

# Programación en Julia: Primeros pasos

CADI

Héctor Medel

Benjamín Pérez

Tecnológico de Monterrey

January 9, 2023

# Toma de asistencia

# Objetivo

Introducir, de manera práctica, las [bases de programación en Julia](#). A través de ejemplos se revisarán los elementos principales de este lenguaje de programación y paquetes más útiles.

# Fechas y horarios

- ▶ Sesiones sincrónicas: Del 9 al 13 de enero (Lunes a Viernes) de 09:00 a 13:00 hrs.
- ▶ Actividades asincrónicas: Del 9 al 13 de enero (Lunes a Viernes) de 14:00 a 16:00 hrs.
- ▶ Modalidad: Virtual.

# Políticas para acreditar el curso


- ▶ **Asistencia** de al menos el 80% del taller.
- ▶ A lo largo de la semana se encargarán aproximadamente **4 tareas**. Las tareas serán entregadas en equipos de 2 integrantes.
- ▶ La correcta solución de las tareas deberá ser entregada a más tardar el **viernes 13 de enero a las 23:59 hrs.**

# Temario del curso

1. Presentación
2. Instalación y editores
3. Paquetes
4. Variables y tipos
5. Flujo de control
6. Funciones
7. Estructuras de datos
8. Input/Output
9. Gráficas simples
10. Ejemplos de despacho múltiple


# Instalación de Julia

- ▶ Para descargar Julia ingresa al sitio <https://julialang.org/downloads/>
- ▶ Descarga instala el archivo correspondiente a tu sistema operativo.



[Download](#)[Documentation](#)[Learn](#)[Blog](#)[Community](#)[Research](#)[JSOC](#)[Sponsor](#)

## Download Julia

 41,229

Please star us on [GitHub](#). If you use Julia in your research, please [cite us](#). If possible, do consider [sponsoring us](#).

### Current stable release: v1.8.4 (December 23, 2022)

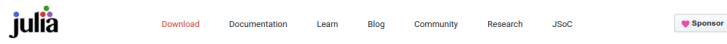
Checksums for this release are available in both [MD5](#) and [SHA256](#) formats.

<b>Windows</b> <a href="#">[help]</a>	64-bit (installer), 64-bit (portable)	32-bit (installer), 32-bit (portable)
<b>macOS x86 (Intel or Rosetta)</b> <a href="#">[help]</a>	64-bit (.dmg), 64-bit (.tar.gz)	
<b>macOS ARM (M-series Processor)</b> <a href="#">[help]</a>	64-bit (.dmg), 64-bit (.tar.gz)	
<b>Generic Linux on x86</b> <a href="#">[help]</a>	64-bit (glibc) (GPG), 64-bit (musl) <sup>[1]</sup> (GPG)	32-bit (GPG)
<b>Generic Linux on ARM</b> <a href="#">[help]</a>	64-bit (AArch64) (GPG)	
<b>Generic FreeBSD on x86</b> <a href="#">[help]</a>	64-bit (GPG)	
<b>Source</b>	Tarball (GPG)	Tarball with dependencies (GPG) <a href="#">GitHub</a>

Almost everyone should be downloading and using the latest stable release of Julia. Great care is taken not to break compatibility with older Julia versions, so older code should continue to work with the latest stable Julia release. You should only be using the long-term support (LTS) version of Julia if you work at an organization where implementing or certifying upgrades is prohibitively expensive and there is no need for new language features or packages. See this description of “[Risk Personas](#)” for more detail on who should be using what versions of Julia based on their risk tolerance. See this blog post on [Julia's Release Process](#) for more information on different kinds of releases.

# Instalación de Julia

- ▶ Por ahora, es recomendable dejar la configuración por defecto durante el proceso de instalación.
- ▶ Dependiendo de tu OS, sigue las instrucciones del instalador.



