Macroeconomics

Week 1

Ricardo Gouveia-Mendes

ricardo.mendes@iscte-iul.pt

Undergraduate in Economics

1st Semester 2023-24

Welcome to julia and Pluto.jl

Why teaching Economics and julia?

- The **nature** of Economic Science
  - Human Science — *which object?*
  - Data
  - Mathematics is to Economics as Cartography to Geography
- The **Julia** programming language
  - General purpose programming language born in 2015 at MIT
  - The high performance promise: *Walks like Python runs like C*

What about Pluto.jl ?

- Pluto.jl is a **Julia package**

- It provides **Notebooks** as *web-based IDEs* for Julia
  - Plain *Julia files*: `*.jl`
  - Structured in *cells* that allow multiple types of contents
    - Chunks of Julia code to run calculations
    - Text to be formatted
  - *Reactive:* all the code is updated when something changes
  - *Interactive tools:* ideal for learning

First things first

1. Zipped files
2. Opening Julia
   1. Standard mode and `Pkg` mode
   2. Installing packages: `add Package`
   3. Updating packages: `] up`
3. Running Pluto.jl: `import Pluto; Pluto.run()` in standard mode
4. Open a Notebook
   1. Static and Dynamic versions
   2. Checking the loading progress
   3. Checking the Notebook location in your PC
5. Save a Notebook: `Ctrl + S`

Working with julia and Pluto.jl

Cells with text: basic formatting

- The simplest solution is to use **Markdown** blocks

```
md"This is a Markdown single line input text."
```

```
md"""
    This is a Markdown multiple line input text.
"""
```

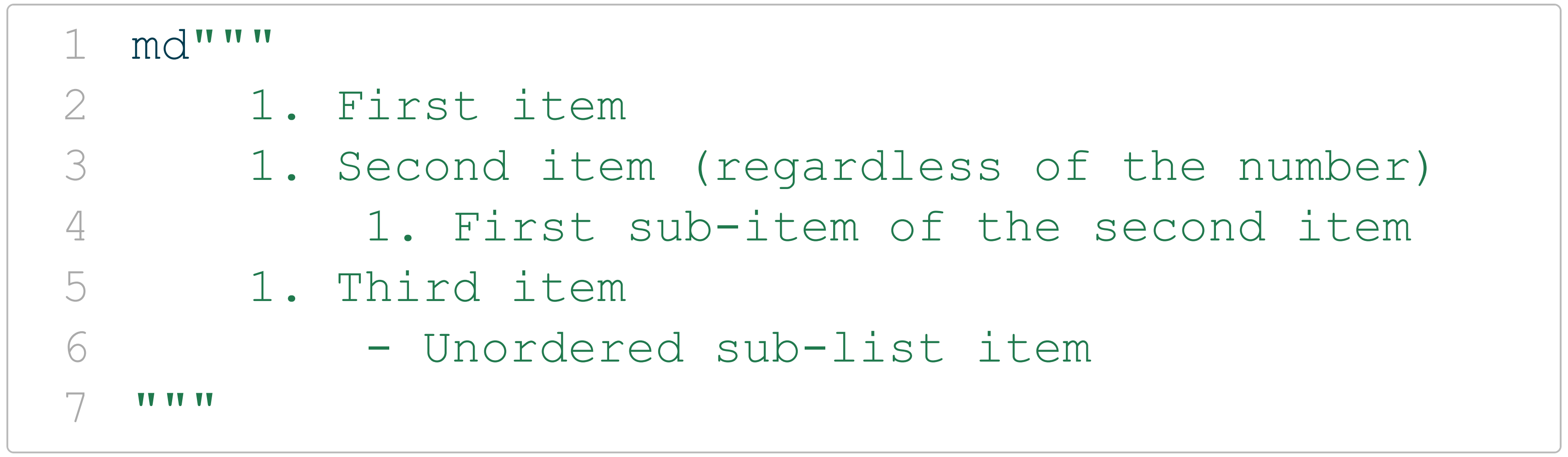
- **Text symbols** declare formatting (as in WhatsApp!)

```
md"""
    **bold** or *italics* or ***bold and italics***

    # Header
    ## Sub-header
    ### Sub-sub-header
"""
```

