Install Julia and Python Together

Table of Contents

Introduction

In this document I go over the details of getting the latest version of Julia installed with packages slanted towards DSP and installing Python and Jupyter Lab installed in the Julia environment using `Conda.jl`. By using `PyCall.jl` you will be able to use the Scipy stack and the package scikit-dsp-comm

Download the Installer

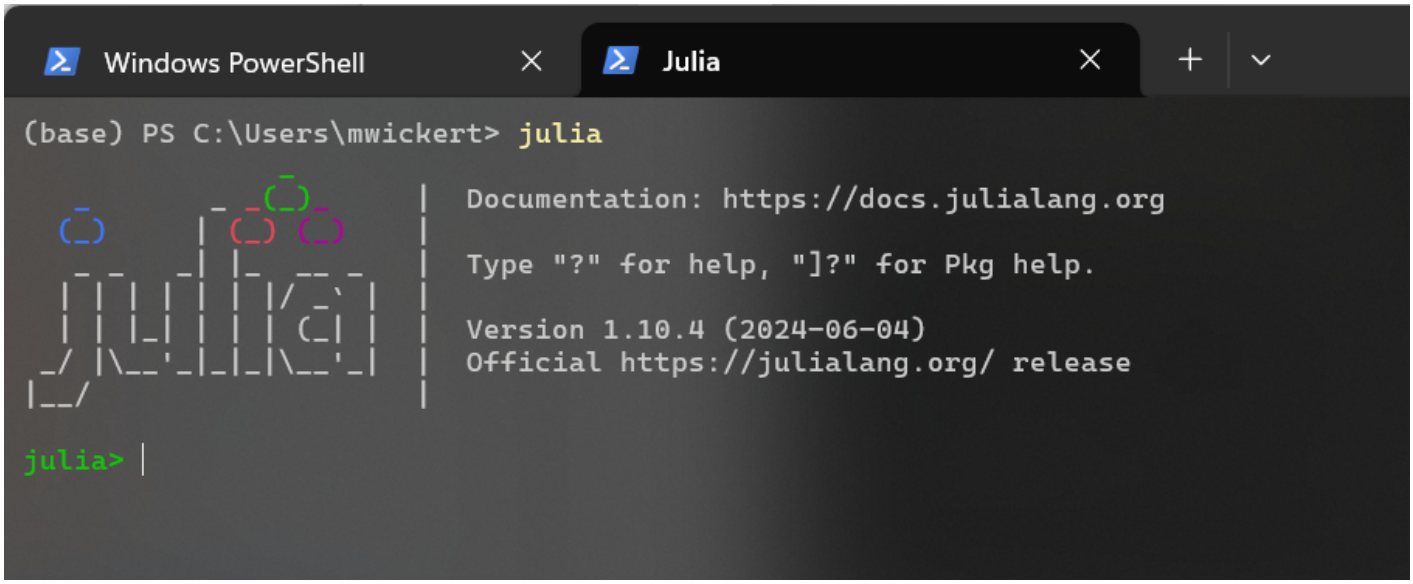
The starting point is Download Julia (julialang.org) and choosing your OS. The example here follows a Windows 11 install. I use Julia on Win11, MaxOS, Linux (Ubuntu), and find that the Julia REPL (read-execute-print-loop) interface is virtually the same on all OS's.

Current stable release: v1.10.4 (June 4, 2024)

Release notes | GitHub tag | SHA256 checksums | MD5 checksums

Platform	64-bit	32-bit
Windows [help]	installer, portable	installer, portable
macOS x86 (Intel or Rosetta) [help]	.dmg, .tar.gz	
macOS (Apple Silicon) [help]	.dmg, .tar.gz	
Generic Linux on x86 [help]	glibc (GPG), musl[1] (GPG)	glibc (GPG)
Generic Linux on ARM [help]	AArch64 (GPG)	
Generic Linux on PowerPC [help]	little endian (GPG)	
Generic FreeBSD on x86 [help]	.tar.gz (GPG)	

- Unpack and and run the installer
- With installation completes launch the REPL using start Julia launcher or my preference is from the terminal, e.g., Windows Powershell, bash shell, or z-shell
- On Windows a Powershell (PWSH) launch produces:



Where is Julia on Your System?