## Download Julia

Star 41,229

Please star us on [GitHub](#). If you use Julia in your research, please [cite us](#). If possible, do consider [sponsoring us](#).

### Current stable release: v1.8.4 (December 23, 2022)

Checksums for this release are available in both [MD5](#) and [SHA256](#) formats.

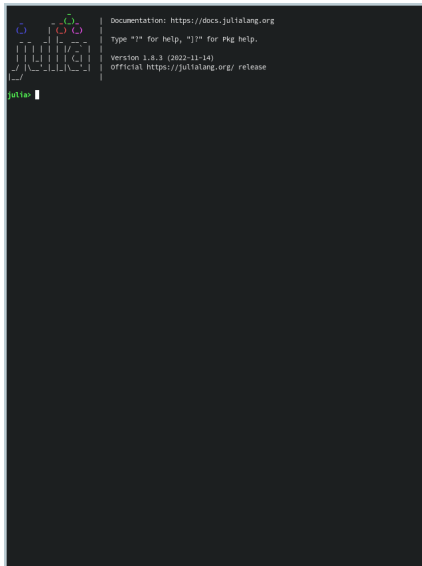
<b>Windows</b> <a href="#">[help]</a>	64-bit (installer), 64-bit (portable)	32-bit (installer), 32-bit (portable)	
<b>macOS x86 (Intel or Rosetta)</b> <a href="#">[help]</a>	64-bit (.dmg), 64-bit (.tar.gz)		
<b>macOS ARM (M-series Processor)</b> <a href="#">[help]</a>	64-bit (.dmg), 64-bit (.tar.gz)		
<b>Generic Linux on x86</b> <a href="#">[help]</a>	64-bit (glibc) (GPG), 64-bit (musl) <sup>[1]</sup> (GPG)	32-bit (GPG)	
<b>Generic Linux on ARM</b> <a href="#">[help]</a>	64-bit (AArch64) (GPG)		
<b>Generic FreeBSD on x86</b> <a href="#">[help]</a>	64-bit (GPG)		
<b>Source</b>	Tarball (GPG)	Tarball with dependencies (GPG)	<a href="#">GitHub</a>

Almost everyone should be downloading and using the latest stable release of Julia. Great care is taken not to break compatibility with older Julia versions, so older code should continue to work with the latest stable Julia release. You should only be using the long-term support (LTS) version of Julia if you work at an organization where implementing or certifying upgrades is prohibitively expensive and there is no need for new language features or packages. See this description of “[Risk Personas](#)” for more detail on who should be using what versions of Julia based on their risk tolerance. See this blog post on [Julia's Release Process](#) for more information on different kinds of releases.



# Trabajando en Julia – Real Evaluate Print Loop (REPL)

- Al correr el archivo ejecutable de Julia, se abrirá una ventana similar a la siguiente.

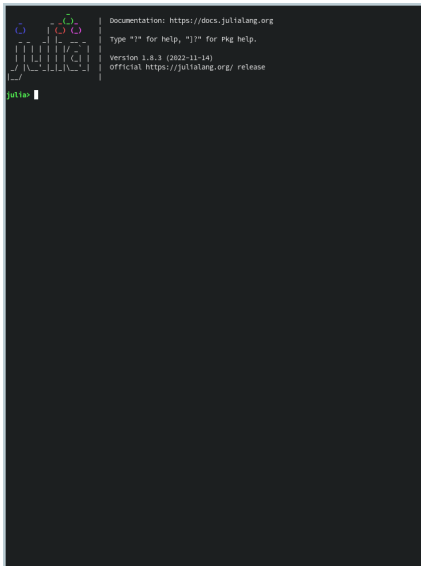
A screenshot of the Julia REPL (Real Evaluate Print Loop) window. The window has a dark background. In the top left corner, there is a small logo consisting of four colored circles (blue, green, red, yellow) arranged in a square. To the right of the logo, the text "Documentation: https://docs.julialang.org" is displayed. Below this, the text "Type '?' for help, ']' for pkg help." is shown. Further down, the text "Version 1.8.3 (2022-11-14)" and "official https://julialang.org/ release" are visible. At the bottom left, the prompt "julia>" is shown in green, followed by a white cursor. The rest of the window is empty.

```
Documentation: https://docs.julialang.org
Type '?' for help, ']' for pkg help.
Version 1.8.3 (2022-11-14)
official https://julialang.org/ release

julia> |
```

