https://gitlab.com/gersonjferreira/computational-quantum-mechanics

Computational Quantum Mechanics

Prof. Gerson J. Ferreira

INFIS/UFU 2020/1

Using Jupyter Notebooks and Markdown

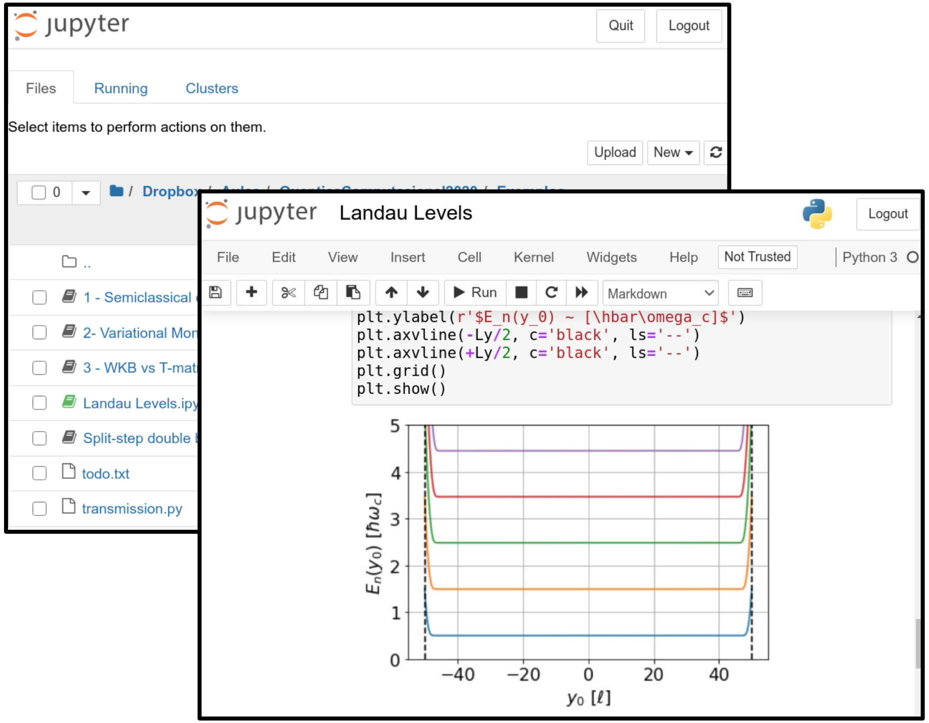
Python with Jupyter Notebooks + Markdown

Jupyter Notebook vs **JupyterLab**

→ Jupyter Notebook: one document on each browser tab

→ JupyterLab: single tab, multiple documents within sub-tabs

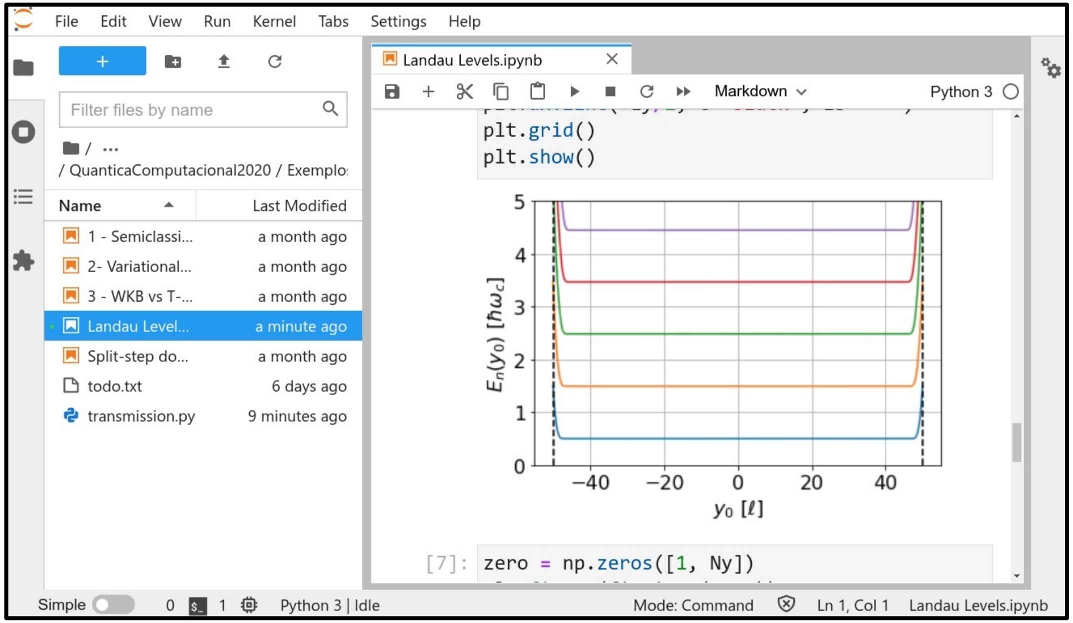
Also: JupyterLab has better visualization for CSV files, allows for plugins, etc...