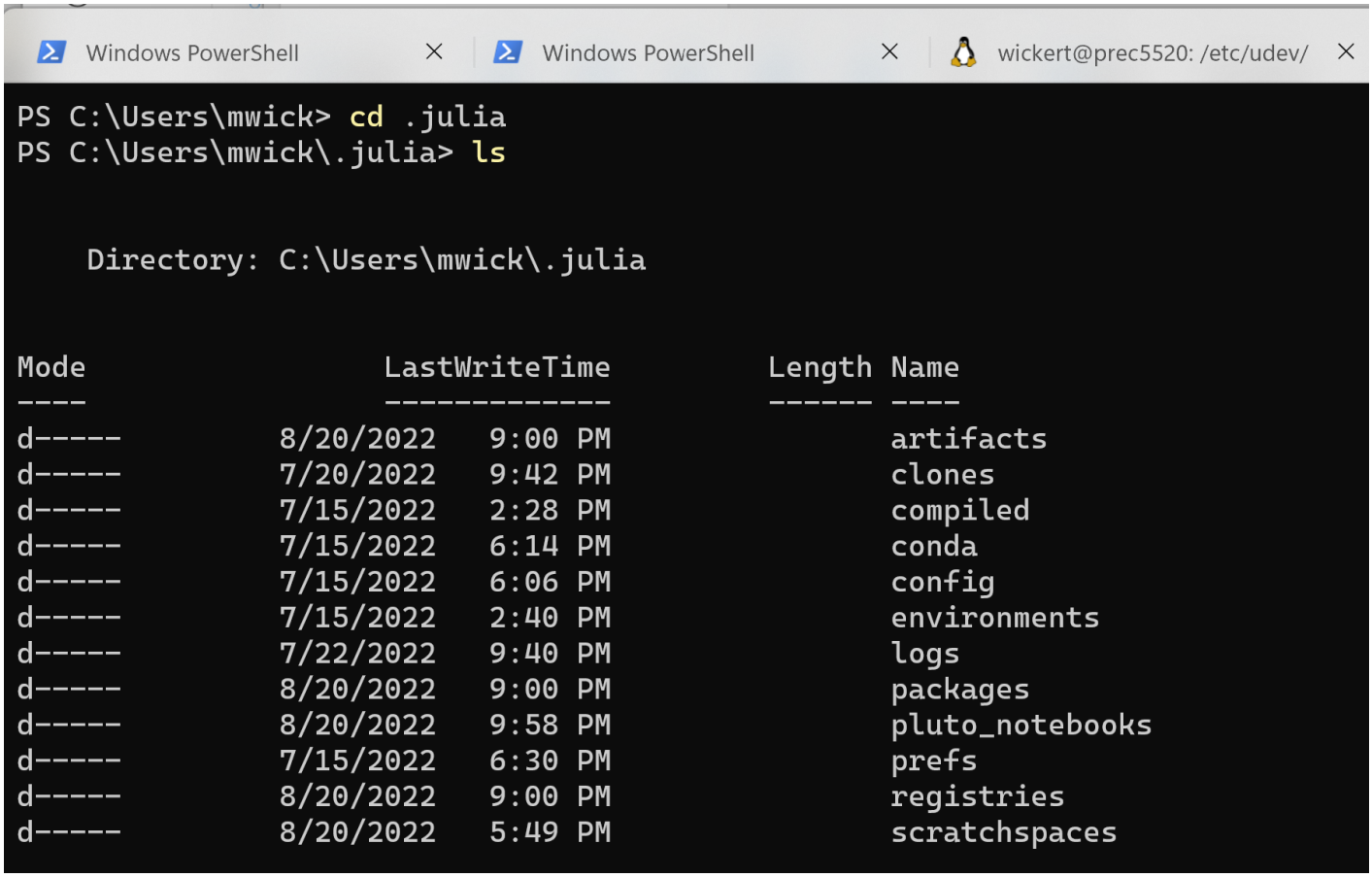
All installs of Julia place the code base in your home or *Users* folder titled `.Julia`. This is a "hidden" file but with PWSH the folders can be seen by typing `ls`

```
PS C:\Users\mwick> ls


    Directory: C:\Users\mwick


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
...
d-----         7/20/2022   9:42 PM                .julia
d-----         7/15/2022   6:45 PM                .jupyter
d-----         7/15/2022   7:04 PM                .matplotlib
d-----         7/15/2022   6:38 PM                .ms-ad
d-----         7/20/2022   9:28 PM                .vscode-insiders
d-r---         8/22/2022  10:04 PM                Desktop
d-r---         7/16/2022   9:53 PM                Documents
d-r---         8/22/2022   9:51 PM                Downloads
...
```

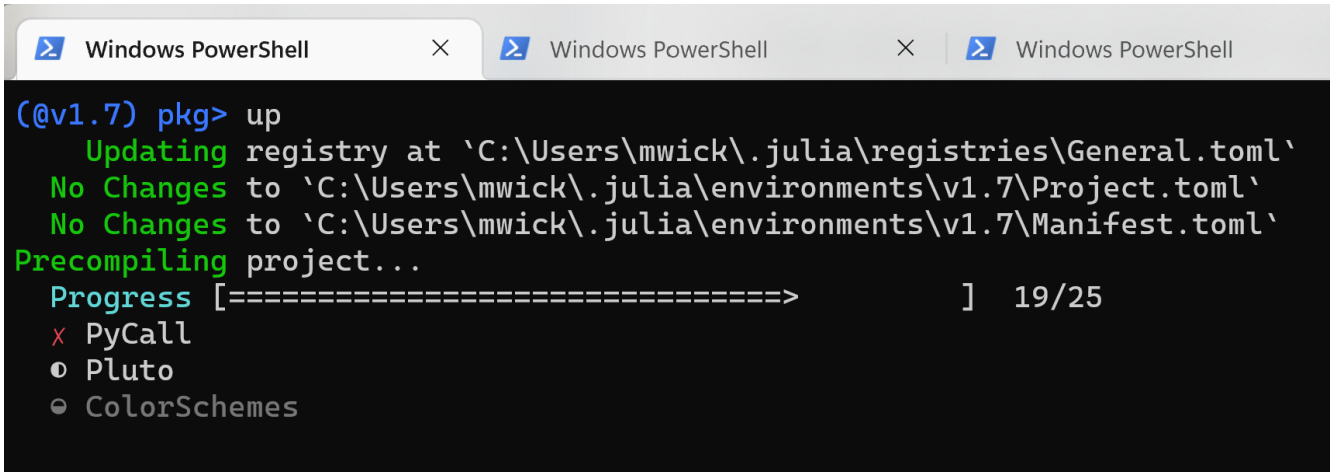
- Going inside the `.Julia` folder we find:



- **Note** At this point in the installation process you will definitely not see `conda`, `config`, and `Pluto_notebooks`. Others may not be present as well
- The additional folders will come as you now move on to install `Julia` packages

Installing Julia Packages

- To install packages you shift the REPL into the `pkg` mode by hitting `]` the key; to return to `Julia>` hit the `return`
- The package mode allows you to manage your Julia install
    - Install new packages using `add package-name`
    - Update your installed packages using `up`, i.e.,



  - **Note:** In the above image you see a red X. Somehow the compiled version of PyCall become corrupt so I had to rebuild it as I could no longer import it. Easily done in the REPL using

```
julia> import Pkg
julia> Pkg.build("PyCall")
```

