

Contents

1	Introduction	1
1.1	Org-mode markup	1
1.2	Embed equations	1
1.3	Embed code and results	2
1.4	Embed tables	2
1.5	Use ipython magic to get inline figures	2
1.6	Store metadata in the ipynb	3
1.7	Use HTML for fancy markup	5

1 Introduction

I wrote this module to allow me to write lecture notes in org-mode, but export them to ipython notebooks for my students. It also makes it easier to share my work with people who don't use org-mode (although it is a one-way conversion of org to ipynb).

To use this, you have to require the ox-ipynd library. Then you can export this file with the key sequence `C-c C-e n o` to create the ipynb and open it.

In the following sections I will demonstrate a few features.

1.1 Org-mode markup

You should get all the regular markup:

- **bold text**
- *italics*
- underlined
- ~~strikethrough~~
- `verbatim`
- `code`
- superscripts, e.g. H^+
- subscripts, e.g. CH_4
- hyperlinks: <http://github.com/jkitchin/ox-ipynd>
- internal links do not work: [Org-mode markup](#)

1.2 Embed equations

You can use L^AT_EX equations like: $\int_0^1 \sin x dx$.

1.3 Embed code and results

Note that code blocks must be either "ipython" or "R", and the first one determines the language used for all of the code blocks.

```
1 import numpy as np
2 from scipy.integrate import quad
3
4 print(quad(lambda x: np.sin(x), 0, 1))
```

(0.45969769413186023, 5.103669643922839e-15)

Here we force a new cell to be created with the `ipynb-newcell` directive. `#+ipynb-newcell`
This text should be in its own cell.

1.4 Embed tables

You can create tables and they render nicely in the notebook. Note that you probably should not use the tables as data sources to code blocks because the ipython notebook does not support that.

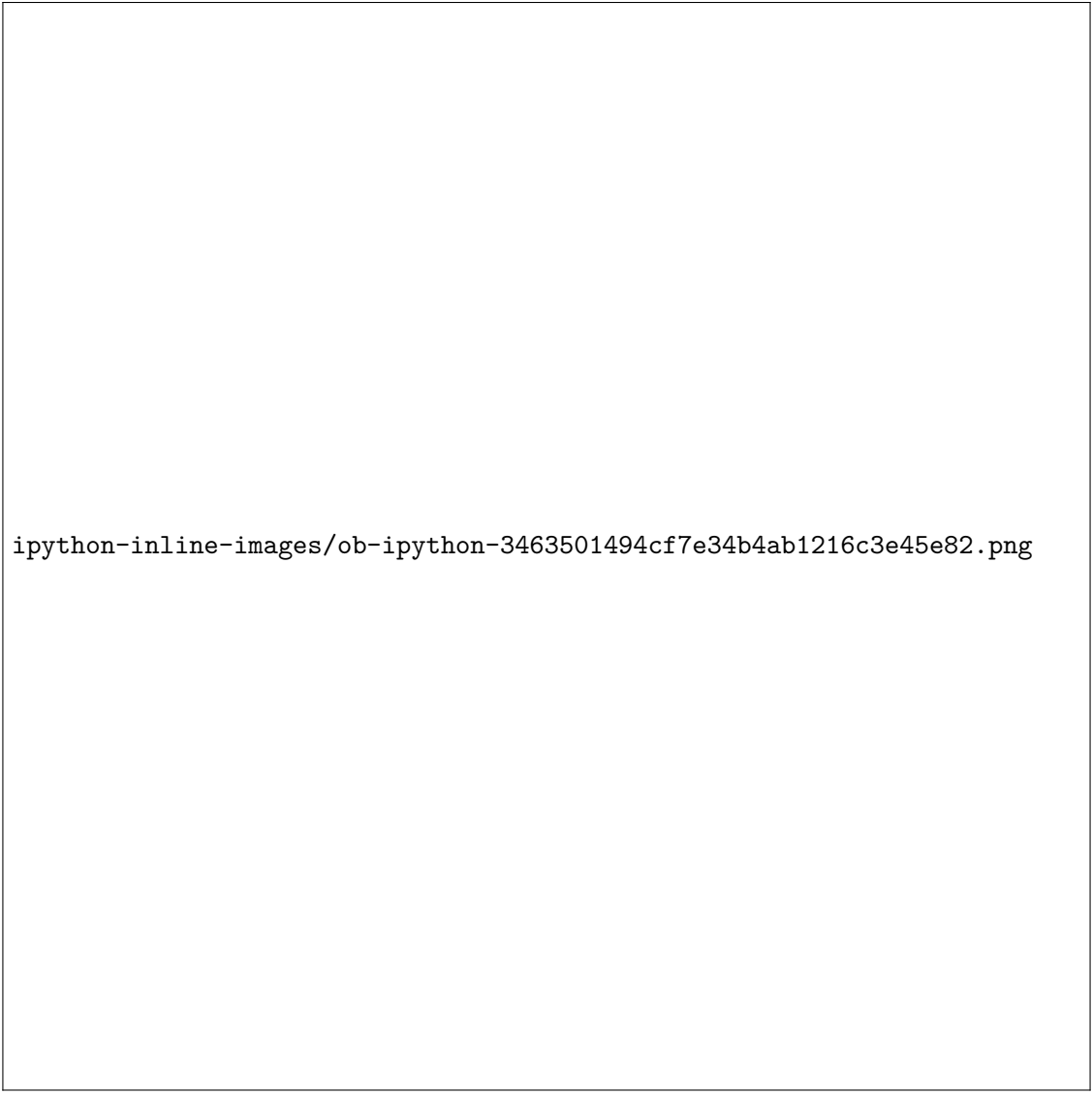
Table 1: A table of numbers.

x	y
1	1
2	4
3	9

1.5 Use ipython magic to get inline figures

```
1 %matplotlib inline
2 import matplotlib.pyplot as plt
3
4 z = np.linspace(0, 20 * np.pi, 500)
5 x = np.sin(z) * np.exp(-0.1 * z)
6 y = np.cos(z) * np.exp(-0.1 * z)
7 plt.plot(x, y)
```

[<matplotlib.lines.Line2D at 0x11a2ef6a0>] <matplotlib.figure.Figure at 0x11a22b588>




`ipython-inline-images/ob-ipython-3463501494cf7e34b4ab1216c3e45e82.png`

1.6 Store metadata in the ipynb


The directive `ox-ipython-keyword-metadata` lists file variables that should be saved as metadata in the ipynb.

We can set values as file variables like this:



`./screenshots/date-05-08-2017-time-08-55-41.png`

On export, you can see these are stored in the metadata as:



`./screenshots/date-05-08-2017-time-08-56-35.png`

These will show as bullets at the top of the ipynb. These are machine readable, so you can access them with any tool that can read json. You can use this to store author names, document id's, etc.

```
1 import json
2
3 with open("example.ipynb") as f:
4     d = json.loads(f.read())
5
6 print(d['metadata']['org'])
```

```
{'KEY1': 'value1', 'KEY2': 'value2'}
```

1.7 Use HTML for fancy markup

You can use html tags directly in the org file. Here are two examples.

<div class="alert alert-warning"> You can use an alert to highlight something you want to stand out. </div>

This text will show up as blue.