Select items to perform actions on them.

- 1 - Semiclassical
- 2- Variational Mon
- 3 - WKB vs T-mat
- Landau Levels.ipy
- Split-step double
- todo.txt
- transmission.py

```
plt.ylabel(r'$E_n(y_0) ~ [\hbar\omega_c]$')
plt.axvline(-Ly/2, c='black', ls='--')
plt.axvline(+Ly/2, c='black', ls='--')
plt.grid()
plt.show()
```



/ QuanticaComputacional2020 / Exemplo:

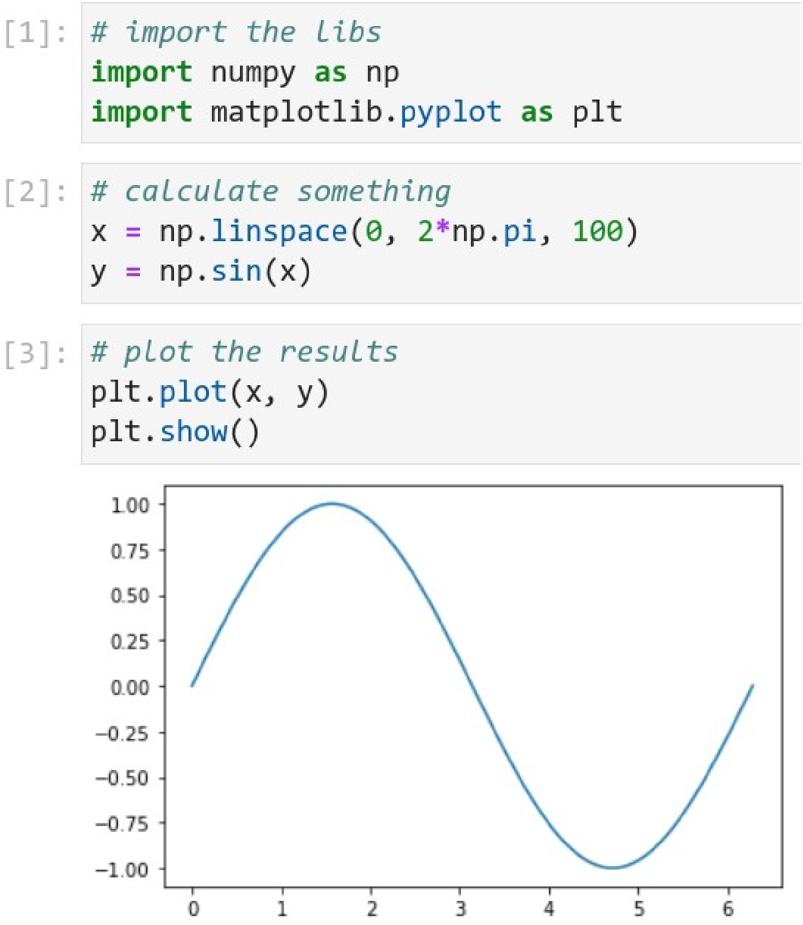
Name	Last Modified
1 - Semiclassi...	a month ago
2- Variational...	a month ago
3 - WKB vs T-...	a month ago
Landau Level...	a minute ago
Split-step do...	a month ago
todo.txt	6 days ago
transmission.py	9 minutes ago

```
plt.grid()
plt.show()
```

```
[7]: zero = np.zeros([1, Ny])
```