# Trabajando en Julia – Real Evaluate Print Loop (REPL)

- Sigamos en la terminal.



```
Documentation: https://docs.julialang.org
Type "?" for help, "?>" for pkg help.
Version 1.8.3 (2022-11-14)
official https://julialang.org/ release

julia> |
```

## Probemos los siguientes comandos

```
julia> 6 * 7  
42  
julia> ans  
42  
julia> ans + 10  
52
```

Si por alguna razón no queremos que se despliegue el resultado, agregamos ; al final.

Podemos **asignar** un valor a una variable

```
julia> a = 6 * 7  
42  
julia> b = "Hola"  
"Hola"
```

# Algunos comandos básicos en el REPL

- ▶ Flecha hacia arriba/abajo nos ayudan a navegar en el historial de comandos ejecutados.
- ▶ Borrar pantalla CTRL+L
- ▶ Interrumpir la ejecución de un comando CTRL+C

## Accesar a la [documentación/ayuda](#)

Cuando ingresamos el caracter ? en el REPL, notemos que cambia de la siguiente manera

```
help?>
```

Busquemos ayuda acerca de la función coseno.

## Accesar a la [documentación/ayuda](#)

Cuando ingresamos el caracter ? en el REPL, notemos que cambia de la siguiente manera

```
help?>
```

Busquemos ayuda acerca de la función coseno.

```
help?> cos
```

```
search: cos cosh cosd cosc cospi acos acosh acosd sincos sincosd sincospi
```

```
cos(x)
```

```
Compute cosine of x, where x is in radians.
```

```
See also [cosd], [cospi], [sincos], [cis].
```

# Podemos correr `scripts` en el REPL

Generemos un archivo llamado `miscript.jl`, e incluyamos lo siguiente

```
# Script que suma a y b  
a = 1  
b = 2  
suma = a+b
```

Posteriormente, dentro del REPL ejecutemos lo siguiente

```
julia> include("miscript.jl")  
3
```



# Manejo de paquetes (pkg)

Cuando ingresamos el caracter ] en el REPL, notemos que cambia de la siguiente manera

```
(@v1.8) pkg>
```

Esto es conocido como el modo pkg. Dentro de este entorno es como instalamos (y compartimos) librerías y paquetes.

## Instalemos un paquete para graficar

Ejecutemos las siguientes líneas dentro del modo pkg

```
(@v1.8) pkg> add Plots
```

## Instalemos un paquete para graficar

Ejecutemos las siguientes líneas dentro del modo pkg

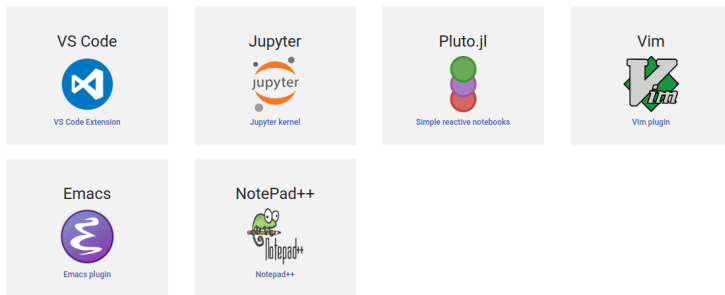
```
(@v1.8) pkg> add Plots
```

Posteriormente, salimos del modo Pkg, y graficaremos una función

```
julia> using Plots  
julia> x = rand(50)  
julia> plot(x)
```