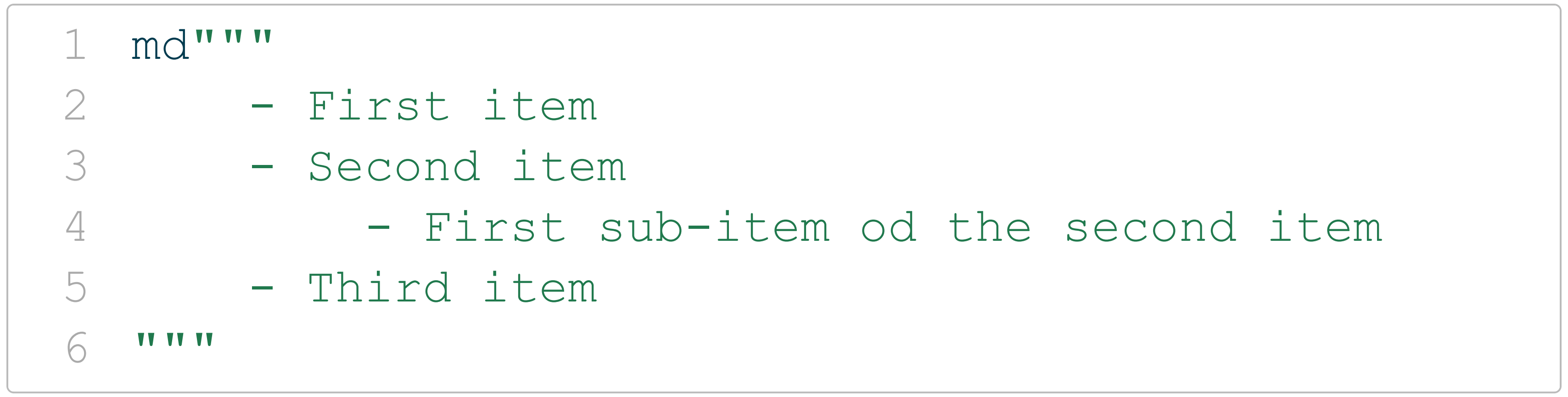
Cells with text: lists

- Ordered lists



```
md"""
    1. First item
    1. Second item (regardless of the number)
        1. First sub-item of the second item
    1. Third item
        - Unordered sub-list item
"""
```

- Unordered lists

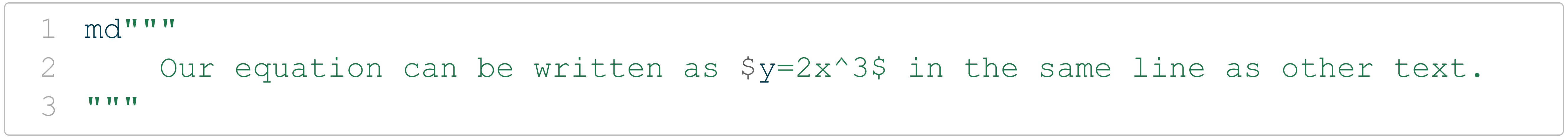


```
md"""
    - First item
    - Second item
        - First sub-item od the second item
    - Third item
"""
```

