nbdime:
Notebook Diffing and Merging

Vidar Tonaas Fauske - @vidartf
Min Ragan-Kelley - @minrk
Martin Sandve Alnæs - @martinal

simula

jupyter

Outline

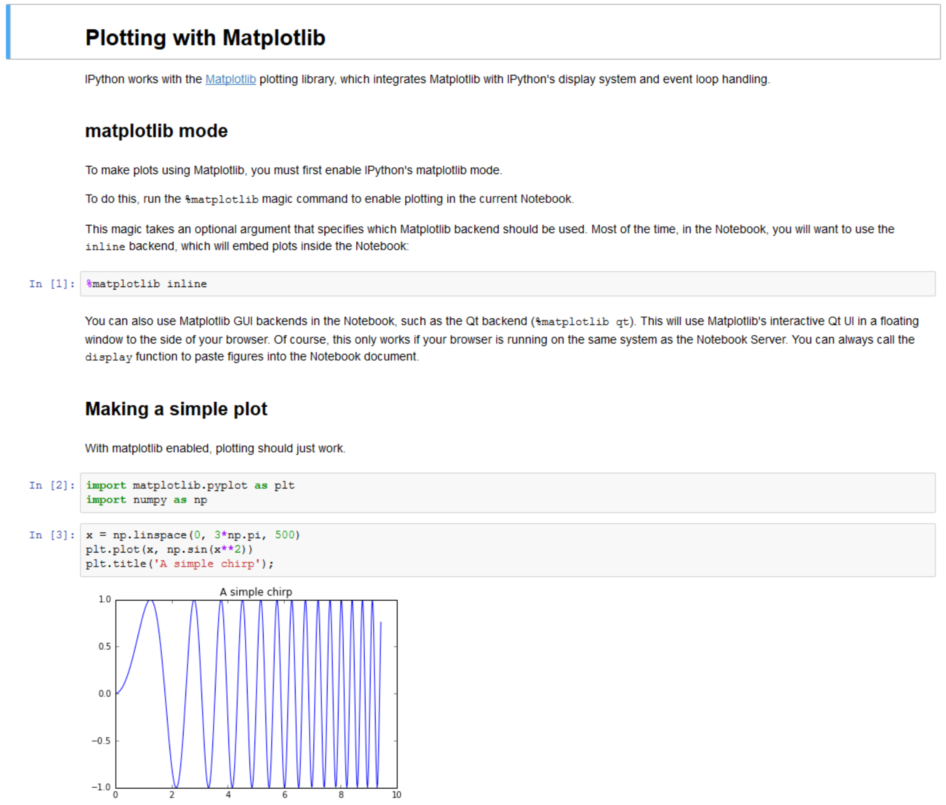
- Why we need custom diff/merge
- Nbdime command line interface (CLI)
- Nbdime web tools
- Interfacing nbdime with git

Notebooks in version control

- History for yourself
- Collaboration with several authors

Why do we need custom diff/merge?

Notebook format:



Plotting with Matplotlib

IPython works with the Matplotlib plotting library, which integrates Matplotlib with IPython's display system and event loop handling.

matplotlib mode

To make plots using Matplotlib, you must first enable IPython's matplotlib mode.

To do this, run the `%matplotlib` magic command to enable plotting in the current Notebook.

This magic takes an optional argument that specifies which Matplotlib backend should be used. Most of the time, in the Notebook, you will want to use the `inline` backend, which will embed plots inside the Notebook:

```
In [1]: %matplotlib inline
```

You can also use Matplotlib GUI backends in the Notebook, such as the Qt backend (`%matplotlib qt`). This will use Matplotlib's interactive Qt UI in a floating window to the side of your browser. Of course, this only works if your browser is running on the same system as the Notebook Server. You can always call the `display` function to paste figures into the Notebook document.

Making a simple plot

With matplotlib enabled, plotting should just work.