Simple 0 1 Python 3 | Idle Mode: Command Ln 1, Col 1 Landau Levels.ipynb

Jupyter: writing and running codes



```
[1]: # import the libs
     import numpy as np
     import matplotlib.pyplot as plt
```

```
[2]: # calculate something
     x = np.linspace(0, 2*np.pi, 100)
     y = np.sin(x)
```

```
[3]: # plot the results
     plt.plot(x, y)
     plt.show()
```

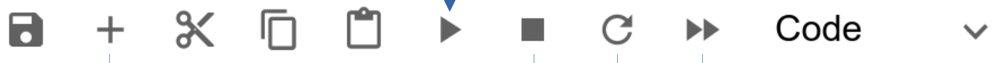
Organize the code in cells by its content

→ import all neeeded libs in single cell

→ run blocks of calculations

→ plot the results

To run each cell: **SHIFT+ENTER**



Add a cell below

Restart and re-run all

Restart

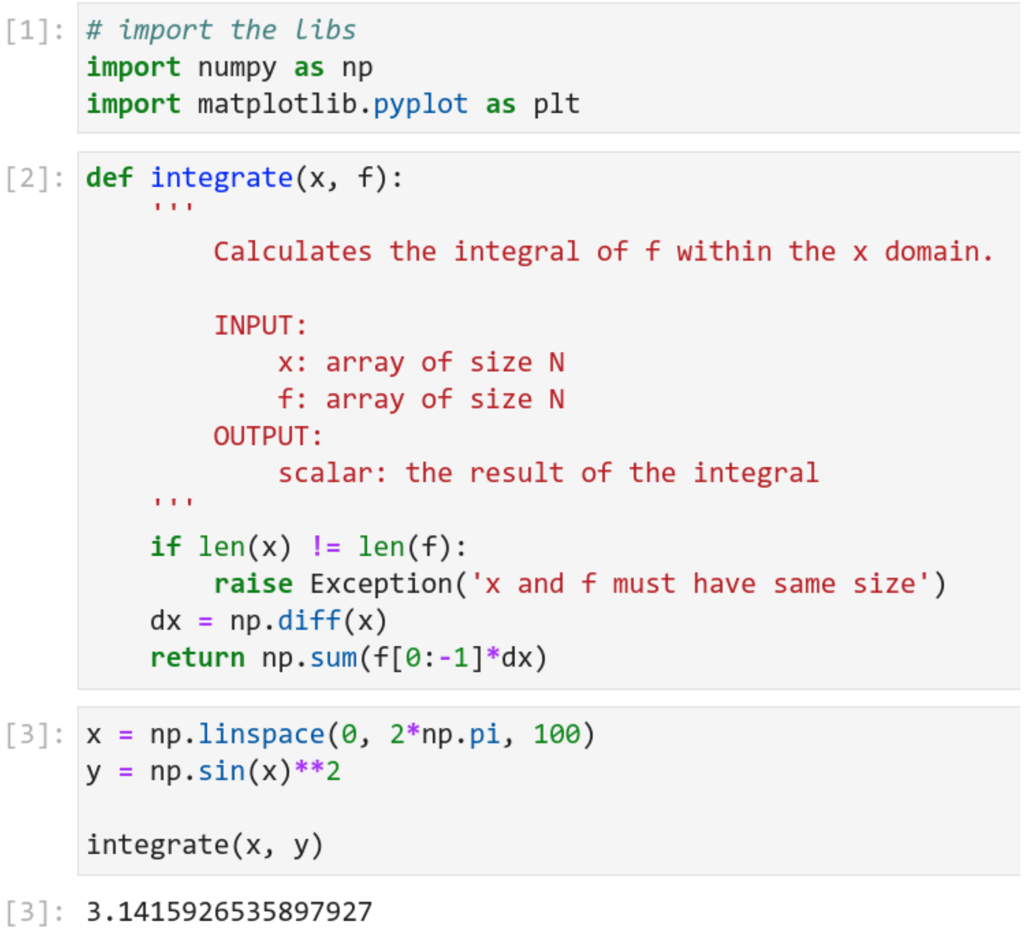
Stop the calculation!