Cells with text: mathematics

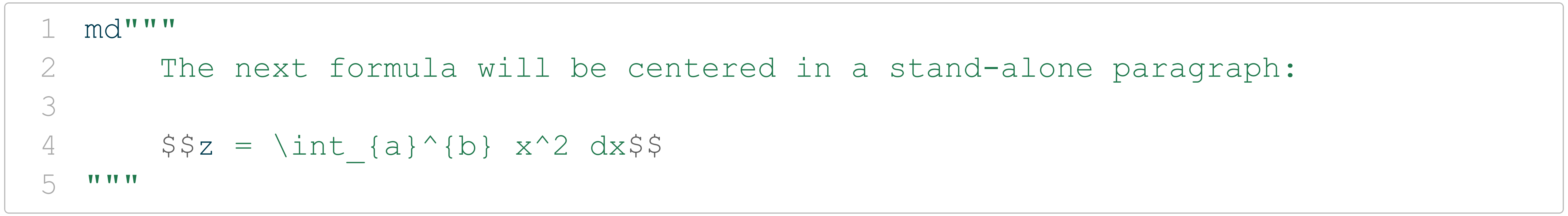
To typeset mathematics we can use **LaTeX syntax** *inside a Markdown block*

- Inline mode



```
md"""
    Our equation can be written as $y=2x^3$ in the same line as other text.
"""
```

- Display mode

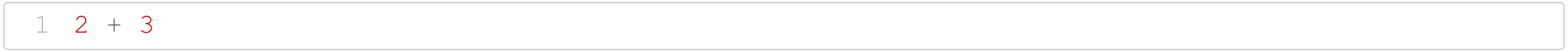


```
md"""
    The next formula will be centered in a stand-alone paragraph:

    $$z = \int_{a}^{b} x^2 dx$$
"""
```