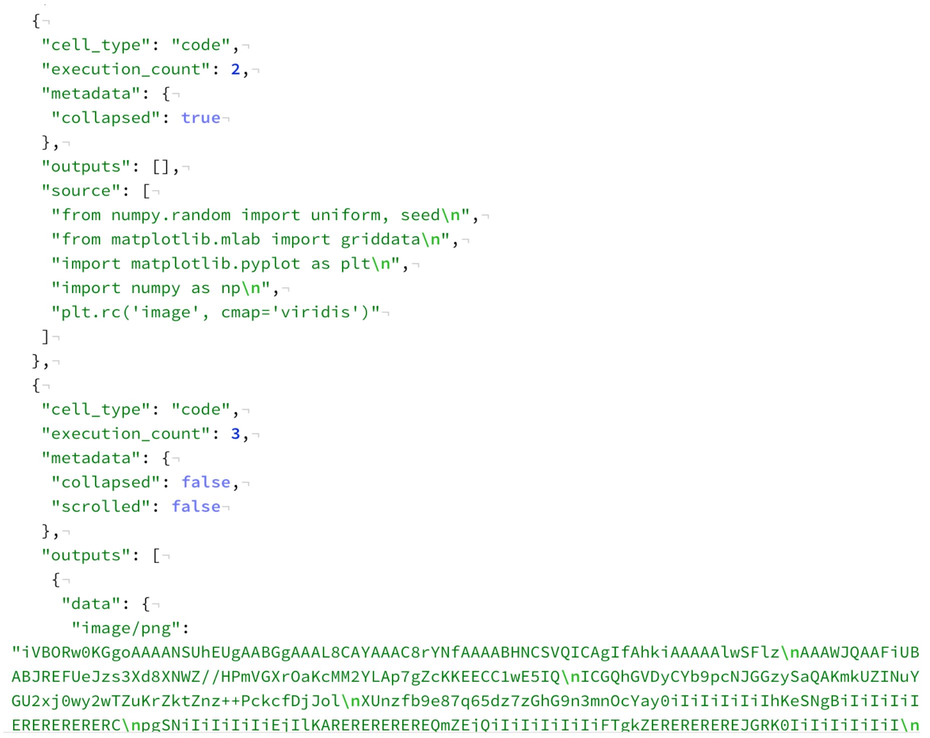
```
In [2]: import matplotlib.pyplot as plt
        import numpy as np
```

```
In [3]: x = np.linspace(0, 3*np.pi, 500)
        plt.plot(x, np.sin(x**2))
        plt.title('A simple chirp');
```

Why do we need custom diff/merge?

Notebook format:

- Lists
- Dictionaries
- Strings
- Atomic values: numbers, booleans, binary data (base64)



```
{
 "cell_type": "code",
 "execution_count": 2,
 "metadata": {
  "collapsed": true
 },
 "outputs": [],
 "source": [
  "from numpy.random import uniform, seed\n",
  "from matplotlib.mlab import griddata\n",
  "import matplotlib.pyplot as plt\n",
  "import numpy as np\n",
  "plt.rc('image', cmap='viridis')"
 ]
},
{
 "cell_type": "code",
 "execution_count": 3,
 "metadata": {
  "collapsed": false,
  "scrolled": false
 },
 "outputs": [
  {
   "data": {
    "image/png":
"iVBORw0KGgoAAAANSUhEUgAABGgAAAL8CAYAAAC8rYNfAAAABHNCSVQICAgIfAhkiAAAAAlwSFlz\nAAAWJQAAFiUB
ABJREFUeJzs3Xd8XNWZ//HPmVGXrOaKcMM2YLAp7gZcKKEECC1wE5IQ\nICGQhGVDyCYb9pcNJGGzySaQAKmkUZINuY
GU2xj0wy2wTZuKrZktZnz++PckcfDjJol\nXUnzfb9e87q65dz7zGhG9n3mnOcYay0iIiIiIiIhKeSNgBiIiIiI
ERERERERC\npgSNiIiIiIiEjIlKARERERERERERQmZEjQiIiIiIiIiFTgkZERERERERERJGRK0IiIiIiIiI\n
```

Why do we need custom diff/merge?