  - Show the packages installed using `status`
  - Remove packages using `remove package-name`

- Add the packages:`Plots`, `Pluto`, `PlutoUI`, `OhMyREPL`, `PyCall`, `Conda`, `IJulia`, `FFTW`, `DSP`, `Polynomials`; **Note** all Julia packages begin with a capital letter (more than trivia in my opinion)
- Perhaps break the package installs into smaller groups or one at a time:

```
# hit ']' to switch package mode
pkg> add Plots, Pluto, PlutoUI, OhMyREPL
pkg> add PyCall, Conda, IJulia
pkg> add FFTW, DSP, Polynomials
```

- When you run `status` in the `pkg` mode you should see:

```
(@v1.10) pkg> status
Status `C:\Users\mwickert\.julia\environments\v1.10\Project.toml`
  [8f4d0f93] Conda v1.10.2
  [717857b8] DSP v0.7.9
  [31c24e10] Distributions v0.25.111
  [7a1cc6ca] FFTW v1.8.0
  [59287772] Formatting v0.4.3
  [7073ff75] IJulia v1.25.0
  [b964fa9f] LaTeXStrings v1.3.1
```

```
  [5fb14364] OhMyREPL v0.5.28
  [91a5bcdd] Plots v1.40.5
  [c3e4b0f8] Pluto v0.19.46
  [7f904dfe] PlutoUI v0.7.60
  [f27b6e38] Polynomials v4.0.11
  [438e738f] PyCall v1.96.4

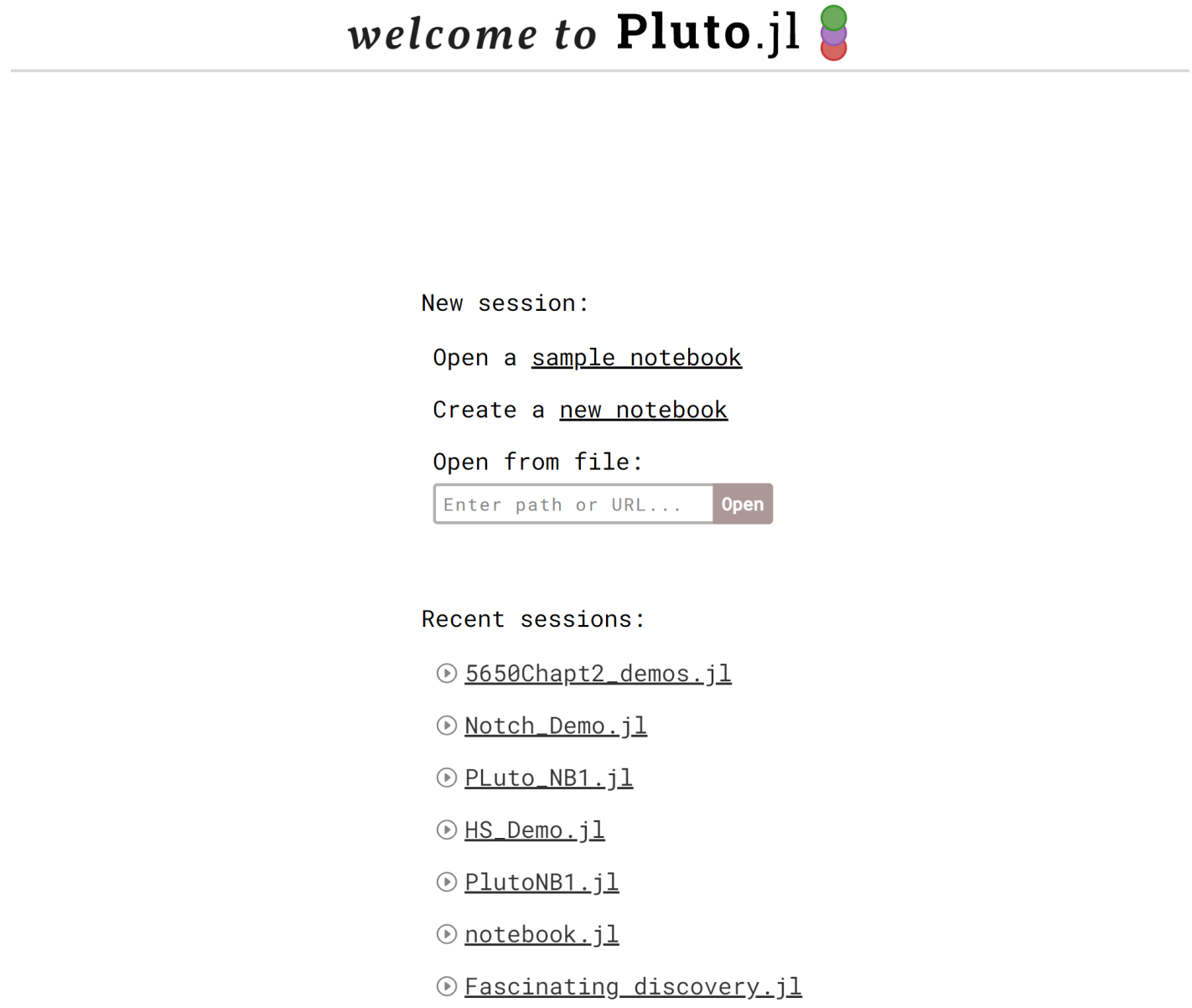
(@v1.10) pkg>
```

- You could start running code in Julia at this point, but first we will install Python packages inside the local `miniconda` package that got installed with `Conda.jl`
- At this point you can also run the `Pluto` notebook by importing the Pluto package into the REPL workspace and then running the command `run()`

```
julia> import Pluto

julia> Pluto.run()
[ Info: Loading...
┌ Info:
└ Opening http://localhost:1234/?secret=HylrACC9 in your default browser... ~
have fun!
┌ Info:
│ Press Ctrl+C in this terminal to stop Pluto
└
# following shutdown ...
^C┌ Info:
│
│ Closing Pluto... Restart Julia for a fresh session.
│
│ Have a nice day! 🎈
```

