* **Nithya Thimmaraju**

UG student, Department of CSE, Anurag group of institutions

* **Pattabhi Rama Chandra Murthy Malladi**

UG student, Department of CSE, Anurag group of institutions.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**“Innovation distinguishes between a leader and a follower.”**

**-Steve Jobs**

**Introduction:**

A fresh approach to Parallel Computing. Julia is a high-level and dynamic programming language. Julia was started in 2009, by Jeff Bezanson, Stefan Karpinski, Viral B. Shah, and Alan Edelman, their vision of creating a free language that was both high level and fast, is a start of something extraordinary.

Julia combines the functionality of quantitative environments such as R and Python with

the speed of production programming languages like Java and C++ to solve big data and analytics problems.

Julia provides parallel and distributed computing capabilities out of the box,

and literally infinite scalability with minimal effort. It delivers dramatic improvements

in simplicity, speed, capacity, and productivity for data scientists, algorithmic traders,

quants, scientists, and engineers who need to solve massive computation problems quickly and accurately*.* Julia offers an unbeatable combination of simplicity plus speed that is thousands of times faster than other mathematical, scientific and statistical computing languages. It provides a sophisticated compiler, distributed parallel execution, numerical accuracy, and an extensive library of fast mathematical functions.

Although designed for numerical computing, Julia is also useful for low-level systems programming, as a specific language, for web programming at both server and client side.

The main feature of Julia is:

**Multiple Dispatch**:

Julia draws significant inspiration from various dialects (Languages such as) of Lisp, including Scheme, Common Lisp, Dylan and Fortress (All these are Programming languages). Each and every programming language include a static and dynamic type system.

**

**Command Shell:**

* The Julia official distribution includes a "full-featured interactive command-line REPL(Compiler)" (read–eval–print loop), with a searchable history, tab-completion, many helpful key bindings, and dedicated help and shell modes which can be used to experiment and test code quickly. The following fragment represents a sample session example where Strings are concatenated automatically by println.

**Julia**> a(x) = 2x^3+1; f(x , y) = 1 + a(x)y

**Julia**> println(“Hello World!”, “ I’m “, f(2,1), “Welcome to Julia!”)

Hello World! I’m 18 Welcome to Julia!

* The REPL gives user access to the system shell and to help mode, by pressing ; or ? after the prompt (preceding each command), respectively.
* Code that can be tested inside the Julia's interactive section or saved into a file with a.jl extension and run from the command line by typing:

$ **Julia** <filename>

**Julia Environment:**

* Julia is supported by Jupyter, an online interactive “notebooks” environment. Julia is in practice interoperable with many languages. Julia's **ccall** keyword is used to call C-exported or Fortran shared library functions individually.

**Universal Transformation Format:**

* Julia has support for the current Unicode, with UTF-8 (Universal Transformation Format) used for strings, which means it ultimately provides common math operations or symbols, such as “**∈**” for the in operator. There are inbuilt Unicode format for several language characters such as Hindi.
* Julia supports markup languages such as HTML, XML, JSON and BSON, also for databases and web use in general.

**Applications:**

* Julia is a pretty programming language, from guiding self-driving vehicles to analysing images from deep space, Data hackers get giddy while talking about potential of Julia to oust R and python.
* The most notable aspect of Julia's implementation is its speed, which is often within a fully optimized C code (and thus often an order of magnitude faster than Python or R).

**Julia for Machine Learning:**

* Flux is a machine learning package for Julia with amazing potential.
* Flux follows an entirely different philosophy which I believe will make machine learning much easier to learn, understand and use even if you are a novice.
* It also integrates very easily with third party libraries, e.g. plug in the ability to run your neural network on a GPU.
* Hence one of the Julia’s strengths for technical computing is it’s meta programming features, which allows users to write code with minimal repetition. One such feature is **Generated functions**, a feature recently implanted in Julia that allows users to write customized kernels at “compile time”.

**Savitzky and Golay:**

* Generated functions can be used to construct a collection of filters to clean up data. One such filter was developed by [Savitzky and Golay](http://pubs.acs.org/doi/abs/10.1021/ac60214a047) in the context of cleaning up spectroscopic signals in analytical chemistry. The filtering method invented by Savitzky and Golay relies on least squares polynomial interpolation (of degree N) within a local moving window (of size 2M+1). An important property which makes the Savitzky-Golay method so incredibly useful in practice is that it [preserves the low moments of the data](https://inst.eecs.berkeley.edu/~ee123/sp15/docs/SGFilter.pdf), and thus the smoothening process preserves essential features of the peak structure in the data.

**Conclusion:**

Hence Julia contains wide range of packages and functions which gives faster and efficient result and build highly complex models which can be used in research and development of the world’s technology.

**References:**

* <https://medium.com/@acidflask/smoothing-data-with-julia-s-generated-functions-c80e240e05f3#.615wk3dle>
* <https://medium.com/@Jernfrost/machine-learning-for-dummies-in-julia-6cd4d2e71a46>
* <https://en.wikipedia.org/wiki/Julia_(programming_language)>
* <https://juliacomputing.com/>

****

**Nithya Thimmaraju** is currently pursuing Second year of B-tech in Computer Science at Anurag Group of Institutions and her interests includes Julia Computing, Machine Learning, Data Analytics, Neural Networks. She is passionate about Science and Space.

**Pattabhi Rama Chandra Murthy Malladi**  is currently pursuing his under graduation in Computer Science at Anurag Group of Institutions. His interests include data Analytics, Machine Learning and Block-chain. He is passionate about cryptocurrencies, digital currency and loves working on huge chunks of data.

**About the Authors**