Standard algorithms:

- Sequence of strings (lines)

We have a lot of a priori information!

```
$ diff a.ipynb b.ipynb
76,77d75
<     "plt.rc('axes', grid=False)\n",
<     "plt.rc('axes', facecolor='white')\n",
90c88
<       "image/png": "iVBORw0KGgoAAAANSUhEUgAABLkAAAMQCAYAAADLj7dlAAAABHNCSVQICAgIfAhki
AAAAAlwSFlz\nAAAWJQAAFiUBSVIk8AAAIABJREFUeJzsvXeYZFd57b12h0maPNJII2lGOaCAkEBCFgozIxkBAp
lY\n1waDyDZg8MX+zMU2F4Mx1x8PwWAwxmBjg4yNi2BfQMa20iiAQFkIjXKWRtJIE3tSz3TXuX+8vV2n\nqyucv
N+9z/o9zzynprvq1D6nqqtqr1prbRNFEQghhBBCCCGEEEII8Zkh1wMghBBCCCGEEEIIISQv\nFLkIIYQQQgghhB
BCiPdQ5CKEEEIIIYQQQggh3kORixBCCCGEEEIIIYR4D0UuQgghhBBCCCGEEOI9\nFLkIIYQQQgghBBCiPdQ5CK
EEEIIIYQQQggh3kORixBCCCGEEEIIIYR4D0UuQgghhBBCCCGEEOI9\nFLkIIYQQQgghBBCiPdQ5CKEEEIIIYQQ
Qggh3kORixBCCCGEEEIIIYR4D0UuQgghhBBCCCGEEOI9\nFLkIIYQQQjzEGHOJMaZljPmo67EkZWq8D7keByGEE
ELChCIXIYQQQirDGPOmKaFj3BhzkMNx/H/G\nmG3GmP/pagwFEbkeQJUYY75gjNlijHmD67EQQgghRB8UuQghB
BSJe+DCDMjAH7L4TjeAmA+gLc5\nHEMRGNcDqJi3AVgI4DddD4QQQggh+qDIRQghhJBKMMacCuBMAFsg4sy7jTH
DjobzZwBuBvBxR/dP\nsvERADcC+LTrgRBCCCFEHxS5CCGEEFIVH4C4uP4SIlQcBOD1LgRfRf3
T7IRRdFf\nRlH0K1EUXe96LIQQQgjRB0UuQgghhJSOMWYpgP8BoAXg7wH8HcTN9Tsux0UIIYQQQsKBIhchhBBC\
nquBdAOYAuDyKoscBfBvALgBnGWOe73RkhBBCCCEkCChyEUIIIaRUjDEGUjIfQRxciKJoDMB3p65C\nNxchhBBC
CMkNRS5CCCGElM3FAA4HsAnAD2I/t5HFNxpjFuW5A2PMXGPMh4wxwxwdxpj\nPmGMmd/l+pcYY1rGm
I92/HzB1M8np/5/mDHmr40xjxpjdhtj7jbGfMwYMy92m30ONMT80xmw0xuyc\nus6njDFLeozV3veHpv5/lDHm88
aY9VO332yMuWnqfg4o4Lz8njHmp1P73P73WmMuXfq/g7Ls++p/c8x\nxrzfGHOlMeYZY8weY8xjxphvGmPOGHDbQ4w
xf2GMeWDqdk8Zy75vjDmzz20+NnXu3tLld48YY/ZO\nXT7KGPMPxpiHTjDETxphvT/38rVWde0IIIYRUD0UuQggh
hJTN70BcXJdGUbTP/jCAoB3\nZN35lEB2I4D/H8ApU/u8DGPBWD4Ajl3/2Ee+iqft4L4BfAng/gEUAbgfwKICTp+7nIWPMG5
Och06MMadNjfkz\nAE4H3PjU/hd03d/dxpg3d/dxpg3Zdn31P7PhJzrLwBYYDVk18xaIcPkmAfxL4BGtmAdzcYY/68x23PhZzb3wEwBuA
OAPMB\nvBrAtcaYV/e42wjdHzf7OxhjXghZGhAMYB/BTAlZ23L/PcE0IIIcQNFLkIIYQQUhrGmGMBrJ36\n79
e7XOXvIaLe3Lz1LczUcBnAhgHYBDoyg6LYqi0wA8Dy6rYSIMDOG1/mDKIriAsp3AFwH4JCplRhf\nAOBoiHB0JoC
vAfg0gK8CWDG16t8LAawxwL4kx5iNgAaAL4mi6JAqoil4SRdGpAA4JCpd\n8HEAswFcaoz5HwnzpHwnPhezc
```