# Por ahora hemos interactuado con Julia vía el REPL...

Existen diversos **IDEs y Editores**, por ejemplo



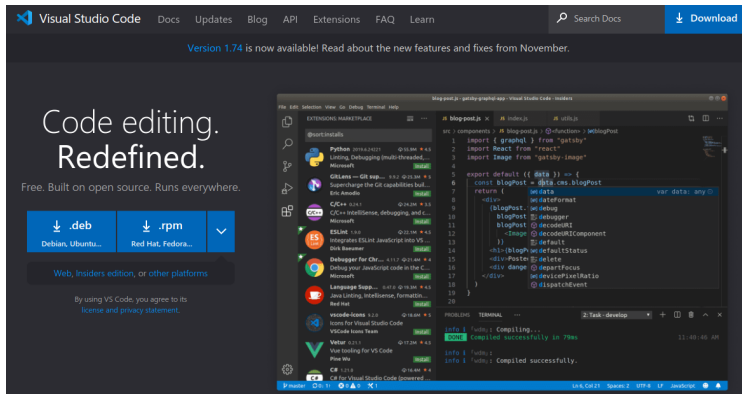
Veremos la instalación de VSCode y Pluto.jl

# ¿Qué es VSCode?

- ▶ Editor de código multiplataforma.
- ▶ Soporta varios lenguajes, entre ellos Julia.

# Instalación de VSCode

- ▶ Para descargar VSCode ingresa al sitio <https://code.visualstudio.com/>
- ▶ Descarga instala el archivo correspondiente a tu sistema operativo.



# Extensión de Julia

- ▶ Dentro de VSCode, instalaremos la extensión para Julia.
- ▶ Abre el menú de extensiones que se encuentra en la barra vertical de la izquierda.
- ▶ En el cuadro de búsqueda escribe `julia`, e instala la extensión.

# ¿Qué es Pluto.jl?

- ▶ Entorno de programación para Julia tipo [notebook](#).
- ▶ Código [interactivo/reactivo](#).
- ▶ Más acerca de esta herramienta en el CADI “Programación en Julia: Herramienta para la enseñanza”.



# Instalemos Pluto.jl

Dentro de la terminal instalaremos el paquete como lo hicimos para el caso del paquete Plots. Es decir

```
(@v1.8) pkg> add Pluto
```

# Instalemos Pluto.jl

Dentro de la terminal instalaremos el paquete como lo hicimos para el caso del paquete Plots. Es decir

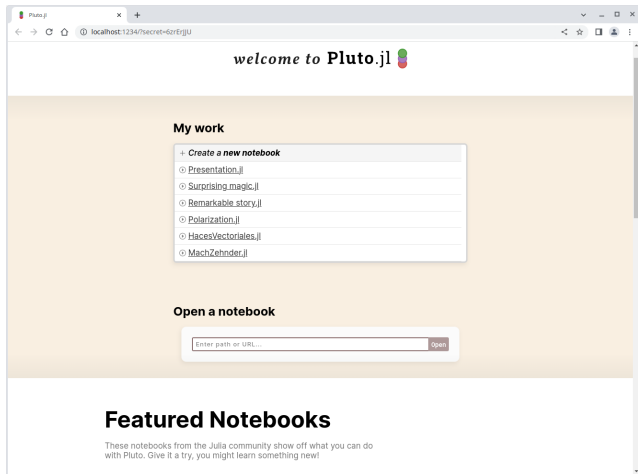
```
(@v1.8) pkg> add Pluto
```

Posteriormente, salimos del entorno Pkg, cargamos Pluto y lo ejecutamos.

```
julia> using Pluto  
julia> Pluto.run()
```

# Instalemos Pluto.jl

Lo anterior abrirá una ventana de nuestro navegador



# Instalemos Pluto.jl

Ahora vamos a trabajar en un nuevo documento... +Create a new notebook