- When you type `Pluto.run()` a Web Browser will launch. On Windows it is best to use `Edge` of `Chrome`, and on MacOS I find it best to use `Chrome`



- A sample DSP notebook is linked later in this document.
- Also, you should see a `.julia` folder with subfloder `config`; inside this folder I place a file `startup.jl` to run things such as `OhMyREPL`, to provide a nicely colorized command line to help with delimiter matching and other things:

```
# My file startup.jl
using OhMyREPL
# add more if you desire
```

Installing Python Packages

- By installing Python packages you will be use `Julia` to launch `Jupyter Lab` and get access to both Julia and Python Jupyter `*.ipynb` notebooks

- Since we are installing Python packages housed inside our Julia install, we use the `Julia REPL` to do the work
- Import the `Conda` package and add a `conda-forge` *channel* for finding certain packages of interest

```
julia> import Conda
julia> Conda.add_channel("conda-forge")
# optionally see the list of channels
julia> Conda.channels()
# to remove channels use (not now)
julia> Conda.rm_channel("channel-name")
```

- Now we install a list of packages:
    1. The core scipy stack - `numpy`, `scipy`
    2. Jupyter lab - `jupyterlab`
    3. My package - `scikit-dsp-comm`
    4. For real-time audio in Python Jupyter notebooks - `pyaudio` and `ipywidgets`
- All of the packages except `pyaudio` are installed one at a time as discussed in https://morioh.com/p/ea09c0b24c18 or https://docs.juliahub.com/Conda/WZE3U/1.5.0/autodocs/, as shown below:

```
julia> Conda.add("package-name")
```

- To install `pyaudio` use `pip`:

```
julia> Conda.pip_interop(true)
julia> Conda.pip("install","pyaudio")
```

- All of the above package installs come with dependencies, so even more packages are getting loaded inside the `conda` folder `.Julia`
- With all the Python installs complete you can check the list of packages inside your Julia install; important packages are shown

```
julia> Conda.list()
[ Info: Running `conda list` in root environment
# packages in environment at C:\Users\mwick\.julia\conda\3:
#
# Name                    Version                   Build  Channel
...         2.0.12             pyhd8ed1ab_0    conda-forge
colorama                  0.4.4              pyh9f0ad1d_0    conda-forge
conda                     4.13.0           py39hcbf5309_2    conda-forge
conda-package-handling    1.8.1            py39hb3671d1_1    conda-forge
```

```
...                4.7.2              pyhd8ed1ab_0    conda-forge
jupyter                1.0.0           py39hcbf5309_7    conda-forge
jupyter_client         7.3.4             pyhd8ed1ab_0    conda-forge
jupyter_console        6.4.4             pyhd8ed1ab_0    conda-forge
jupyter_core           4.10.0          py39hcbf5309_0    conda-forge
jupyter_server         1.18.1            pyhd8ed1ab_0    conda-forge
jupyterlab             3.4.3             pyhd8ed1ab_0    conda-forge
jupyterlab_pygments    0.2.2             pyhd8ed1ab_0    conda-forge
jupyterlab_server      2.15.0            pyhd8ed1ab_0    conda-forge
jupyterlab_widgets     1.1.1             pyhd8ed1ab_0    conda-forge
...                2.1.1           py39hb82d6ee_1    conda-forge
matplotlib             3.5.2           py39hcbf5309_0    conda-forge
matplotlib-base        3.5.2           py39h581301d_0    conda-forge
matplotlib-inline      0.1.3             pyhd8ed1ab_0    conda-forge
...                   3.11                       py_1    conda-forge
portaudio              19.6.0              h0e60522_5    conda-forge
...                0.2.2             pyhd8ed1ab_0    conda-forge
pyaudio                0.2.12                  pypi_0    pypi
...             0.15.80         py39hb82d6ee_1007    conda-forge
scikit-dsp-comm        2.0.1             pyhd8ed1ab_0    conda-forge
scipy                  1.8.1           py39hc0c34ad_0    conda-forge
...             0.5.1                      py_1    conda-forge
webio-jupyter-extension   0.1.0          pyhd8ed1ab_0    conda-forge
...                1.5.2               h6255e5f_2    conda-forge
```

