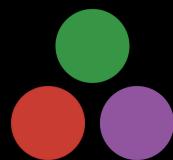




# JULIA

UN LENGUAJE MODERNO PARA LA COMPUTACIÓN  
CIENTÍFICA Y TÉCNICA DE ALTO RENDIMIENTO

---



# ¿Qué es Julia?

Julia es un **lenguaje de programación** de alto nivel, de código abierto, diseñado en 2012 por Jeff Bezanson, Stefan Karpinski, Viral B. Shah y Alan Edelman.

¿Su objetivo? Combinar la velocidad de lenguajes informáticos muy rápidos como C, con la facilidad de escritura y lectura de un lenguaje simple como **Python**.



```
Example.jl - Example.jl - Visual Studio Code
File Terminal Help
...
src > Example.jl
/workflows
ot.yml
jl
ct.toml
ple.jl
sts.jl
vor.yml
ov.yml
ore
E.md
:toml
1E.md
Example.jl X
src > Example.jl
1 module Example
2 export hello, domath
3 """
4 """
5     ...hello(who::String)
6
7     Return "Hello, `who`".
8 """
9 hello(who::String) = "Hello, $who"
10 """
11 """
12     ...domath(x::Number)
13
14     Return `x + 5`.
15 """
16 domath(x::Number) = x + 5
17
18 end
19
```

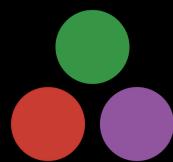


# ¿Por qué aprender Julia?

## ¿SIMPLICIDAD Y RENDIMIENTO NATIVO?

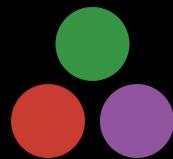
- *Rendimiento cercano al lenguaje de máquina..*
- *Una sintaxis accesible para científicos y desarrolladores*
- *Interoperabilidad sin fricción*
- *Paralelismo y cálculo distribuido integrados*
- *Un ecosistema en plena expansión*





# BASES DE JULIA

| Concepto básico                 | Lo que necesitas saber                                                                                                                                      |
|---------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Declaración de variables</b> | Julia detecta automáticamente el tipo de cada variable.                                                                                                     |
| <b>Estructuras de control</b>   | Las instrucciones if, for, while se escriben como en Python, con una sintaxis clara e intuitiva.                                                            |
| <b>Definición de funciones</b>  | Sintaxis simple: function cuadrado(x) return x^2end                                                                                                         |
| <b>Tipos de colecciones</b>     | Julia ofrece varias estructuras: Arreglos: [1, 2, 3] Diccionarios: Dict(«a» => 1) Tuplas: (1, «a»)                                                          |
| <b>Funcionalidad clave</b>      | El <i>multiple dispatch</i> permite ejecutar diferentes versiones de una función según el tipo de los argumentos, ofreciendo más flexibilidad y eficiencia. |



# Julia vs Python, R, Matlab

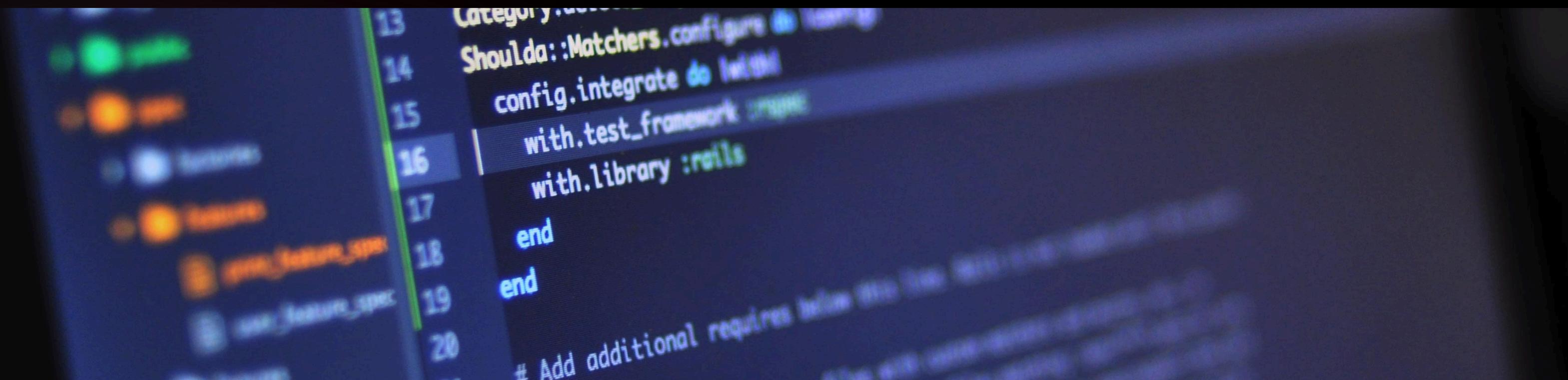
| Criterio     | Julia            | Python   | R             | Matlab     |
|--------------|------------------|----------|---------------|------------|
| Velocidad    | ▲ Excelente      | Buena    | Media         | Buena      |
| Sintaxis     | Simple           | Simple   | Especializada | Técnica    |
| Open Source  | Sí               | Sí       | Sí            | No         |
| Especialidad | Cálculo numérico | Versátil | Estadística   | Ingeniería |