Merging notebooks

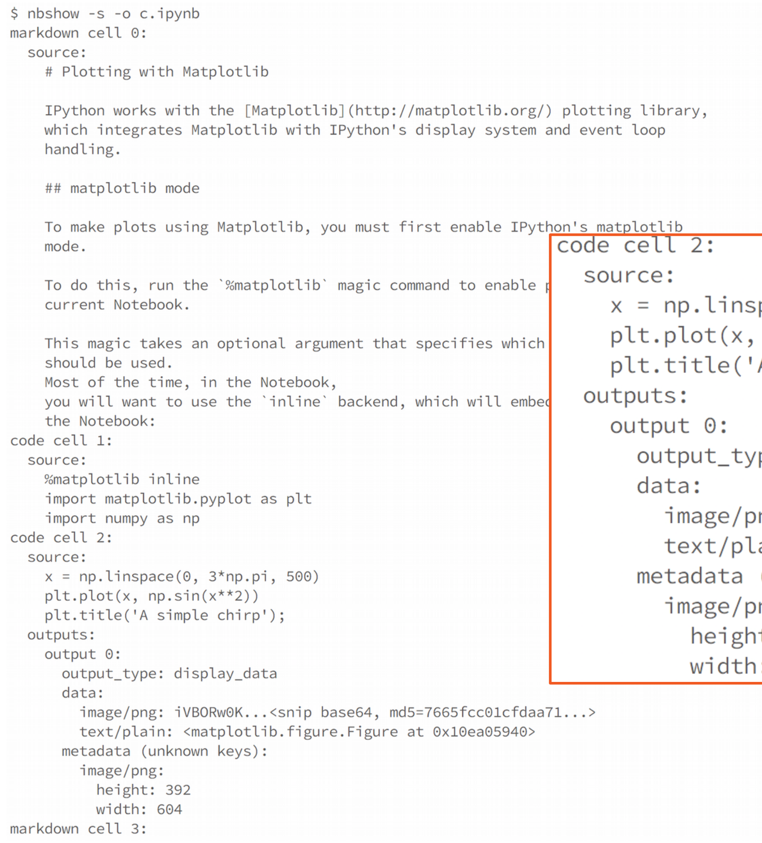
- Challenge: Hierarchical merge
- Major unit: Cell
- Input is more important than outputs!
- Input is “traditional” string merge
- Outputs are best treated as atomic
- Certain fields can typically be ignored

What can nbdime offer?

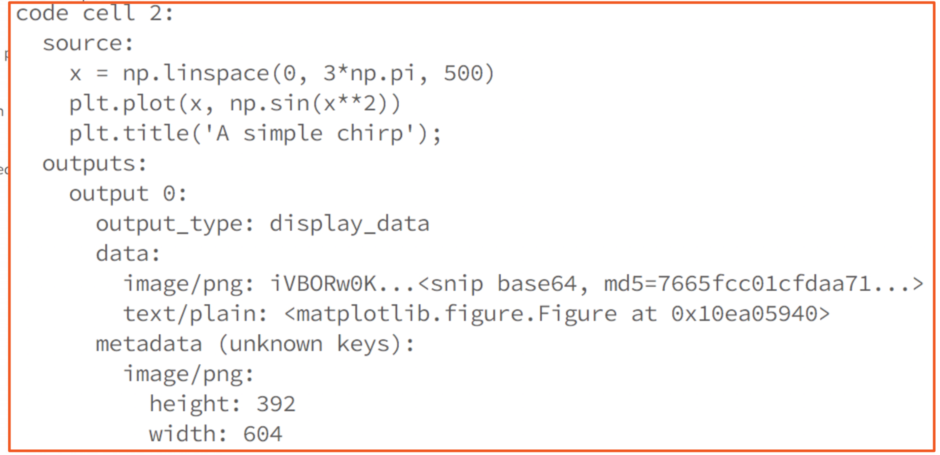
- Backend/library for diffing and merging notebooks
- CLI applications
  - Diff/merge
  - Nbshow
- Web applications (rich diff/merge view)
- git drivers (diff/merge) and mergetool (web)