To delete a cell:

→ While coding on it: ESC + D + D, or

→ Click its number to select and: D + D

Jupyter: autocomplete and docstrings



```
[1]: # import the libs
     import numpy as np
     import matplotlib.pyplot as plt
```

```
[2]: def integrate(x, f):
         '''
             Calculates the integral of f within the x domain.

             INPUT:
                 x: array of size N
                 f: array of size N
             OUTPUT:
                 scalar: the result of the integral
         '''
         if len(x) != len(f):
             raise Exception('x and f must have same size')
         dx = np.diff(x)
         return np.sum(f[0:-1]*dx)
```

```
[3]: x = np.linspace(0, 2*np.pi, 100)
     y = np.sin(x)**2

     integrate(x, y)
```

```
[3]: 3.1415926535897927
```

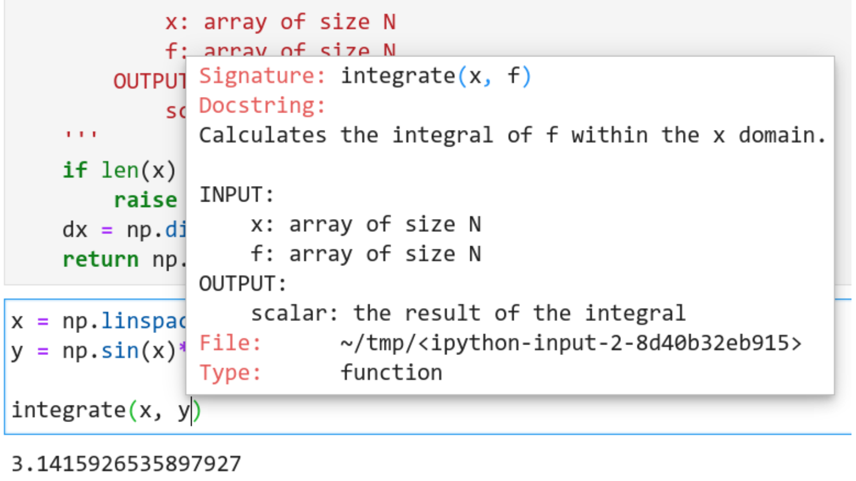
Organization:

→ Use a cell to define each function

→ Use docstrings to describe the function, and its input & output quantities

Autocomplete: press TAB

Show docstring: press SHIFT+TAB with the cursor within the function call



```
            x: array of size N
            f: array of size N
        OUTPUT
            sc
    '''
    if len(x)
        raise
    dx = np.di
    return np.
```

```
Signature: integrate(x, f)
Docstring:
Calculates the integral of f within the x domain.

INPUT:
    x: array of size N
    f: array of size N
OUTPUT:
    scalar: the result of the integral
File:      ~/tmp/<ipython-input-2-8d40b32eb915>
Type:      function
```