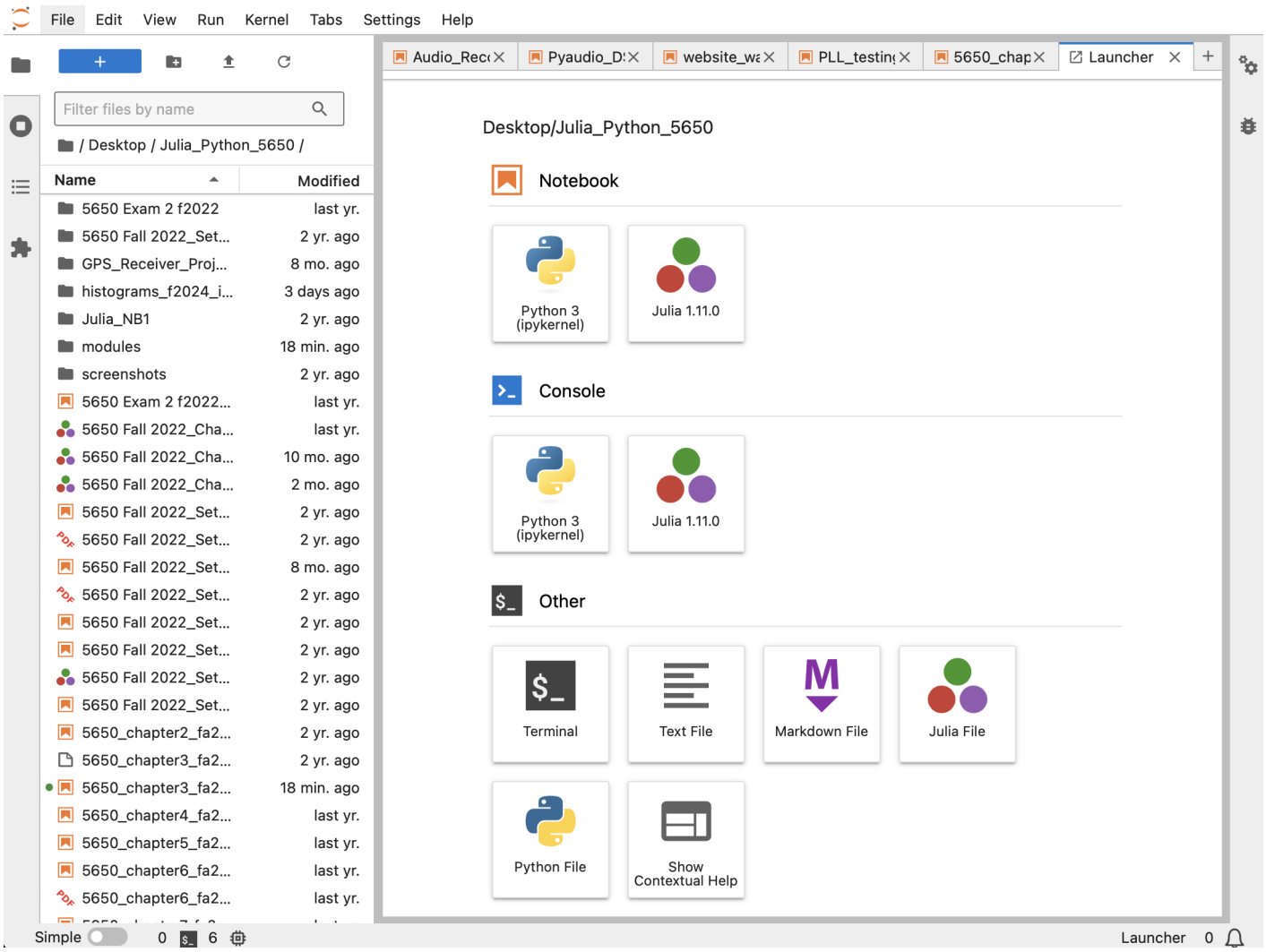
- Maintaining/updating the Python packages is something I have not explored

Launching Jupyter Lab using `IJulia`

- A lot. Of packages have been installed
- Now its time get get `Jupyter Lab` up and run some sample notebooks ()
- Starting from a fresh Julia REPL run these commands:

```
julia> import IJulia
julia> IJulia.jupyterlab()
```

- A Web browser (`edge`, `chome`, `firefox`) and you should see a `launcher` panel



- Using the file browser panel on the left (not s down above) navigate to a folder in say your `Documents` folder where you can drop some files from http://ece.uccs.edu/~mwickert/ece5650/notes/jupyter_pluto_notebook_samples.zip
- Create a subfolder called `modules` for three Julia module files I have created (a work in progress) from http://ece.uccs.edu/~mwickert/ece5650/notes/julia_modules.zip

Jupyter Lab Launcher Issues

You may experience issues when the `launcher` panel opens. In particular I have find when Julia is upgraded old version launchers remain or at the other extreme no Julia launchers are present for the Julia version you just installed.

- You can remove old launchers from the terminal by first listing them, then remove the old unwanted launchers

```
> jupyter kernelspec list
Available kernels:
  python3       /opt/miniconda3/envs/dsp-comm39/share/jupyter/kernels/python3
  julia-1.11    /Users/mwickert/Library/Jupyter/kernels/julia-1.11
```

```
> jupyter kernelspec remove julia-1.10
```

- If the needed kernel launcher does not appear in the list I suggest rebuilding the `IJulia package from within Julia

```
julia> import Pkg
julia> Pkg.build("IJulia")
    Building Conda ─→ `~/.julia/scratchspaces/44cfe95a-1eb2-52ea-b672-
e2afdf69b78f/b19db3927f0db4151cb86d073689f2428e524576/build.log`
    Building IJulia → `~/.julia/scratchspaces/44cfe95a-1eb2-52ea-b672-
e2afdf69b78f/1702f79fa30f56b68d5b2fd6fb3a9a14ff6f9130/build.log`
```

- Now when you run `jupyter kernelspec list` you should see the kernel of interest listed and when you launch jupyter lab the correct Julia launcher should appear

Setup and Using `vs code`

In this section I give instructions on setting up the `vs code` IDE for applications ranging from markdown editing, Julia code and debug development, Python code and debug, Julia and Python Jupyter notebook editing and debugging, and C++ coding and debugging. What makes `vs code` so useful is the vast array of `extensions` that can be installed. The extensions transform a very basic code editor into a power application development tool.

I personally use `vs code` for HTML files editing on my Website, LaTeX document writing and PDF production, such as my course notes, writing C and Python code for embedded systems. The list goes on.

This section will be a work in progress. The first subsection will be on using `vs code` to take a Jupyter notebook exported as a markdown folder (file + graphics) and do some simple editing to allow conversion to a PDF with page breaks where you want them for assignments.

