Teoría de Lenguaje

Oscar Uriel Alvarado Garnica

Equipo: MexArg

MATRICES / VECTORES

- Julia proporciona una sintaxis intuitiva y eficiente para trabajar con matrices y vectores.

```
matriz = [7 8 9;
          6 5 4]
```

```
vector = [1, 2, 3]
```

```
matriz * vector
```

ANÁLISIS DE DATOS Y ESTADÍSTICA

- En general, Julia es una excelente opción para el análisis de datos y la estadística debido a su rendimiento y flexibilidad.
- Puedes combinar la facilidad de uso de paquetes como DataFrames con el poder de Julia para análisis más avanzados.
- Además, Julia se integra bien con otros paquetes populares y es compatible con las mejores prácticas de análisis de datos.

PLOTS

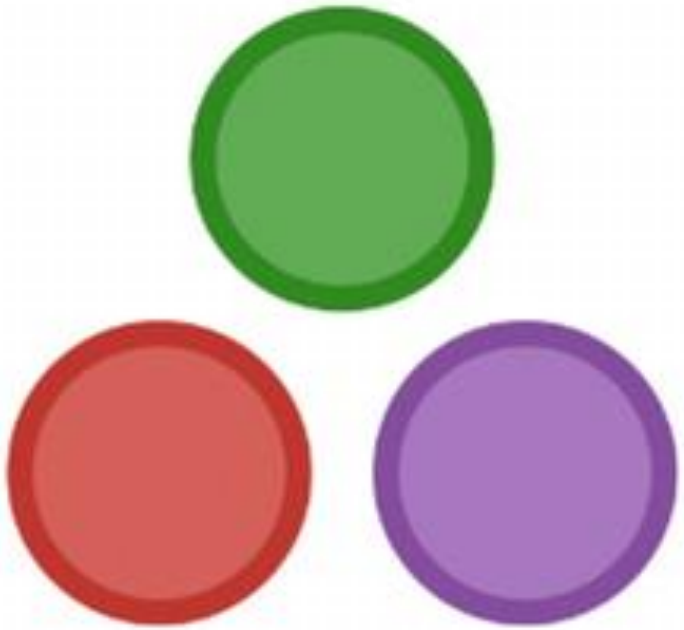
1. **Facilidad de Uso:** La sintaxis de Plots es sencilla y fácil de aprender. Puedes crear visualizaciones de datos con pocas líneas de código.
2. **Versatilidad:** Plots soporta una amplia variedad de gráficos, desde simples gráficos de líneas hasta gráficos 3D complejos. También es compatible con múltiples formatos de salida, como PNG, SVG, PDF, etc.
3. **Interactividad:** Algunos backends de Plots, como PlotlyJS, ofrecen funcionalidades interactivas que te permiten explorar y manipular gráficos en tiempo real.
4. **Documentación Completa:** Plots tiene una documentación exhaustiva y una comunidad activa. Puedes encontrar recursos en línea y recibir ayuda de la comunidad Julia.

Funciones anónimas

- Son funciones que no tienen un nombre explícito.
- Se definen utilizando la palabra clave -> (flecha)
- Son especialmente útiles cuando se necesita una función simple para realizar una tarea específica.

MACROS

1. **Generación de Código Dinámico:** Las macros permiten generar y manipular código de manera dinámica durante el tiempo de compilación.
2. **Eficiencia:** Dado que las macros trabajan durante el tiempo de compilación, el código generado puede ser altamente eficiente y adaptarse específicamente a los tipos y contextos particulares.
3. **Abstracciones Más Poderosas:** Las macros permiten crear abstracciones más poderosas que las funciones regulares. Pueden introducir nueva sintaxis, simplificar la escritura de código y proporcionar interfaces más expresivas.
4. **Meta-programación:** Las macros permiten realizar meta-programación, lo que significa que el código puede escribirse para manipular y generar código automáticamente. Esto es especialmente útil en situaciones donde se necesitan construcciones de código complejas o repetitivas.