Command Line Interface

```
> nbshow notebook1.ipynb
```



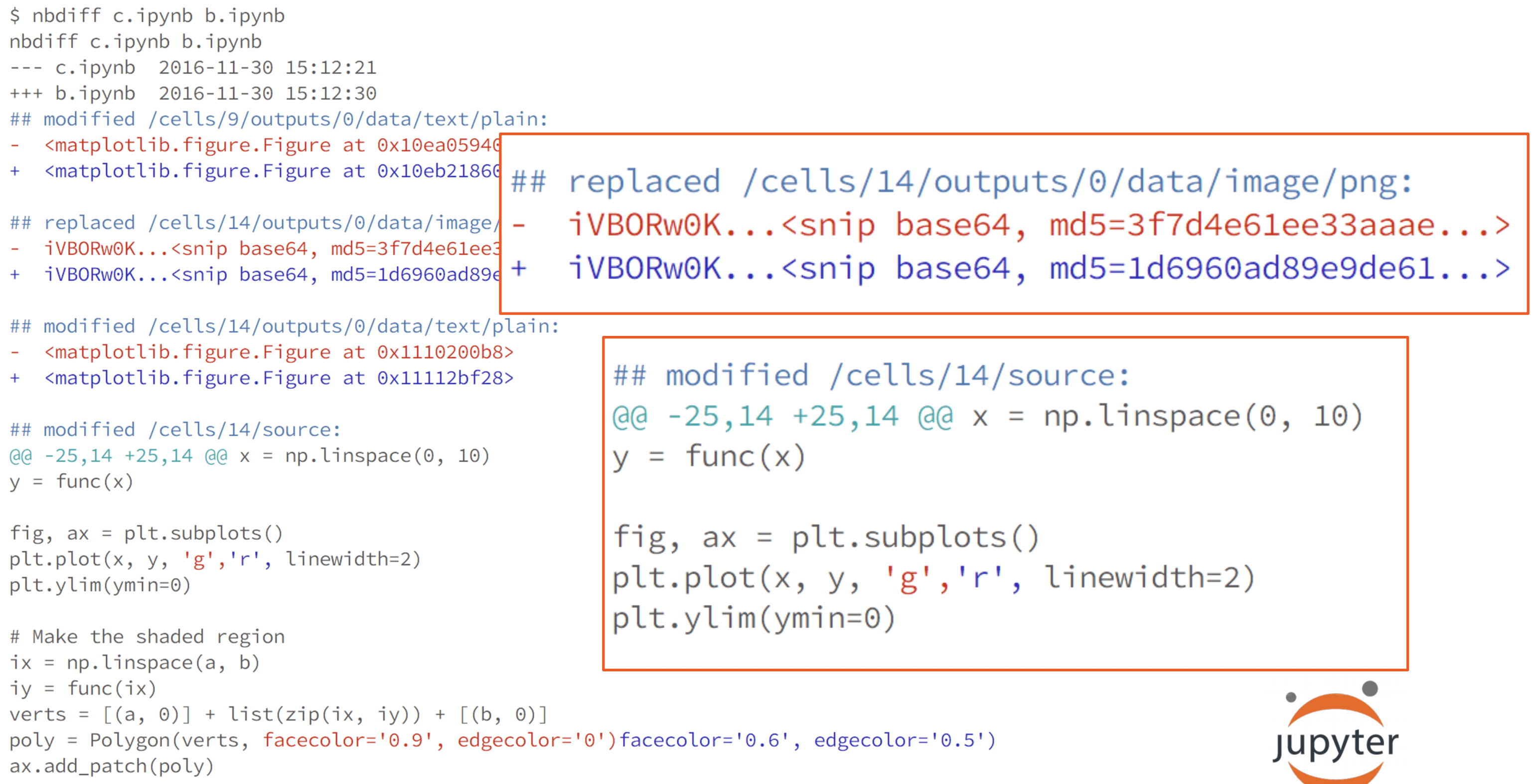
```
$ nbshow -s -o c.ipynb
markdown cell 0:
  source:
    # Plotting with Matplotlib

    IPython works with the [Matplotlib](http://matplotlib.org/) plotting library,
    which integrates Matplotlib with IPython's display system and event loop
    handling.

    ## matplotlib mode

    To make plots using Matplotlib, you must first enable IPython's matplotlib
    mode.

    To do this, run the `%matplotlib` magic command to enable p
    current Notebook.

    This magic takes an optional argument that specifies which
    should be used.
    Most of the time, in the Notebook,
    you will want to use the `inline` backend, which will embed
    the Notebook:
code cell 1:
  source:
    %matplotlib inline
    import matplotlib.pyplot as plt
    import numpy as np
code cell 2:
  source:
    x = np.linspace(0, 3*np.pi, 500)
    plt.plot(x, np.sin(x**2))
    plt.title('A simple chirp');
  outputs:
    output 0:
      output_type: display_data
      data:
        image/png: iVBORw0K...<snip base64, md5=7665fcc01cfdaa71...>
        text/plain: <matplotlib.figure.Figure at 0x10ea05940>
      metadata (unknown keys):
        image/png:
          height: 392
          width: 604
markdown cell 3:
```



```
code cell 2:
  source:
    x = np.linspace(0, 3*np.pi, 500)
    plt.plot(x, np.sin(x**2))
    plt.title('A simple chirp');
  outputs:
    output 0:
      output_type: display_data
      data:
        image/png: iVBORw0K...<snip base64, md5=7665fcc01cfdaa71...>
        text/plain: <matplotlib.figure.Figure at 0x10ea05940>
      metadata (unknown keys):
        image/png:
          height: 392
          width: 604
```