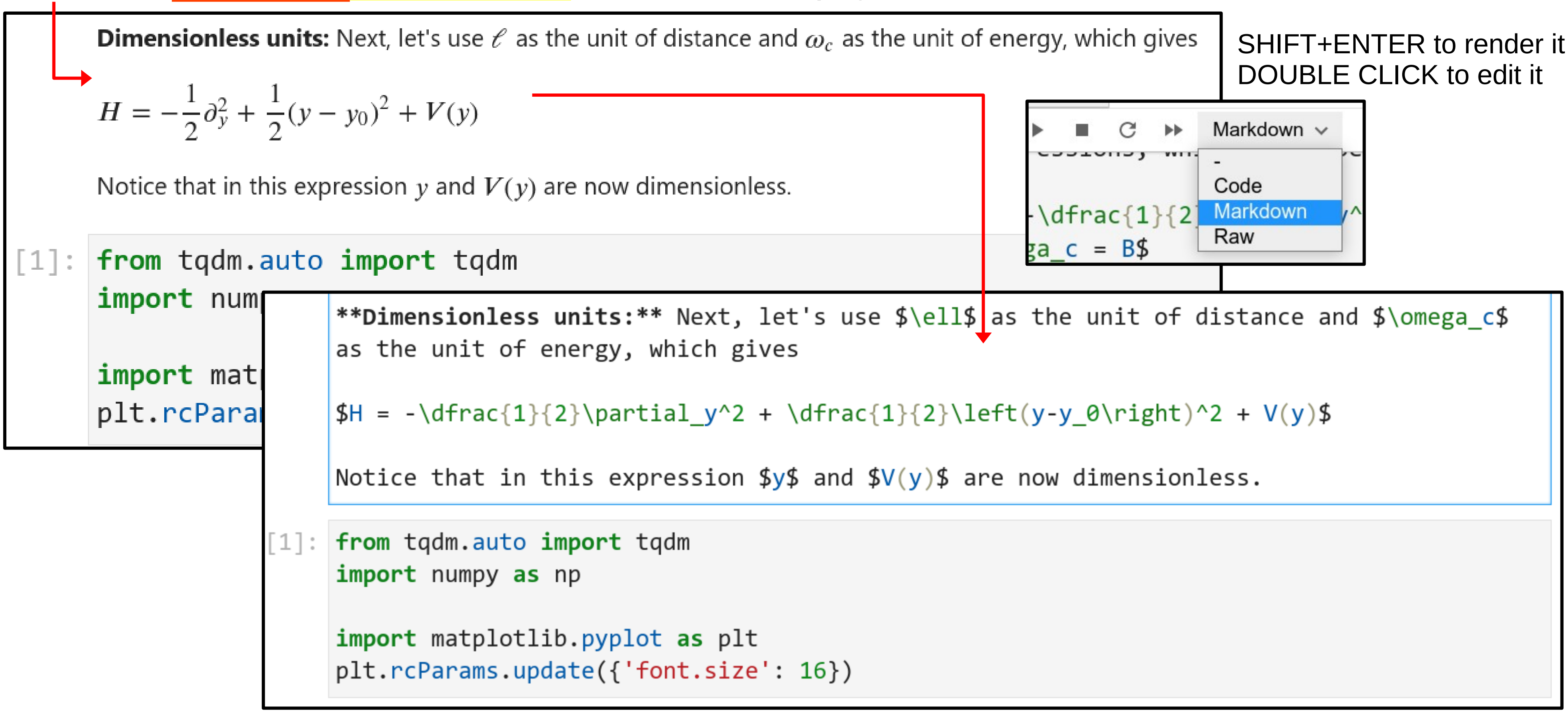
```
x = np.linspac
y = np.sin(x)*

integrate(x, y)
```

```
3.1415926535897927
```

Markdown + Latex

Let’s use Markdown to document the math and physics behind the codes

SHIFT+ENTER to render it
DOUBLE CLICK to edit it



Dimensionless units: Next, let's use ℓ as the unit of distance and ω_c as the unit of energy, which gives

$$H = -\frac{1}{2}\partial_y^2 + \frac{1}{2}(y - y_0)^2 + V(y)$$

Notice that in this expression y and $V(y)$ are now dimensionless.

