Northwestern
INFORMATION TECHNOLOGY

Julia – Week 1 Day 2

Sophie van Genderen, Assoc Computational Specialist
Julia Giannini, Computational Specialist

Northwestern IT Research Computing and Data Services

November 20th 2025

Julia

- **Quiz**
- Review
- Julia usage in 'the real world'
- Comparing Python to Julia
- Run a Julia script
- Julia Projects and Kernels

Quiz!

- https://create.kahoot.it/share/julia-review/158b9bb6-dddb-4d08-85fd-89bc2bee5235

Julia

- Quiz
- **Review**
- Julia usage in 'the real world'
- Comparing Python to Julia
- Run a Julia script
- Julia Projects and Kernels

Review: comparing Python to Julia

Key Indicator	Julia	Python
Maturity	Created in 2012	Created in 1991
Scope	General-purpose, but data-oriented	General-purpose and used for almost everything
Language Type	High Level, (Just in Time) Compiled	High Level, Interpreted
Typing	Dynamically-typed language, but also offers the ability to specify types (Static)	Dynamic, the type for a variable is decided at runtime
Open-source	Yes	Yes
Usage	Data Science and Machine Learning – especially work with data models	Mobile/web Dev, AI, Data Science, web scripting, game development, security ops.
Data Science	Math functions are easy to write and understand – no external libraries are needed for math functions	Requires NumPy or other libraries for advanced math
Performance	Fast development and production, high speed runtime, can handle millions of data threads	Fast for development, slow for production

Review: Connecting to Quest

- ssh
  - Done via terminal
  - ssh <netid>@login.quest.northwestern.edu

Review: Log into Quest and Load Julia

- ssh to Quest

```
[netid@quser]$ module load julia/1.11.4
[netid@quser]$ julia


 _       _ _(_)_     |  Documentation: https://docs.julialang.org
(_)     | (_) (_)    |
 _ _   _| |_  __ _   |  Type "?" for help, "]?" for Pkg help.
| | | | | | |/ _` |  |
| | |_| | | | (_| |  |  Version 1.11.4 (2025-03-10)
_/ |\__'_|_|_|\__'_|  |  Official https://julialang.org/ release
|__/                   |


julia> 2 + 4
```

Julia

- Quiz
- Review
- **Julia usage in 'the real world'**
- Comparing Python to Julia
- Run a Julia script
- Julia Projects and Kernels

What Companies Use Julia

- https://juliahub.com/case-studies

Zipline

Julia

- Quiz
- Review
- Julia usage in 'the real world'
- **Comparing Python to Julia**
- Run a Julia script
- Julia Projects and Kernels

Comparing Python to Julia

- Indexing
- Loops
- Functions / Methods
- **Installing / Loading / Using external packages**
- Dictionaries
- Global
- Modules
- Classes

Comparing Python to Julia - **Packages**

Python

To install a package:

```
Pip/pip3/mamba install <package_name>
```

To remove a package:

```
Pip/pip3/mamba uninstall <package_name>
```

To load installed packages:

```
import numpy as np
from time import time
```

https://rcdsdocs.it.northwestern.edu/systems/quest/software/python/python-quest.html

https://rcdsdocs.it.northwestern.edu/tutorials/software-management/conda-mamba-quest/mamba-conda-quest.html

Comparing Python to Julia – Virtual environments

Python

For python package management, virtual environments are isolated, self-contained collections of software for different projects with different requirements

```
(base) [@quser41 ~]$ mamba activate my_python310_env
(my_python310_env) [@quser41 ~]$ mamba list
# packages in environment at my_python310_env:
#
# Name                    Version                   Build  Channel
libblas                   3.9.0           31_h59b9bed_openblas    conda-forge
ncurses                   6.5                 he02047a_1    conda-forge
numpy                     2.2.6           py310hefbff90_0    conda-forge
openssl                   3.6.0                h26f9b46_0    conda-forge
python                    3.10.16         he725a3c_1_cpython    conda-forge
scipy                     1.15.2          py310h1d65ade_0    conda-forge
(my_python310_env) [@quser41 ~]$ mamba deactivate
(base) [@quser41 ~]$
```

Comparing Python to Julia – Virtual environments

Example: GPU-enabled software such as pytorch has specific requirements that have to be installed and configured carefully.