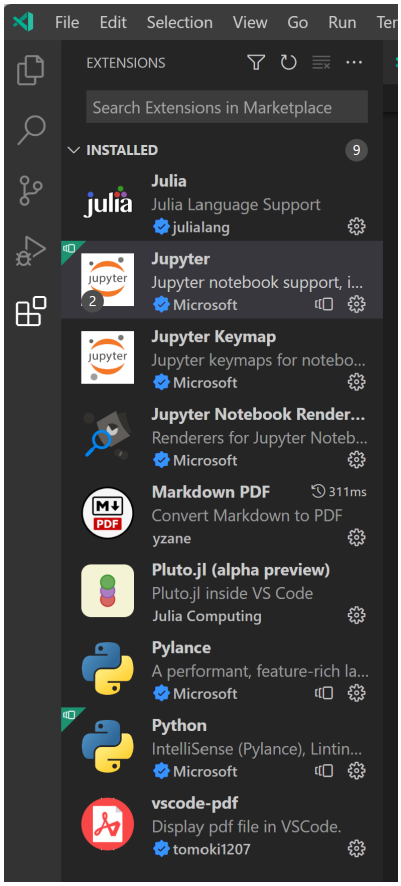
Install `vs code`

I recommend the *Insiders Edition* which gives you the option to update the `vs code` app everyday. If you don't want to see the update indicator in the lower left of the app window, go with the *Stable Build*.

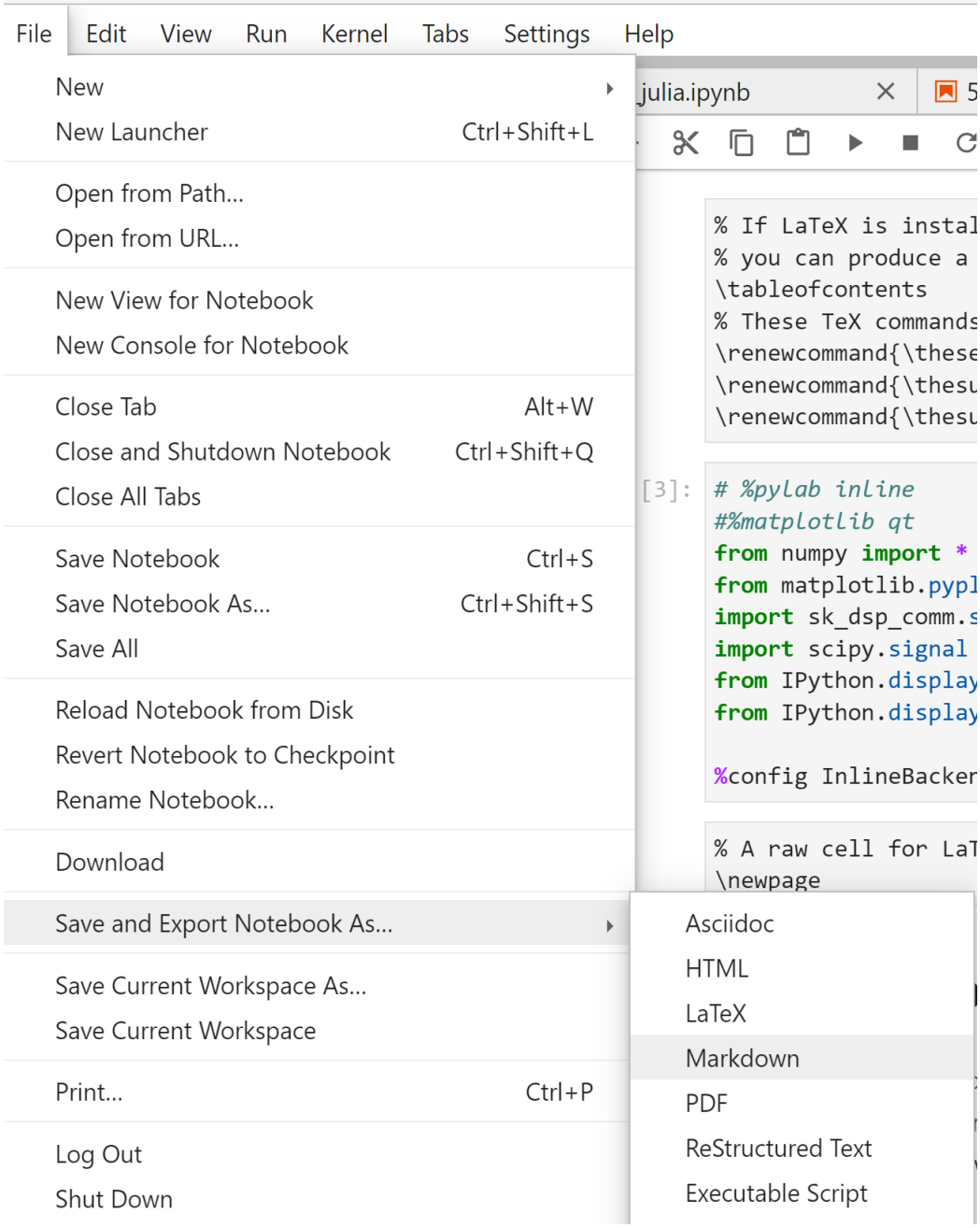
- Go to https://code.visualstudio.com and download and install the version of your choice, for the OS you are using (Mac, Windows, Linux); all versions work well
- We will now move on to installing specific extensions for code specific development

Post Processing Jupyter Markdown Exports into Fine Tuned PDF Docs

- Install `vs code` extensions to for now support just Markdown editing, PDF export, and PDF preview in `vs code`
- On my Win11 system, where I am editing now, I click on the *Blocks* icon on the far left column to list installed and/or install new `vs code` extensions