Command Line Interface

```
> nbdiff notebook1.ipynb notebook2.ipynb
```



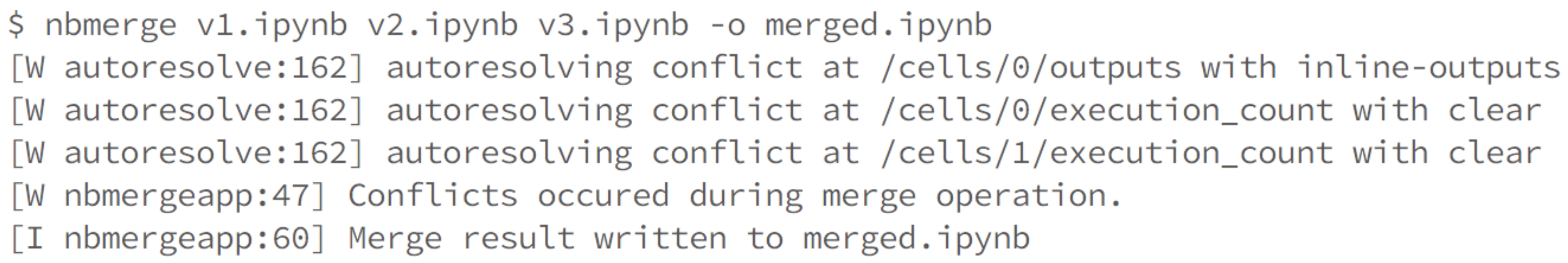
```
$ nbdiff c.ipynb b.ipynb
nbdiff c.ipynb b.ipynb
--- c.ipynb  2016-11-30 15:12:21
+++ b.ipynb  2016-11-30 15:12:30
## modified /cells/9/outputs/0/data/text/plain:
-  <matplotlib.figure.Figure at 0x10ea0594
+  <matplotlib.figure.Figure at 0x10eb2186

## replaced /cells/14/outputs/0/data/image/
-  iVBORw0K...<snip base64, md5=3f7d4e61ee3
+  iVBORw0K...<snip base64, md5=1d6960ad89e

## modified /cells/14/outputs/0/data/text/plain:
-  <matplotlib.figure.Figure at 0x1110200b8>
+  <matplotlib.figure.Figure at 0x11112bf28>

## modified /cells/14/source:
@@ -25,14 +25,14 @@ x = np.linspace(0, 10)
y = func(x)

fig, ax = plt.subplots()
plt.plot(x, y, 'g','r', linewidth=2)
plt.ylim(ymin=0)

# Make the shaded region
ix = np.linspace(a, b)
iy = func(ix)
verts = [(a, 0)] + list(zip(ix, iy)) + [(b, 0)]
poly = Polygon(verts, facecolor='0.9', edgecolor='0')facecolor='0.6', edgecolor='0.5')
ax.add_patch(poly)
```

```
## replaced /cells/14/outputs/0/data/image/png:
-  iVBORw0K...<snip base64, md5=3f7d4e61ee33aaae...>
+  iVBORw0K...<snip base64, md5=1d6960ad89e9de61...>
```

```
## modified /cells/14/source:
@@ -25,14 +25,14 @@ x = np.linspace(0, 10)
y = func(x)

fig, ax = plt.subplots()
plt.plot(x, y, 'g','r', linewidth=2)
plt.ylim(ymin=0)
```

