

Julia introduction

Why Julia?

Previously we used Matlab for CV1

- Easy to learn and use
- Many built-in tools for image processing and optimization
- Licence hassle
- Students needed to be in our lab / HRZ / CS-Pool or get there „own“ licence

Why Julia?

Julia promises to combine ease-of-use of Matlab with fast execution and open source licence.

We will use Julia **v0.5.0**

Tutorial with IJulia notebook

Try it: www.juliabox.org

Package manager

`Pkg.add("pkgname")` to install packages

`Pkg.update()` to update all packages

`using pkgname` or `import pkgname` to import package
into namespace

Common packages for our course

- [PyPlot package](#) and [matplotlib documentation](#)
- [Images package](#) and [Function Reference](#)
- [JLD package](#) for loading/storing data

Differences to MATLAB

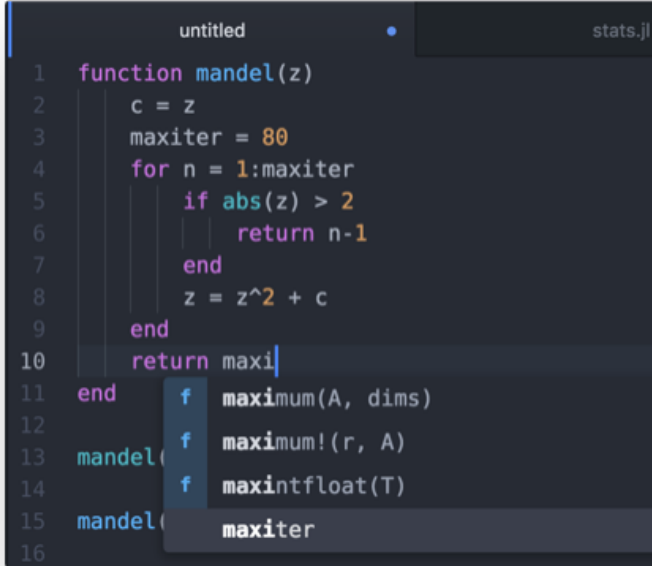
- Indexing via square brackets: `x[1]`
- Loops are fast
- Broadcasting is simple: [broadcasting](#)
- Multiple functions can be declared in a single file
- Packages need to be imported

- Take care of the current path and working directory (relative vs. absolute paths)
- Have second thoughts when indexing multi-dimensional arrays (row, column)
- Dynamic typing and implicit conversions can lead to unexpected data types
- Broadcasting is implicit and may give unexpected results

Juno

Walks like Python. Runs like C.

Juno builds on Julia's unique combination of ease-of-use and performance. Beginners and experts can build better software more quickly, and get to a result faster.



```
untitled • stats.jl
1 function mandel(z)
2     c = z
3     maxiter = 80
4     for n = 1:maxiter
5         if abs(z) > 2
6             return n-1
7         end
8         z = z^2 + c
9     end
10    return maxi
11 end
12
13 mandel(
14
15 mandel(
16
```