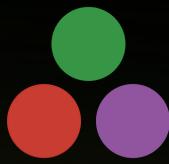


# VENTAJAS



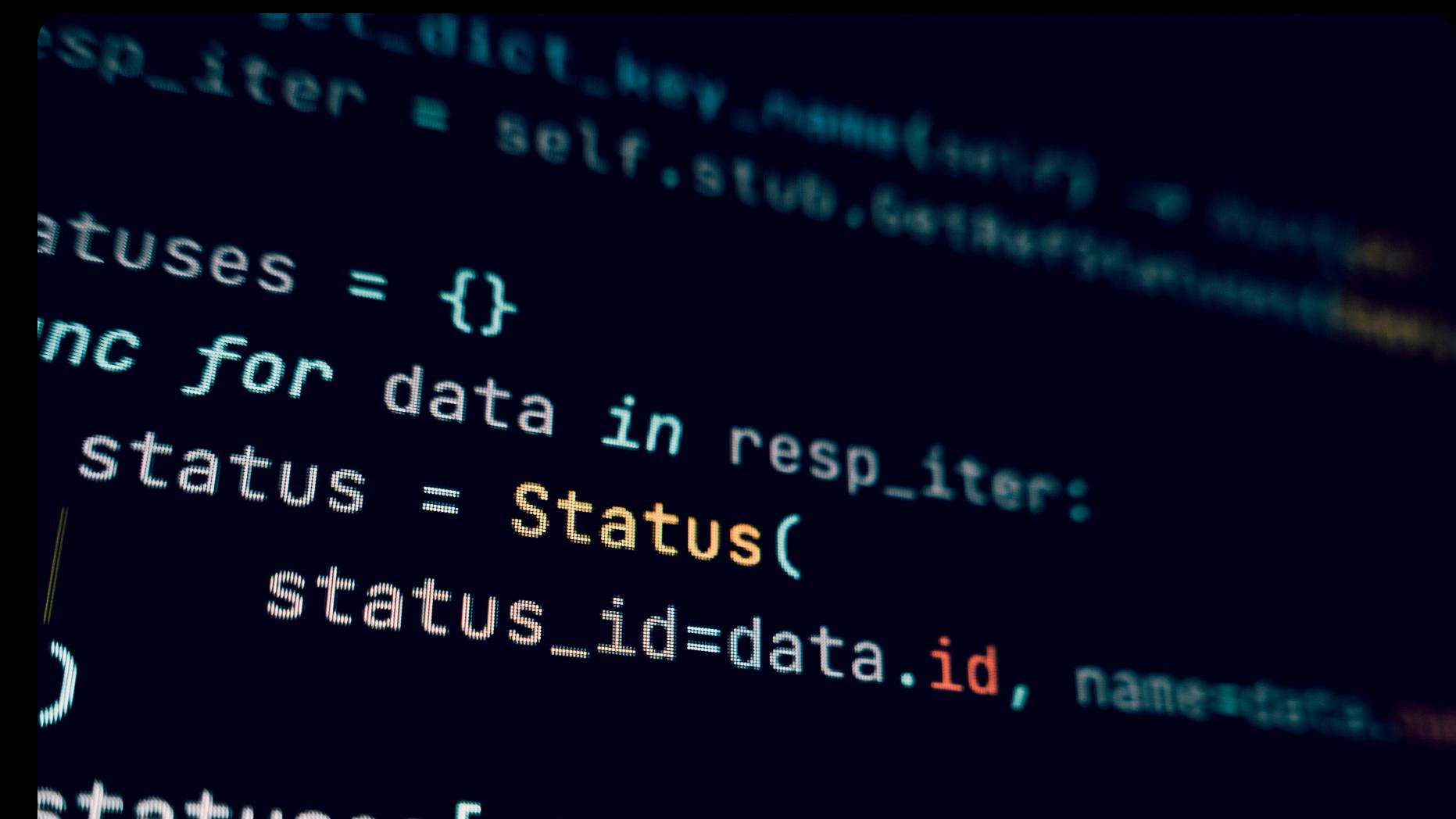
- *Más rápida que otros lenguajes de ciencia de datos.*
- *Julia tiene un amplio conjunto de estructuras de datos y funcionalidades integradas*
- *Está diseñado teniendo en cuenta la computación paralela y distribuida*
- *Julia tiene una sintaxis simple y expresiva*





# DESVENTAJAS

- *Al ser un lenguaje relativamente nuevo, su comunidad no es muy grande a comparación de otros lenguajes como R o Python.*
  - *Algunas bibliotecas y paquetes para ciencia de datos aún no están disponibles en Julia, por lo que es posible que deba usar otros lenguajes para ciertas tareas o usar el paquete PyCall para llamar a las bibliotecas de Python.*
  - *Algunos usuarios pueden encontrar que la sintaxis de Julia es menos familiar que la de Python, lo que puede dificultar el aprendizaje para algunas personas.*

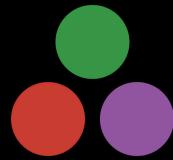




## RESUMEN

En general, Julia es un lenguaje poderoso para la ciencia de datos y ofrece muchos beneficios sobre otros lenguajes, pero puede que no sea la mejor opción para todos los casos de uso. Vale la pena evaluar las necesidades específicas de su proyecto y determinar si Julia es la opción correcta para sus tareas de ciencia de datos.

# CONCLUSIÓN



GRACIAS  
GRACIAS  
GRACIAS