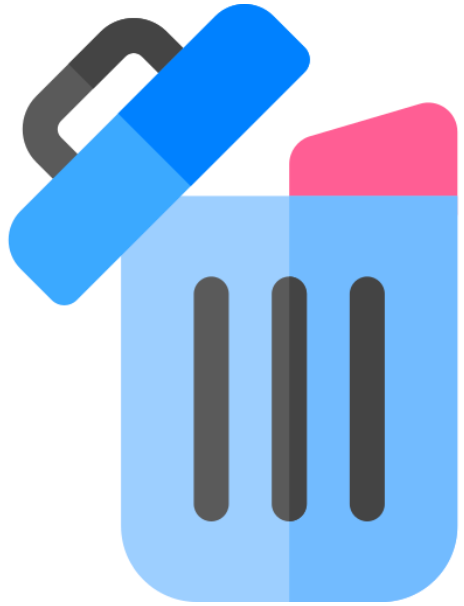


MULTIPLE DISPATCH

- Paradigma de programación que Julia adopta.
- Se refiere a la capacidad de seleccionar automáticamente qué método o función específica debe ejecutarse según los tipos de todos sus argumentos.
- Este enfoque permite escribir código genérico que puede manejar diferentes tipos de manera eficiente y elegante.

¿ELIMINAR MÉTODOS ?

- Es posible eliminar métodos de funciones existentes, aunque es una característica que pocos desarrolladores conocen y que no se utiliza comúnmente.
- Esta capacidad de eliminar métodos puede tener consecuencias impredecibles y no se recomienda a menos que se comprenda completamente el impacto de dicha acción.
- En la mayoría de los casos, es preferible extender o especializar métodos existentes en lugar de eliminarlos por completo.



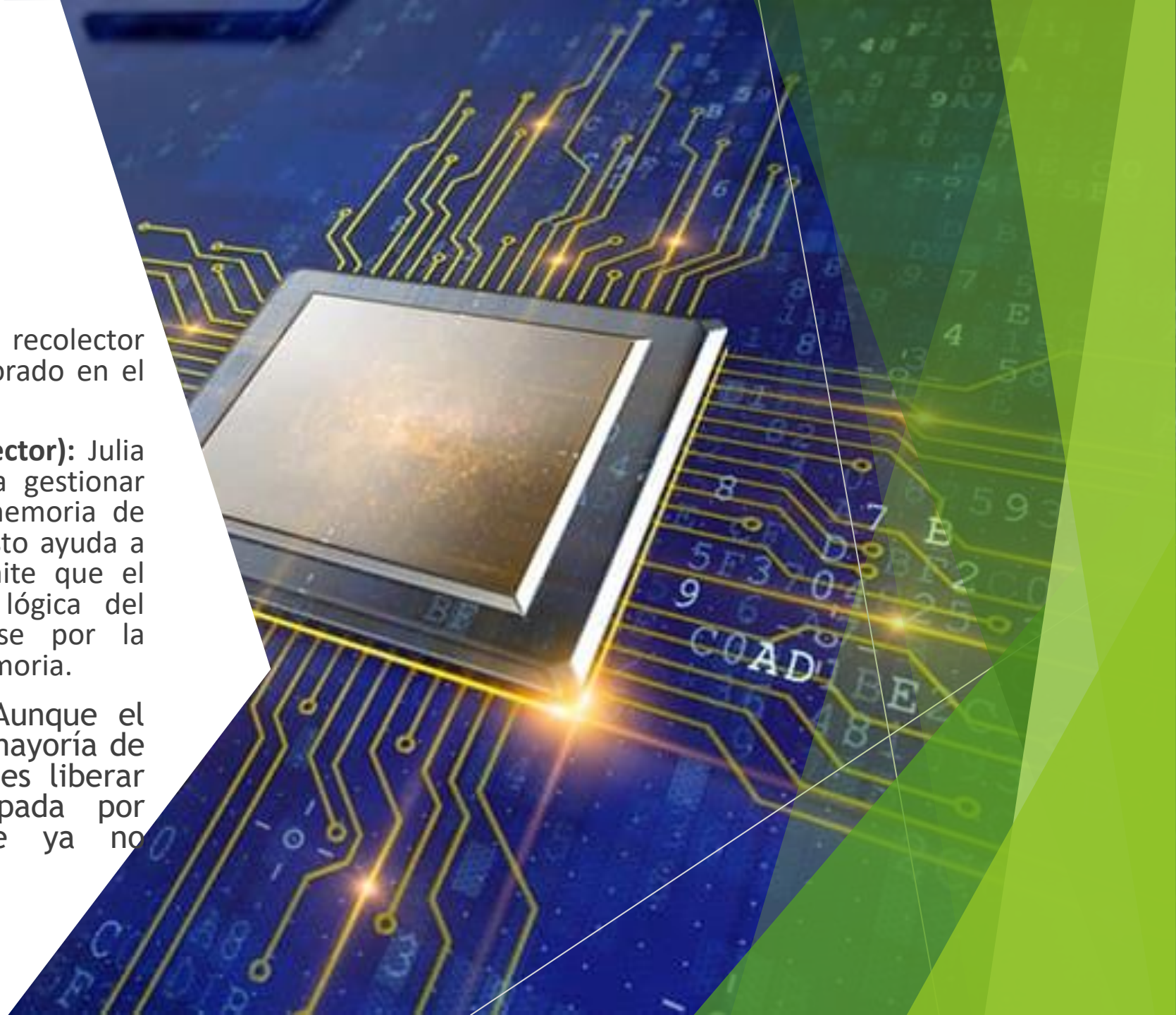
```
import Base: println
methods(Base.println)

Base.delete_method(methods(println)[1])
```