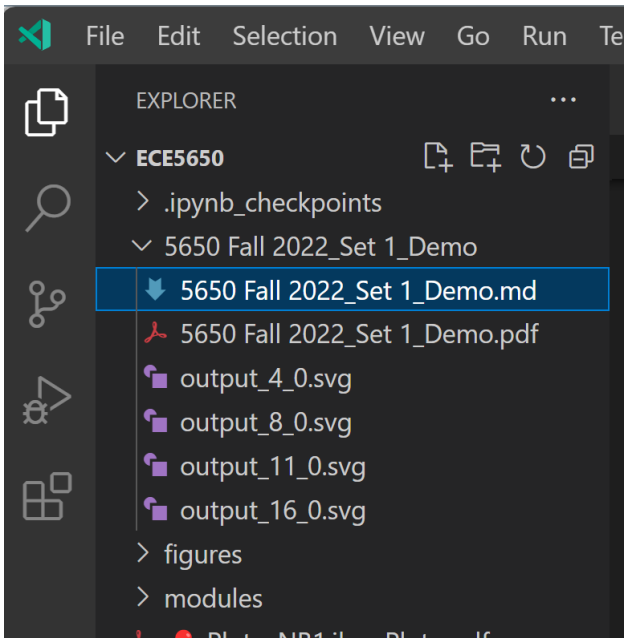
- The above shows all the extensions I currently have installed; using *Search Extensions* install for now just `Markdown PDF` and `vscode-pdf`
- Move now to Jupyter Lab and export a notebook to Markdown:



- The export produces a ZIP file that you can unpack and enter, e.g.,:

Name	Status	Date modified	Type	Size
5650 Fall 2022_Set 1_Demo	✓	9/1/2022 6:32 AM	Markdown File	6 KB
5650 Fall 2022_Set 1_Demo	✓	9/1/2022 6:32 AM	Microsoft Edge P...	137 KB
output_4_0	✓	8/31/2022 8:56 PM	Microsoft Edge H...	19 KB
output_8_0	✓	8/31/2022 8:56 PM	Microsoft Edge H...	50 KB
output_11_0	✓	8/31/2022 8:56 PM	Microsoft Edge H...	41 KB
output_16_0	✓	8/31/2022 8:56 PM	Microsoft Edge H...	48 KB

- In this folder you see a PDF already because I have already carrier out the step on am going to explain next
- The remaining files are the Markdown (MD) file you will edit in `vs code` and SVG graphics files that were produced when you ran Python or Julia code cells in Jupyter Lab
- In `vs code` open the folder that contains the MD export folder (this can be a folder that sits way about the MD folder); double click the file and you will be able to start editing it in `vs code`



- At the top of the file insert the three `javascript` lines of code (the lines are wrapped in this view only); Link to mathjax_scripts.md for easier entry into your MD files.

```
<script type="text/javascript"
src="http://cdn.mathjax.org/mathjax/latest/MathJax.js?config=TeX-AMS-
MML_HTMLorMML"></script>
<script type="text/x-mathjax-config"> MathJax.Hub.Config({ tex2jax:
{inlineMath: [['$', '$']]}, messageStyle: "none" });</script>
<script type="text/x-mathjax-config"> MathJax.Hub.Config({ tex2jax:
{displayMath: [['$$', '$$']]}, messageStyle: "none" });</script>
```

- The above code will allow the LaTeX math equation to appear properly formatted in the PDF export
- As needed you can/will insert the below HTML code to force page breaks in the final PDF; make sure to surround this code with blank lines in the MD file

```
<div style="page-break-after: always;"></div>
```

- It is also possible to add screen shots or other figures with adjustable zoom; maybe even scanned imaged of your handwritten homework problems

```
<img src="screenshots.png" alt="alt_text" style="zoom:50%;" />
```

- **Note:** This may be a way to bring everything together in one PDF on your own with out needing to use PDF merging software such as:
  Adobe Merge Files or similar
- Finally navigate to the MD file and open it, place the JS scripts at the top of the file
- Insert any page breaks and `png` files, etc.
- To get a quick MD preview in `vscode` enter `ctr-shift-V` with your cursor in the MD file
- Finally convert MD to PDF use the extensions you have installed by:
  - Placing the cursor in the MD file
  - Using ctrl-shift-P and select or type in the upper command dialog: *Markdown PDF: Export (pdf)*; see the screenshot below the MD file snippet immediately below

````
... snippet from MD file
x = ss.dimpulse(n-2)
stem(n,x)
ylim([-.1,1.1])
grid();
```



![svg](output_4_0.svg)


