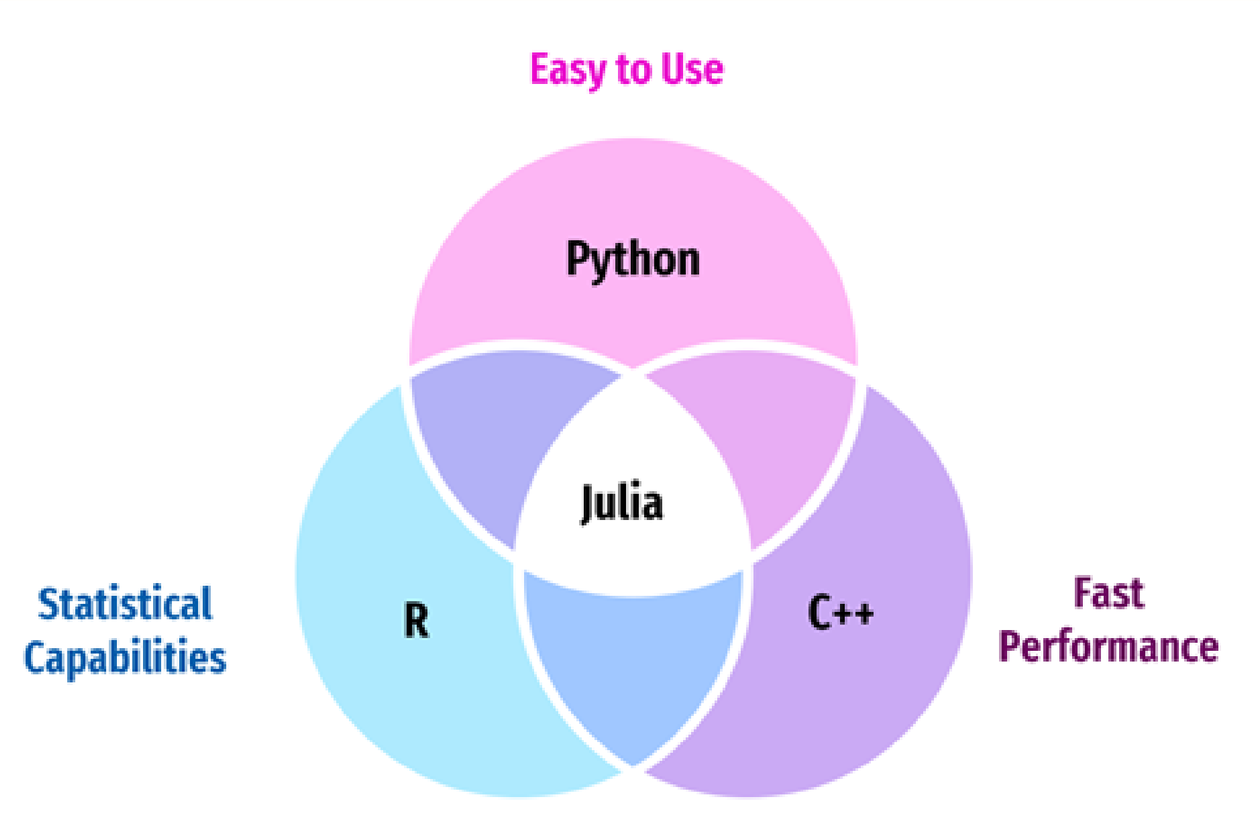
Audio Development

You can do audio processing, recording, development, manipulating, controlling, and other related tasks in Julia. Some of the packages that support these abilities are WAV.jl, PortAudio.jl, and MIDI.jl.