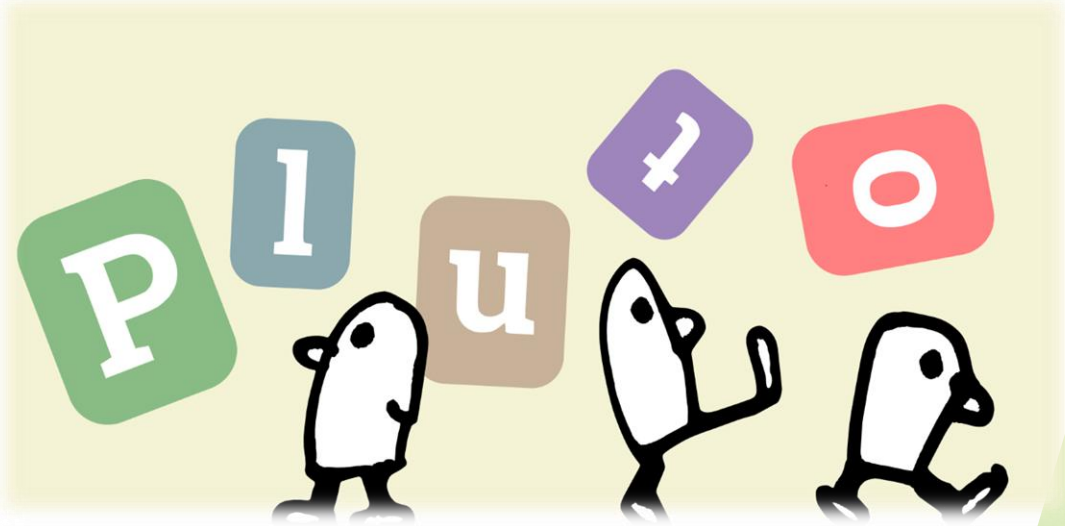
DISTRIBUIDOS

- Capacidad de ejecutar programas en un entorno distribuido, es decir, en múltiples procesadores o nodos de una red.
- La computación distribuida en Julia es compatible con el procesamiento paralelo y permite ejecutar operaciones simultáneas en diferentes partes de los datos.
- Julia proporciona un módulo llamado **Distributed** que incluye funciones y herramientas para la computación distribuida.

MANEJO DE MEMORIA

- Gestionado automáticamente por el recolector de basura (garbage collector) incorporado en el lenguaje.
- **Recolector de Basura (Garbage Collector):** Julia utiliza un recolector de basura para gestionar automáticamente la liberación de memoria de objetos que ya no son necesarios. Esto ayuda a prevenir fugas de memoria y permite que el desarrollador se concentre en la lógica del programa en lugar de preocuparse por la asignación y liberación manual de memoria.
- **Limpiar Objetos Innecesarios:** Aunque el recolector de basura gestiona la mayoría de los objetos, es posible que desees liberar manualmente la memoria ocupada por objetos grandes o datos que ya no necesitas.





PLUTO

- Pluto.jl es un entorno interactivo y una interfaz de programación para el lenguaje de programación Julia.
- Fue desarrollado con el objetivo de proporcionar una experiencia de programación más interactiva y exploratoria.
- A diferencia de los notebooks tradicionales, Pluto.jl se centra en la interactividad y la capacidad de modificar y experimentar con el código de manera dinámica.

Bibliografía

- * https://docs.julialang.org/en/v1/
- * https://help.juliahub.com/juliahub/stable/
- * https://introajulia.org/#_el_primer_programa
- * https://chifi.dev/weird-things-you-can-do-in-julia-3f10cacb8ef4