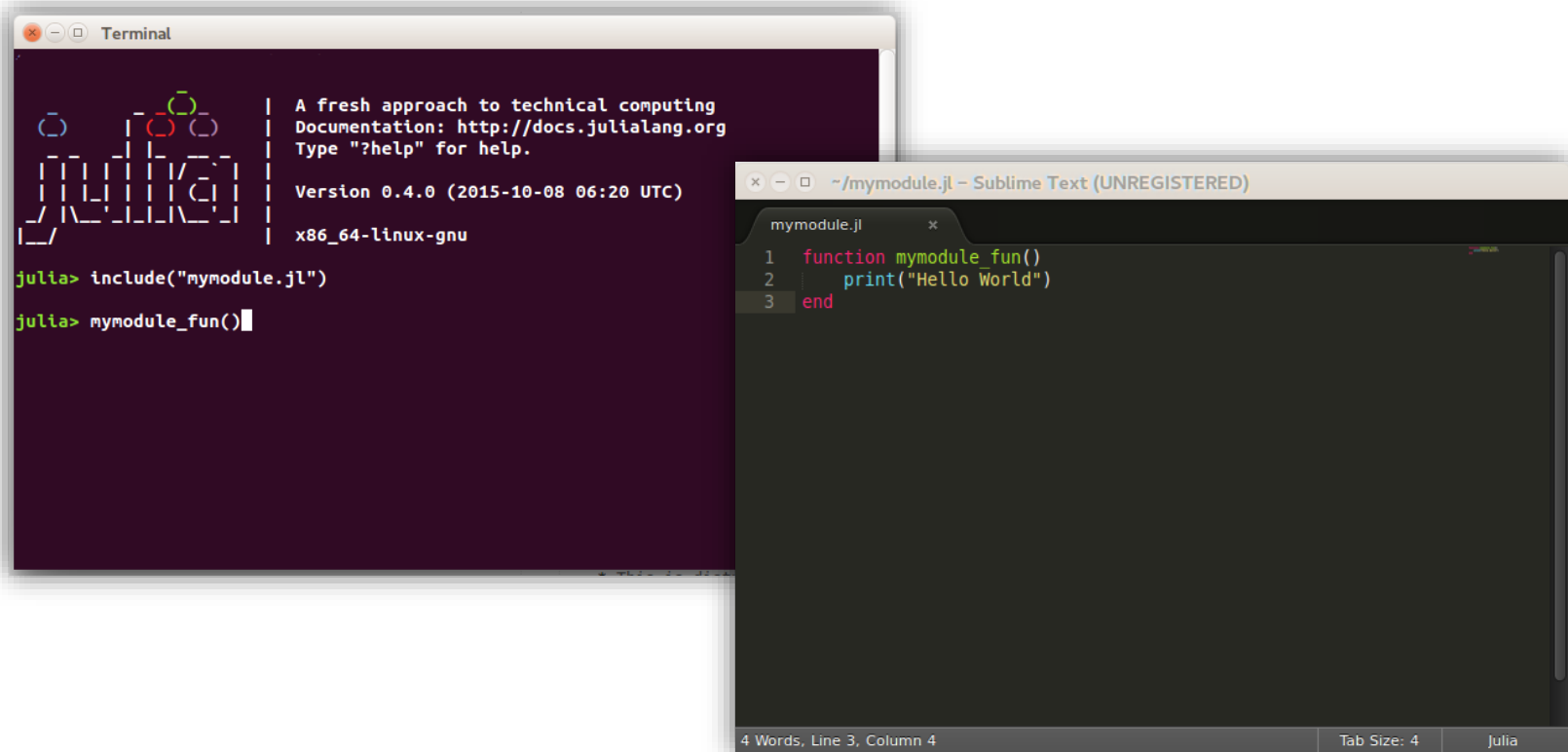
The screenshot shows the Juno IDE interface. The main editor window, titled 'untitled', contains a Julia function definition for a Mandelbrot set iteration. The code is as follows:

```
function mandel(z)
    c = z
    maxiter = 80
    for n = 1:maxiter
        if abs(z) > 2
            return n-1
        end
        z = z^2 + c
    end
    return maxi
end
```

A dropdown menu is visible below the 'return maxi' line, showing suggestions for the variable 'maxi':

- f maximum(A, dims)
- f maximum!(r, A)
- f maxintfloat(T)
- maxiter

The file explorer on the right shows a file named 'stats.jl'.



Debugging Julia

[Gallium.jl](https://gallium.jl)

- Interactive debugger (setting breakpoints, stepping through code, etc)
- Integrates with Juno

Console output

`@assert, @test`

Plotting things and **look at your results**

Best practices

- Use explicit *return* keyword in functions
- Exploit multiple dispatch
- Avoid global code: functions instead of scripting

Supplemental links

[Julia Homepage](#)

[Julia Documentation](#)

[Julia Package List](#)

[Performance Tips](#)

Group assignment in Moodle

- [Link](#)
- Groups of exactly two people
- Please form your groups until next Monday
- If haven't found a partner yet
 - Find him right after the exercise class
 - Use the discussion board