Features of Julia : Ease of Use

Ease of Use

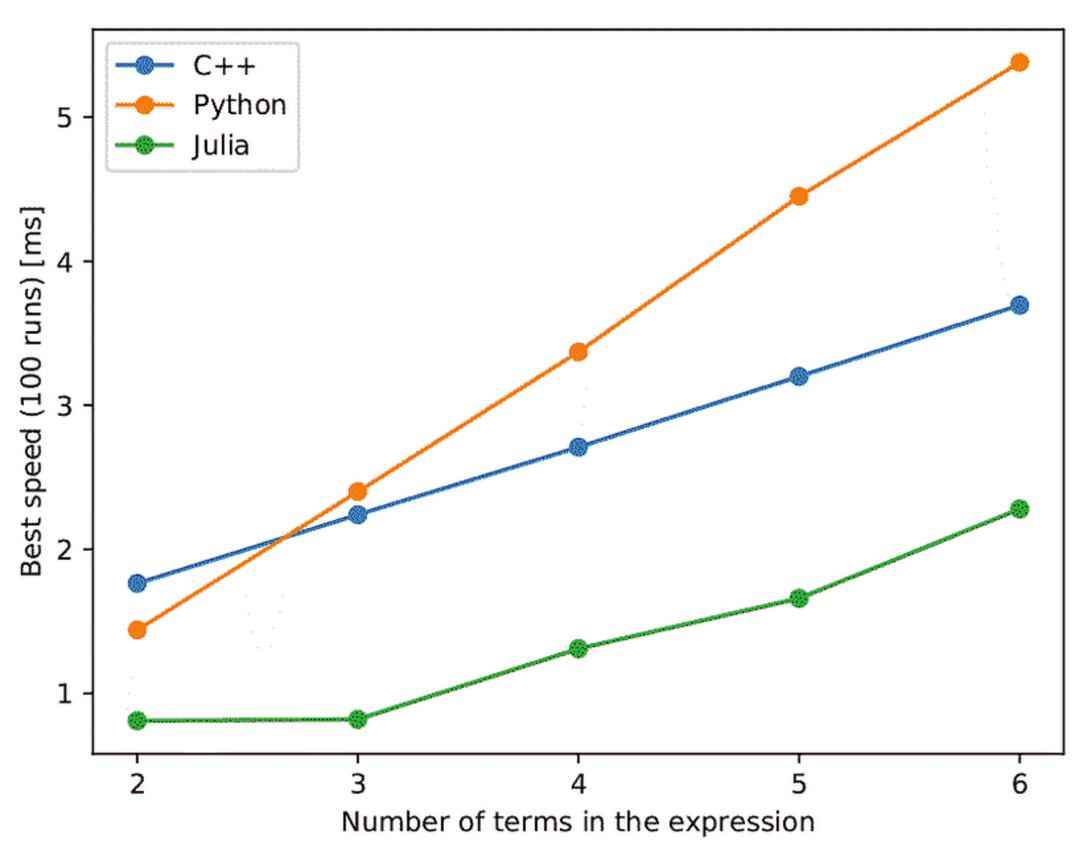
- User-friendly syntax
- Syntax similar to that of other popular programming languages like Python, MATLAB, and R.
- Easier to transition to Julia



Features of Julia : Performance

Performance

- Julia is known for its exceptional performance.
- Just-in-time (JIT) compiler
- Julia undoubtedly well-suited for tasks involving heavy computations, such as numerical simulations, large-scale data processing, and machine learning.



Special Features

Rich Ecosystem

- Despite being relatively young compared to some other programming languages, Julia has a growing ecosystem of packages and libraries.

- The Julia community is active and continuously developing new tools and resources.

Interoperability

- Julia is interoperable with other languages, meaning that you can include any other programming language such as Python, R, C, or C++ in your Julia program.

- You can easily call functions from C, Fortran, and Python libraries without much overhead, making it seamless to integrate existing codebases.

Special Features

Performance : Julia's outstanding performance is one of its most notable qualities. With its just-in-time (JIT) compiler, Julia is made to efficiently translate high-level code into optimized machine code.

Multiple Dispatch : Based on the types of each function input, Julia's multiple dispatch system allows for dynamic function overloading. As a result, writing generic and reusable functions is made simpler and extremely expressive and extensible code is enabled.

Summary

- The journey of Julia, from its inception at MIT to its current state, underscores its evolution as a powerful solution for high-performance computing needs.

- With a broad spectrum of applications spanning scientific research, data science, finance, engineering, and machine learning, Julia's versatility cannot be understated.

- The language's distinctive attributes, including exceptional performance, multi-dispatch capabilities, and seamless interoperability, position it at the forefront of modern programming languages, promising a bright and impactful future across various domains.

References

- https://juliadatascience.io/
- https://www.freecodecamp.org/news/applications-of-julia/
- https://docs.julialang.org/en/v1/
- http://web.mit.edu/julia_v0.6.2/julia/share/doc/julia/html/en/manual/documentation.html

Thank You

Feel free to ask questions!