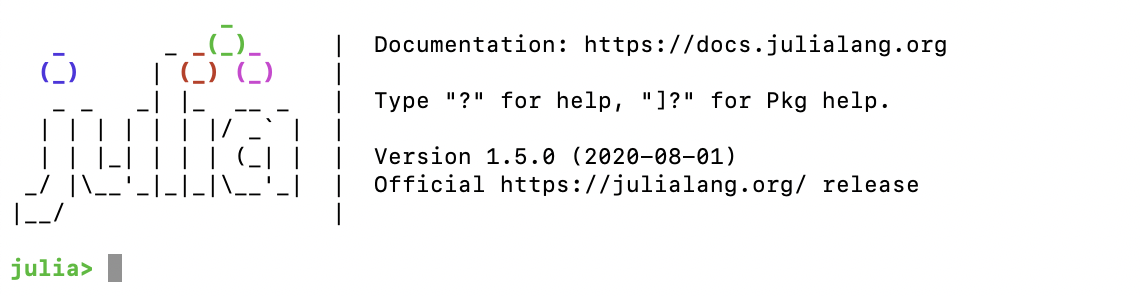
Getting Started with Julia, JuMP, and IJulia notebooks

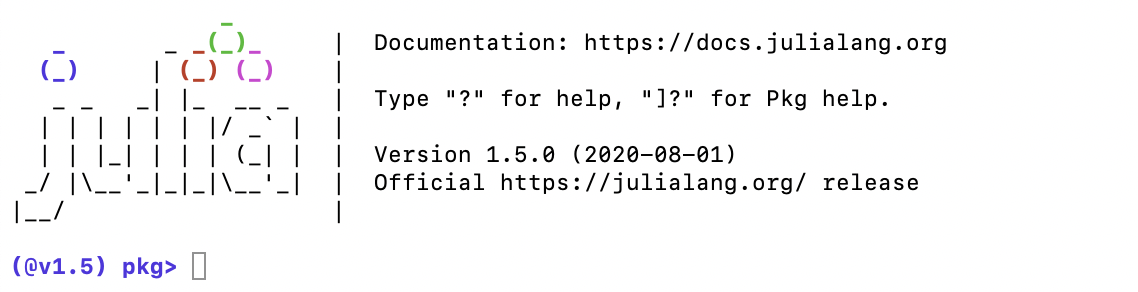
**1. Download and install Julia:**

* Go to <https://julialang.org/downloads/>
* Download and install the current stable release (v1.8.4 as of Dec 23, 2022)

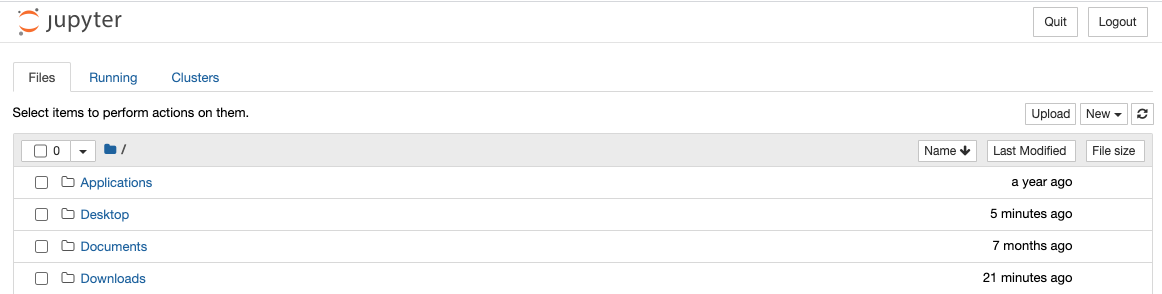
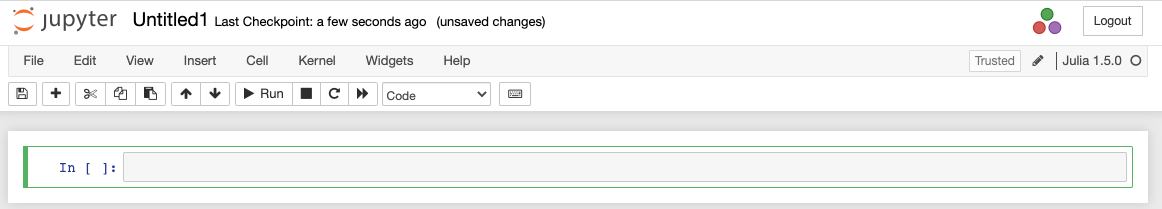
**2. Running Julia terminal:**

* Find and double click the Julia application you just installed, which should be titled Julia-1.8 (or Julia-1.Y where Y is the installed version number)
* This will open a new Julia terminal window or [REPL](https://docs.julialang.org/en/v1/stdlib/REPL/) (read-evaluate-print loop), similar to a command line terminal like this:  
  
* You can enter Julia syntax directly into the REPL and hit enter to evaluate, such as:  
  

**3. Installing packages:**

* Now you can install a set of packages useful for optimization and data management…
* To enter the pkg mode, type ‘]’. This will enter the pkg REPL mode.  
  