<div style="page-break-after: always;"></div>

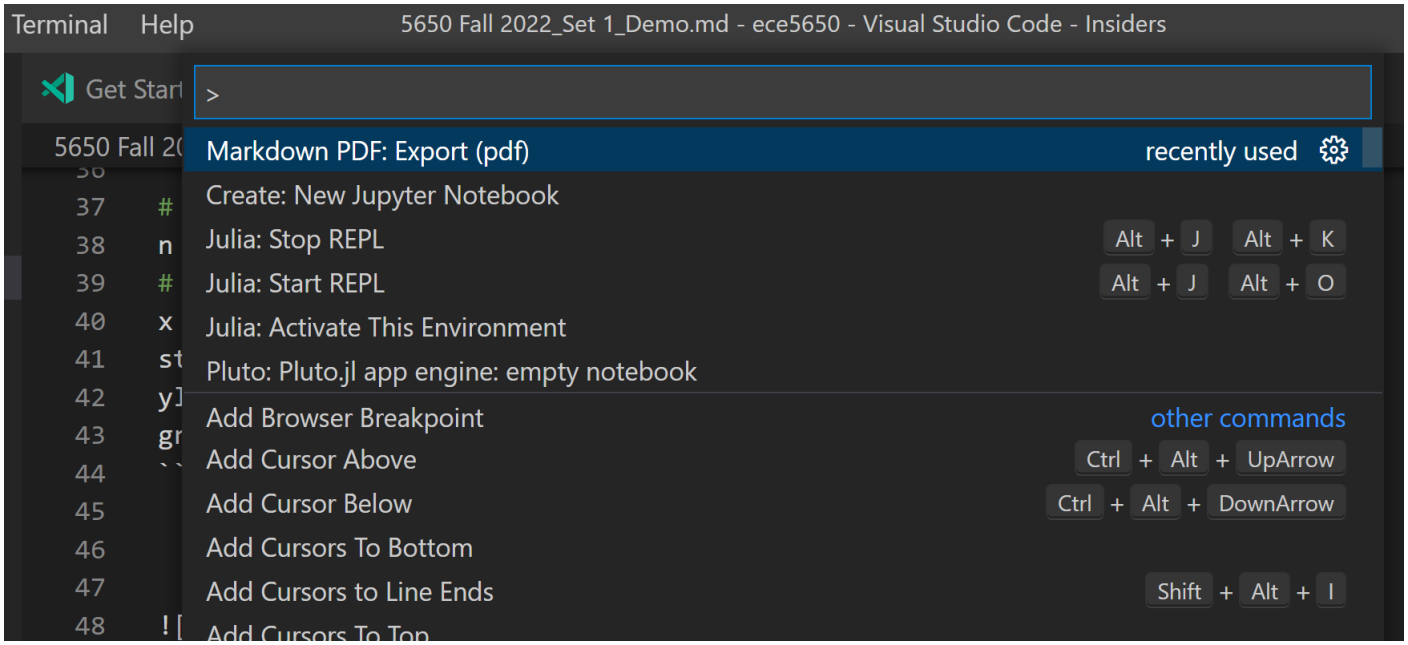
# Worked Problem

<img src="my_scan.png" alt="alt_text" style="zoom:50%;" />

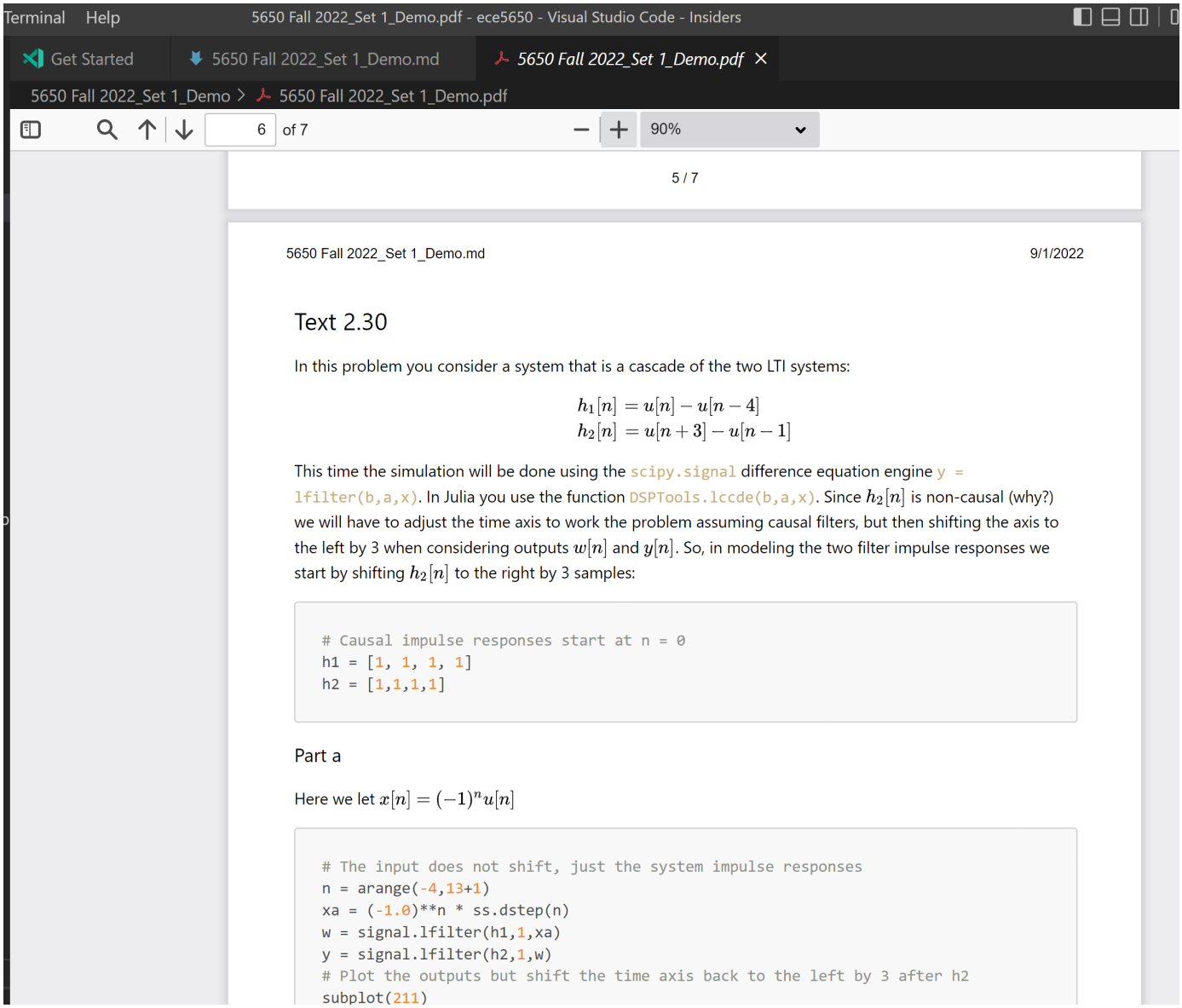
## Text 2.3
Consider



$$
    y[n] = x[n]\ast h[n]
$$
...
````
````



- The PDF file will be generated and will show in the left pane file *Explorer*; since you have installed the `PDF viewer` extension you should be able to open the PDF right inside `vs code` and can check that page breaks are occuring as expected
- You can of course edit any of the text or equations to suit your needs and then re-export the PDF



More vs code Extension Usage To Come