

# Workshop for Economic Modeling in Julia

Y.-H. Henry Chen  
Research Scientist, MIT CS3

Jan 6, 2026

# Arithmetical operations

- Turn on VS code
  - Terminal -> New Terminal
  - Ctrl + shift + p -> Julia: Start REPL (Read Evaluate Print Loop)
- Order of operations
  - 1<sup>st</sup> things inside the parenthesis ()
  - 2<sup>nd</sup> exponentiation ^
  - 3<sup>rd</sup> multiplication and division, from left to right \*, /
  - 4<sup>th</sup> addition and subtraction, from left to right +, -

# Logical expressions

- Logical operators

- ! NOT
- && AND
- || OR

- Comparison operators

- < less than
- == equal to
- > greater than
- <= less than or equal to
- >= greater than or equal to

} create logical expressions, e.g.,

true&&true	->	true
true&&false	->	?
true    false	->	?
!true	->	?

-2 > 1	->	false
3 <= 0	->	?
-2 == -2	->	?

# Common data types

- Scalar types

• Int64	Standard 64-bit integer	26
• Float64	Standard 64-bit floating point	2.08
• Bool	Boolean values	true / false
• String	UTF-8 encoded text	"Hey Julia"
• Char	A single character	'B'
• Symbol	Colon + String or Char used as an identifier	:label_a

# Common data types

- Scalar types
  - In VS code, under **julia >**, make assignments, such as `a = 3`, `b = "Hi"`, etc.
  - To see the type of `a` and `b`, use **typeof()**, e.g., `typeof(a)`, `typeof(b)`, etc.
  - Do `2` and `2.0` have the same data type?
  - For a variable named `D`, is it the same as `d`?

# Common data types

- Collection types

- Arrays (Vector, Matrix)
- Tuples
- Dictionaries
- NamedTuples

One can access an element by its position



Ordered collections of a specific type [0, -1, 3]

Ordered, fixed-size, immutable (2, 3, -8)

Key-value pairs d = Dict("cat" => 3, "dog" => 2)

~ tuples, each element has a name nt = (name= "John", age=25)

# Common data types

- Collection types

- Using VS code, let's make some assignments like `x1, x2, y1, y2, z, w` below:
  - `x1 = [0, -1, 3] ; x2 = [1, 2, 5]`                           `x1[1] = ? x1+x2 = ? X1[1:2] = ?`
  - `y1 = (2, 3, -8) ; y2 = (-3, 0, -1)`               `y1[2] = ? Can we do y1+y2 = ?`
  - `z = Dict("cat" => 3, "dog" => 2)`               `z["dog"] = ?`
  - `w = (name="John", age=25)`                       `w[1] = ?`

# Common data types

- Collection types

- Using VS code, suppose we want to:
- Add an element 7 to  $x1 = [0, -1, 3]$ , we can do **push!(x1, 7)**. Now  $x1$  is  $[0, -1, 3, 7]$
- Remove the 1<sup>st</sup> and 3<sup>rd</sup> element of  $x1$ , which is  $[0, -1, 3, 7]$ , we can do: **deleteat!(x1, [1, 3])**
- $z = \text{Dict}(\text{"cat"} \Rightarrow 3, \text{"dog"} \Rightarrow 2)$   
Sum over “cat” and “dog” to get the total: **sum(z[k] for k in ["cat", "dog"])**

# Check data types

- `typeof()`
  - Check the data type of an object
  - `z = Dict("cat" => 3, "dog" => 2)`, e.g., `typeof(z) = ?`
  - `A = [0, -1, 3]`
  - `b = (2, 3, -8)`
  - `r = "Good morning"`

# Check methods of a function

- `methods()`
  - Present all implementations of a function
  - For instance, `+` is a function, when you type `methods(+)`, you will see something like this:

```
# 223 methods for generic function "+" from Base:
```

...

```
[62] +(a::Integer, b::Integer)
[161] +(c::Union{Int16, Int32, Int8}, x::BigFloat)
[8] +(x::Bool, y::Bool)
[48] +(A::Array, Bs::Array...)
[52] +(z::Complex, x::Real)
```

...

# REPL modes

- **julia>**
  - Julian mode (default)
  - Run Julia code
- **pkg>**
  - Package manager
  - Manage packages and environments
  - In julia>, press **]** to enter, press **Backspace** to go back to >Julia
- **shell>**
  - To execute the command prompt without leaving Julia
  - cmd /c dir; cmd /c mkdir test; cmd /c rmdir test ...
  - Press **;** to enter, press **Backspace** to go back to >Julia
- **help?>**
  - View documentations for types or functions
  - In julia>, press **?**, and then enter the types or functions you would like to check

# Using VS code

- Open a new file and save it
  - Step 1: **File => New File => Julia File**
  - Step 2: **File => Save As => Enter file name**

# Using VS code

- Edits
  - Undo            Ctrl + z
  - Redo            Ctrl + y
  - Find            Ctrl + f
  - Replace        Ctrl + h
  - Copy            Ctrl + c
  - Paste           Ctrl + v

# Using VS code

# Using VS code

- Compare two files
  - Step 1: **File => Open Folder** => select the folder / working directory
  - Step 2: In Explorer on the left, right click the 1<sup>st</sup> file => **Select for Compare**
  - Step 3: In Explorer on the left, right click the 2<sup>nd</sup> file => **Compare with Selected**

# Using VS code

- Split the window
  - `Ctrl + \` -> split the window vertically
  - `Ctrl + \` and then `Ctrl + k` followed by  -> split the window horizontally
  - `Ctrl + w` -> eliminate the split window

# Bibliography

- Bezanson, J., Edelman, A., Karpinski, S., & Shah, V. B. (2017). Julia: A fresh approach to numerical computing. *SIAM Review*, 59(1), 65-98. <https://doi.org/10.1137/141000671>
- Klopper, J. and H. Laurie (2025). Julia Scientific Programming. University of Cape Town.