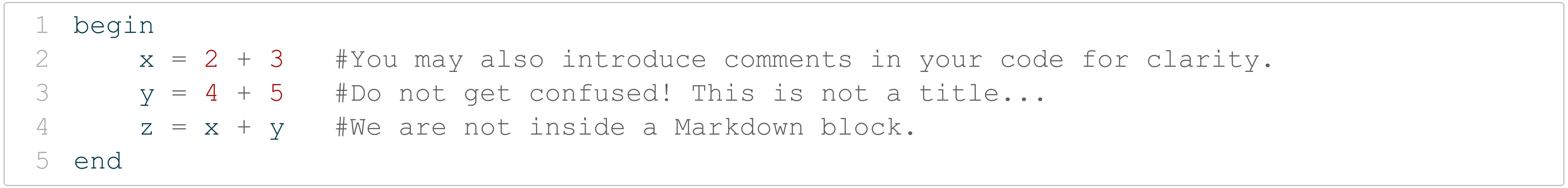
Cells with Julia code: input rules

- Each cell must contain a **single line of code**



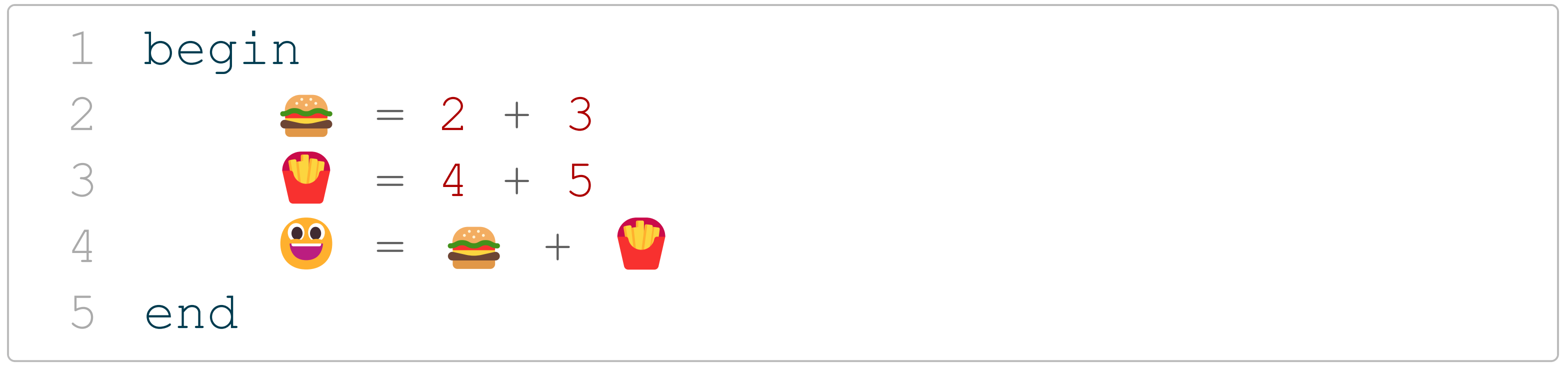
```julia
2 + 3
```

- Otherwise we need to use a `begin...end` block

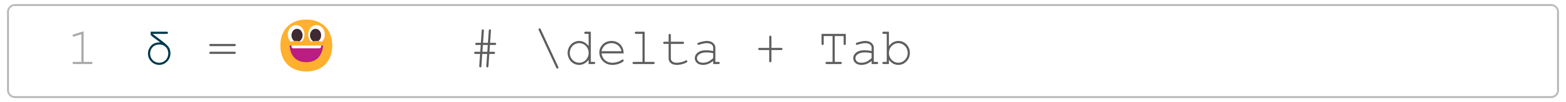


```julia
begin
    x = 2 + 3    #You may also introduce comments in your code for clarity.
    y = 4 + 5    #Do not get confused! This is not a title...
    z = x + y    #We are not inside a Markdown block.
end
```

- Any Unicode character or even Emojis may be used in your code



```julia
begin
    🍔 = 2 + 3
    🍟 = 4 + 5
    😃 = 🍔 + 🍟
end
```

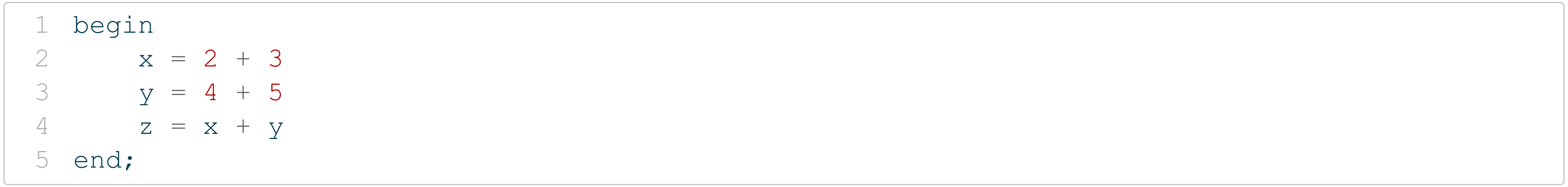


```julia
δ = 😃      # \delta + Tab
```

Cells with Julia code: run and control output

- To **run a cell** (i.e., execute the code inside), press `Shift + Enter` or hit the ▶ icon in the bottom-right corner of the cell

- To hide the output use `;` at the very end like in:



```
begin
    x = 2 + 3
    y = 4 + 5
    z = x + y
end;
```

Algebra and Julia: calculator way

- Maybe you are not, but Julia is an expert in Algebra 😏

- You may use Julia as a super-power calculator

  - Defining a Matrix

```
Romeo = [1 2; 3 4]
```

  - Calculate the determinant of `Romeo`

```
det_Romeo = 1 * 4 - 2 * 3
```

  - Write `Romeo`’s adjunct matrix

```
adj_Romeo = [4 -2; -3 1]
```

  - Invert `Romeo`

```
inv_Romeo = (1 / det_Romeo) * adj_Romeo
```

Algebra and Julia: built-in functions

- Are we correct?

```julia
inv_Romeo * Romeo
```

- Clever alternatives

```julia
inv(Romeo)
```

```julia
Romeo'  #Careful: this is the adjunct not the transpose()
```

```julia
begin
    using LinearAlgebra  #Several functions are provided: tr()
    det(Romeo)           #eigenvals(), eigenvecs(), factorize()
end
```

Using sophisticated packages

- **Pacakges** allow you to benefit from the work of others
- For **data analysis** we will use:
  - DataFrames.jl
  - CSV.jl
  - PlotlyJS.jl and/or Plotly.jl
  - Statistics.jl
- For **numerical solution** of complex systems of equations we will use NLSolve.jl