* *Install JuMP:* JuMP is a package for constructing and managing constrained optimization problems in Julia. You will get very familiar with JuMP throughout this course. Dive into [the documentation](https://jump.dev/). To install, type ‘add JuMP’ and enter.
* *Install DataFrames:* [DataFrames](https://juliadata.github.io/DataFrames.jl/stable/) are a useful data structure for storing tabular data with column headers (and will be familiar to many with experience with R or Python). To install, type ‘add DataFrames’ and enter.
* *Install Plots:* The [Plots](http://docs.juliaplots.org/latest/) package contains a variety of data visualization tools and is used for plotting and graphs. To install, type ‘add Plots’ and enter
* *Install open source solvers:* Install an open source solver, [HiGHS](https://highs.dev/) for use in linear and mixed integer programs. Type ‘add HiGHS’ and Enter.
  + In general, we will use HiGHS as our solver for example problems. Alternate choices include [Clp](https://github.com/jump-dev/Clp.jl) and [Cbc](https://github.com/jump-dev/Cbc.jl), for linear and mixed integer programs respectively, as well as [GLPK](https://www.gnu.org/software/glpk/), which works for both linear and mixed integer problems. Type ‘add Clp’ and Enter and then ‘add Cbc’ and Enter and then ‘add GLPK’ and Enter if you wish to test out the performance of these solvers.
  + Note that packages for interfacing with commercial solvers (which generally outperform open source solvers) are also available, including for [Gurobi.jl](https://github.com/jump-dev/Gurobi.jl) and [CPLEX.jl](https://github.com/jump-dev/CPLEX.jl). Both Gurobi and CPLEX are available for academic researchers for free ([for Gurobi, see here](https://www.gurobi.com/academia/academic-program-and-licenses/); and [CPLEX, see here](https://community.ibm.com/community/user/datascience/blogs/xavier-nodet1/2020/07/09/cplex-free-for-students?CommunityKey=ab7de0fd-6f43-47a9-8261-33578a231bb7&tab=)). You will have to obtain an academic license and install the commercial solver prior to adding the appropriate Julia package for interfacing between Julia/JuMP and these solvers. You can feel free to use these solvers in this course as well, and they are recommended for any large-scale modeling for academic research purposes.
* *Install IJulia*: [IJulia](https://juliahub.com/ui/Packages/IJulia/nfu7T/1.21.3) is a Julia backend combined with the [Jupyter](http://jupyter.org/) interactive environment (also used by [IPython](http://ipython.org/)). This combination allows you to interact with the Julia language using Jupyter/IPython's powerful [graphical notebook](http://ipython.org/notebook.html), which combines code, formatted text, math, and multimedia in a single document. We will use IJulia notebooks for tutorial code in this course, and it is a good environment for light code development work. It also works with [JupyterLab](https://jupyterlab.readthedocs.io/en/stable/), a Jupyter-based integrated development environment for notebooks and code. Install IJulia by typing ‘add IJulia’ and Enter.
* Now you can exit the Pkg mode by hitting Backspace to return to the normal REPL mode.
* Note that you can enter Pkg mode in the REPL at any time by entering ‘]’.
* While in Pkg mode, you can check the status and version number of your installed packages by typing ‘status’ and then enter; update packages to the latest versions by typing ‘update’ and enter, or update specific packages by typing ‘update Example’ (e.g. ‘update JuMP’).
* To add a specific version of a package, you use ‘add Example@0.4’ where Example is the name of the package and the part following @ is the version number (e.g. ‘add JuMP@0.21’)
* More on managing packages can be found at <https://julialang.github.io/Pkg.jl/v1.1/managing-packages/>

**4. Using IJulia notebooks:**

* *Launching IJulia/Jupyter:* To launch IJulia/Jupyter, open a Julia terminal, and enter:  
  ‘using IJulia’  
  ‘notebook()’
* Note that the first time you do this, it will ask you if you want to install a minimal Python/Jupyter install via Conda and then launch IJulia/Jupyter. **If you have already installed Python/Jupyter, see note below.**
* Alternatively, you may open Jupyter at your computer’s command prompt / terminal via: ‘jupyter notebook’.
* IJulia/Jupyter notebooks run locally on your computer using your default browser. You can also run IJulia using JupyterLab, which is custom built for this. See <https://jupyterlab.readthedocs.io/en/stable/> for installation and guide to use.
* The Notebook Dashboard, which can be launched by repeating the above looks as follows, giving you access to files on your local computer and saved notebook projects:  
  