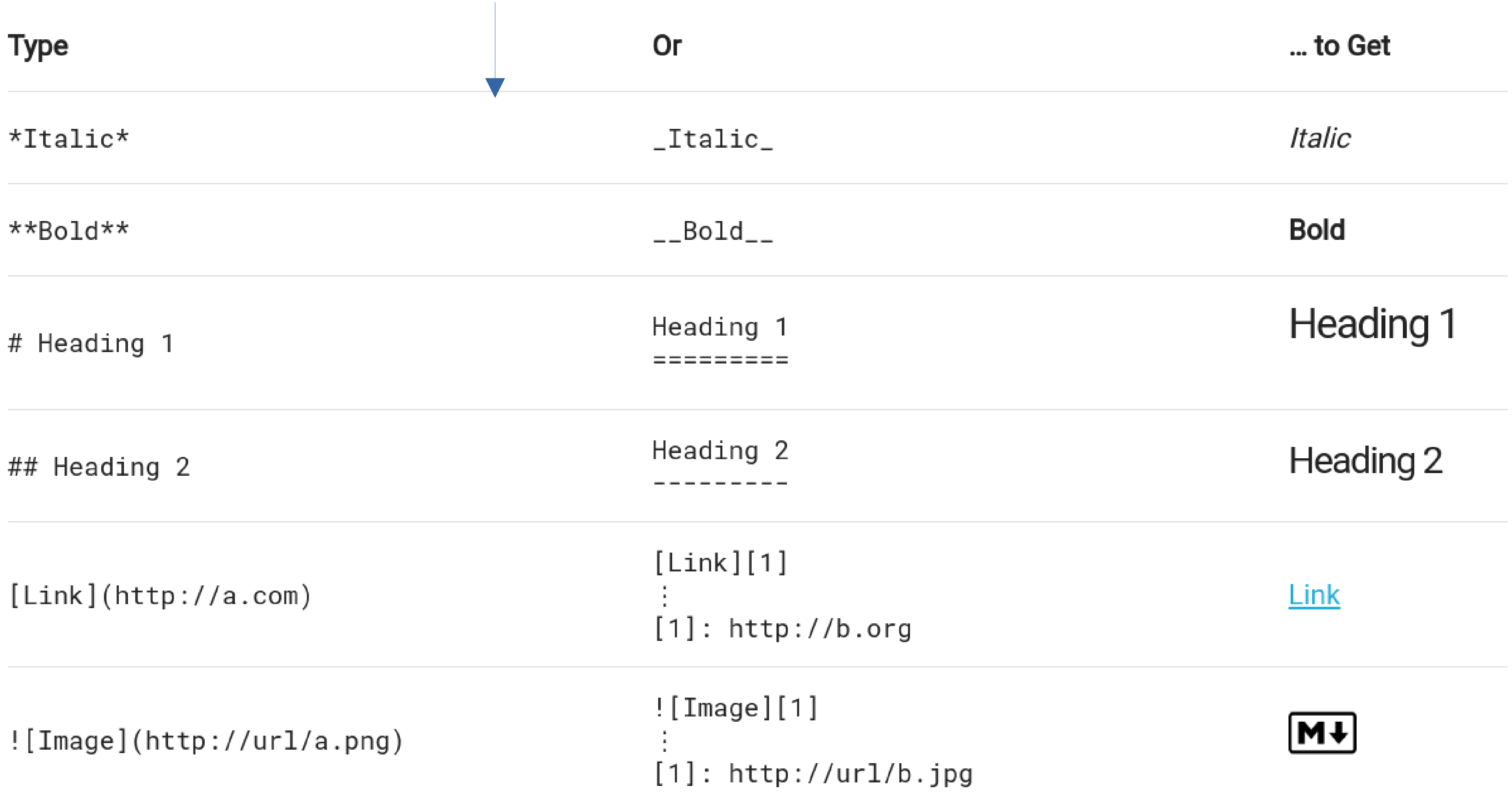
```
[1]: from tqdm.auto import tqdm
     import num

     import mat
     plt.rcPara
```

```
**Dimensionless units:** Next, let's use $\ell$ as the unit of distance and $\omega_c$
as the unit of energy, which gives

$H = -\dfrac{1}{2}\partial_y^2 + \dfrac{1}{2}\left(y-y_0\right)^2 + V(y)$

Notice that in this expression $y$ and $V(y)$ are now dimensionless.
```

```
[1]: from tqdm.auto import tqdm
     import numpy as np

     import matplotlib.pyplot as plt
     plt.rcParams.update({'font.size': 16})
```