Avoid version conflicts and package incompatibility with virtual environments!

```
(base) [@quser41 ~]$ mamba activate my_python312_cuda118_env
(my_python312_cuda118_env) [@quser41 ~]$ mamba list
# packages in environment at my_python312_cuda118_env:
#
# Name                    Version                   Build  Channel
cuda-libraries            11.8.0                        0    nvidia
cuda-version              11.8                          3    nvidia
libblas                   3.9.0           16_linux64_mkl    conda-forge
ncurses                   6.5                 h59595ed_0    conda-forge
numpy                     2.0.1           py312h1103770_0    conda-forge
openssl                   3.3.1               h4bc722e_2    conda-forge
python                    3.12.3          hab00c5b_0_cpython    conda-forge
pytorch                   2.4.0           py3.12_cuda11.8_cudnn9.1.0_0    pytorch
pytorch-cuda              11.8                h7e8668a_5    pytorch
```

Comparing Python to Julia - **Packages**

Julia:

To install a package:

In Julia command line, type "]" and then

```
add <Package_name>
Pkg> add JSON StaticArrays
```

To Remove a package:

```
Pkg> rm <Package>
```

To get out of Pkg>, use Ctrl+C or backspace

To load installed packages:

```
import <Package>
using <Package>
```

Alternatively, in a script (or on the Julia command line)

```
using Pkg
Pkg.add("<package_name1>")

## For multiple packages
Pkg.add(["<package_name1>","<package_name2>
"])

using <package_name1>
```

Comparing Python to Julia – **Pin Packages**

Julia 

- Pinned packages will never be updated, they're "frozen" on that version
  - To pin: `Pkg> pin <Package name>`
  - To unpin: `Pkg> free <Package name>`
  - Be mindful of dependencies – when pinning packages, it will install all dependencies needed. Version conflicts may occur because of this. There are ways to address this.

- `.toml` files (these are written to the path `~/.julia/environments/v<version_no.>/` when you install packages)
  - `Manifest.toml`
  - `Project.toml`

- Can install multiple packages from `Project.toml`
  - `julia --project=/path/to/myproject`
  - If `myproject/Project.toml`

Comparing Python to Julia – Virtual environments

Julia 

- There is no direct equivalent for virtual environments in julia, but you can get very close with 'projects' which we will discuss later

- Note:
  - For (python) virtual environments, the environment itself is simply a directory, and all packages get installed within that directory
  - For (julia) projects, the project has a directory associated with it, but mostly consists of definition files. All packages get installed in your /home directory.

Julia – Looking up and Installing Packages

- https://juliapackages.com/
- Look up a package, ex: Enzyme.jl
  - https://juliapackages.com/p/enzyme
- Install package in Julia terminal:

Julia – Looking up and Installing Packages

- https://juliapackages.com/
- Look up a package, ex: Enzyme.jl
  - https://juliapackages.com/p/enzyme
- In julia:
  - `Julia> ]`
  - `pkg> add Enzyme`
  - `pkg> <BACKSPACE>`
  - `Julia> import Enzyme`
  - `Julia> using Enzyme`

Julia – Looking up and Installing Packages

- Try finding and installing another package!

Julia

- Quiz
- Review
- Julia usage in 'the real world'
- Comparing Python to Julia
- **Run a Julia script**
- Julia Projects and Kernels

Run a Julia Script

- Go to /projects/e33102
- Load the julia module
- Take a look at the script
  - `cd example-code`
  - `cat <name-of-file>`

Run a Julia Script

- Go to /projects/e33102
- Load the julia module
- Take a look at the script
- Run the script
  - `julia <name-of-file>.jl`
- Make sure all packages used in the script are added to your Julia environment!

Julia

- Quiz
- Review
- Julia usage in 'the real world'
- Comparing Python to Julia
- Run a Julia script
- **Julia Projects and Kernels**

Julia Projects and Kernels

- Project
  - Folder with associated specific packages
  - .toml file
  - Julia scripts that rely on those packages
- Make a new project

Make a New Julia Project

- Project
  - Specific packages for the project you're working on
  - Make a directory yourself in /projects/e33102
    - `cd /projects/e33102`
    - `mkdir <NAME_OR_NETID>`
    - `cd <NAME_OR_NETID>`
  - In your directory, make a directory for your julia project
    - `mkdir my-project`
    - `cd my-project`
    - Load Julia module
    - `julia --project=.`
    - Add package `Plots`
- Run a script using your project (from your `my-project` directory)
  - `julia --project=. /projects/e33102/example-code/test_plots_simple.jl`

Share a Julia Project

- As long as `Project.toml` is available to you, you can share the project with other users

```
[netid@quser]$ cd /projects/e33102/<NAME>/
[netid@quser]$ mkdir friends_project
[netid@quser]$ cd friends_project
[netid@quser]$ cp /projects/e33102/example_code/Project.toml .
[netid@quser]$ julia -project=.
julia > using Pkg
julia > Pkg.status()
julia > Pkg.instantiate()
julia > exit()
[netid@quser]$ julia --project=. /projects/e33102/example-code/test_plots_simple.jl
```

Make a Kernel

```
[netid@quser]$ julia --project=.
julia > using Pkg
julia > using Conda
julia > using IJulia
julia > installkernel("Julia 1.11.4 - friends_project", "--
project=/projects/e33102/<NAME>/friends_project")
```

- Login to Quest onDemand and start a Jupyter session
- Create an iPython notebook and activate your kernel
- More information next time!
- Documentation: https://rcdsdocs.it.northwestern.edu/systems/quest/ondemand/ondemand.html

Thank You!

Questions about Quest? Email us at: quest-help@northwestern.edu