* To open a new IJulia notebook, go to the New dropdown box in the upper right and select Julia v1.X.Y (your installed version, e.g. Julia 1.5.0):  
  
* This will open a new IJulia notebook project in a new tab:  
  
* Try running a command to ensure the kernel is running properly: ‘2+2’
* Please take time to go through the Jupyter documentation and familiarize yourself with the structure of IJulia/Python notebooks, the different types of cells, how to evaluate code cells and write/edit/view markdown cells here: <https://jupyter-notebook.readthedocs.io/en/stable/examples/Notebook/examples_index.html>

**Note on IJulia for Python users:**

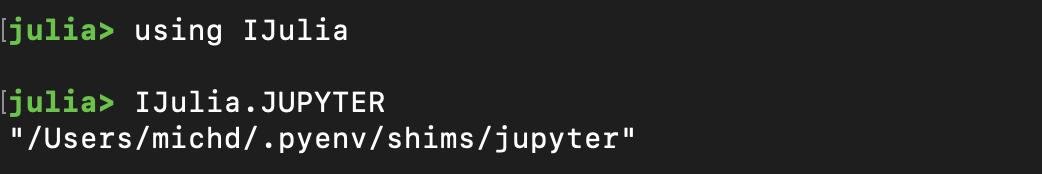
* Jupyter notebooks can be written in a wide number of programming languages, accessed via Jupyter “kernels” which connect to the underlying language processor. A list of the Jupyter kernels installed can be seen by clicking ‘New’:



* (Jupyter can, in fact, run multiple languages within a single notebook with the appropriate commands, passing variables and loading packages relatively seamlessly: <https://blog.jupyter.org/i-python-you-r-we-julia-baf064ca1fb6>)
* If you have already installed Python and Jupyter, you may use that Jupyter installation instead of creating a new one with Julia by running in the REPL: ENV["JUPYTER"]="/path/to/jupyter" and

Pkg.build("IJulia")

* **If you are having trouble running an IJulia notebook, you should check your Jupyter installation and kernels**.
  + Running the following commands will tell you which jupyter installation you are using in Julia. (Note: yours may be different. Here, I am using the one installed through my python environment.)



* + If you are able to open a Jupyter notebook but not run any commands, you may try reinstalling the Julia kernel via ‘installkernel("Julia")’ in the REPL.
  + See <https://julialang.github.io/IJulia.jl/stable/manual/installation/> for more information on IJulia installation.

Resources:

**Julia resources:**

1. Julia Cheat Sheet <https://juliadocs.github.io/Julia-Cheat-Sheet/>

2. Julia Academy [https://juliaacademy.com/](https://slack-redir.net/link?url=https%3A%2F%2Fjuliaacademy.com%2F) and [https://julialang.org/learning/](https://slack-redir.net/link?url=https%3A%2F%2Fjulialang.org%2Flearning%2F)

3. “Learn X in Y Minutes” Julia edition:<https://learnxinyminutes.com/docs/julia/>

4. Julia Discourse forum (for help/questions/forum):<https://discourse.julialang.org/>

5. *Introduction to Programming* tutorial available in the three high-level, interpretive programming languages commonly used in scientific research: Python, Julia, and Matlab. [https://github.com/PraCTES/MIT-PraCTES/tree/master/demos/Lecture01\_Intro\_Programming](https://slack-redir.net/link?url=https%3A%2F%2Fgithub.com%2FPraCTES%2FMIT-PraCTES%2Ftree%2Fmaster%2Fdemos%2FLecture01_Intro_Programming)

**JuMP resources:**

1. JuMP documentation:<http://jump.dev/>

2. JuMPTutorials.jl [https://github.com/jump-dev/JuMPTutorials.jl](https://slack-redir.net/link?url=https%3A%2F%2Fgithub.com%2Fjump-dev%2FJuMPTutorials.jl) with video [https://www.youtube.com/playlist?list=PLbCV3cNeLZUTvD2\_yBZAf97hk5YBW\_ZRQ](https://slack-redir.net/link?url=https%3A%2F%2Fwww.youtube.com%2Fplaylist%3Flist%3DPLbCV3cNeLZUTvD2_yBZAf97hk5YBW_ZRQ)

3. Course Github with example JuMP optimization models for electric power systems:<https://github.com/east-winds/power-systems-optimization>

**IJulia and Jupyter Notebook resources:**

1. Examples and introduction to Jupyter notebooks:<https://jupyter-notebook.readthedocs.io/en/stable/examples/Notebook/examples_index.html>

2. IJulia installation guide:<https://julialang.github.io/IJulia.jl/stable/manual/installation/>