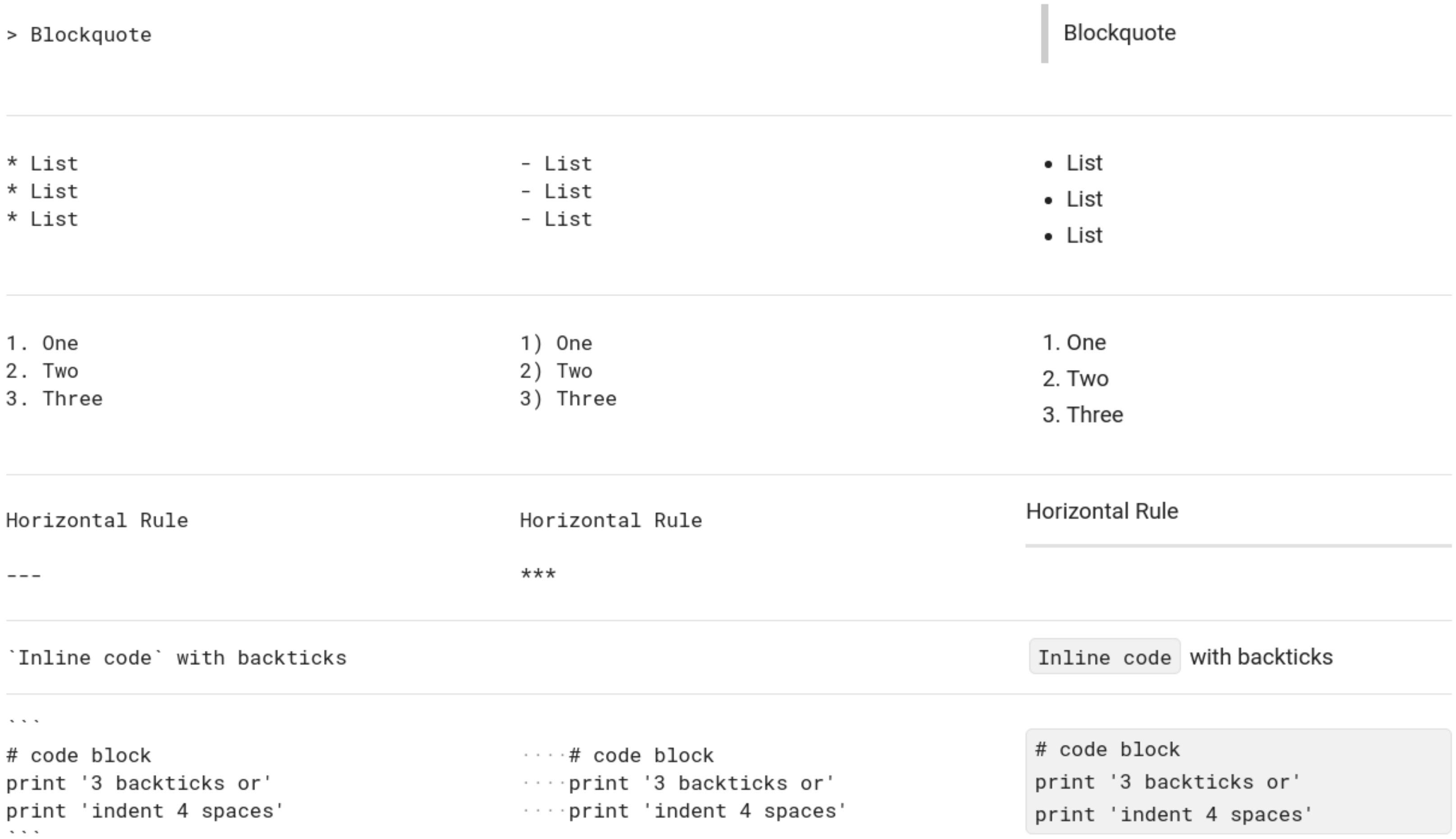
Markdown examples

Check these links: [https://commonmark.org/help/] [https://www.markdownguide.org/cheat-sheet/]



Type	Or	… to Get
Italic	_Italic_	*Italic*
Bold	__Bold__	**Bold**
# Heading 1	Heading 1 =========	Heading 1
## Heading 2	Heading 2 ---------	Heading 2
[Link](http://a.com)	[Link][1] ⋮ [1]: http://b.org	Link
![Image](http://url/a.png)	![Image][1] ⋮ [1]: http://url/b.jpg	M↓

Markdown examples



> Blockquote		Blockquote
* List * List * List	- List - List - List	• List • List • List
1. One 2. Two 3. Three	1) One 2) Two 3) Three	1. One 2. Two 3. Three
Horizontal Rule ---	Horizontal Rule ***	Horizontal Rule
`Inline code` with backticks		Inline code with backticks

```
# code block
print '3 backticks or'
print 'indent 4 spaces'
```
	····# code block
	····print '3 backticks or'
	····print 'indent 4 spaces'
		# code block
		print '3 backticks or'
		print 'indent 4 spaces'