Command Line Interface

```
> nbmerge base.ipynb local.ipynb remote.ipynb
```

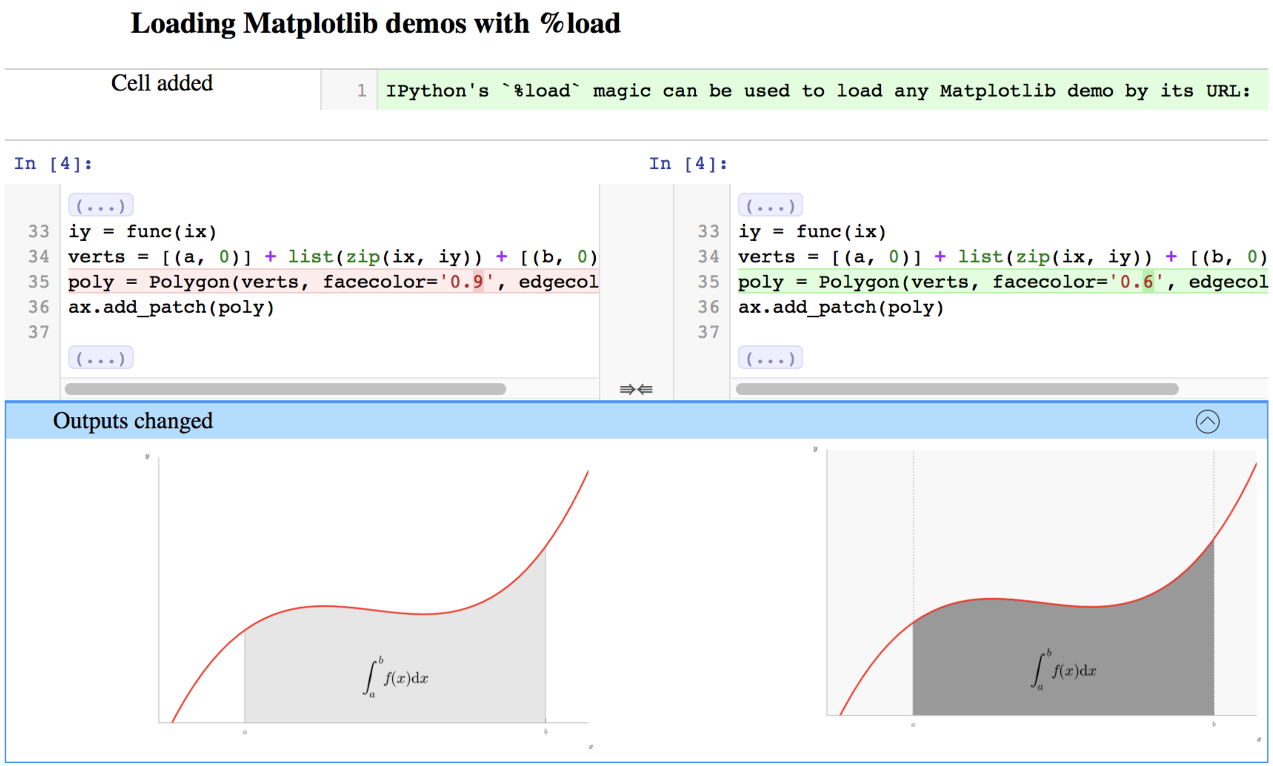


```
$ nbmerge v1.ipynb v2.ipynb v3.ipynb -o merged.ipynb
[W autoresolve:162] autoresolving conflict at /cells/0/outputs with inline-outputs
[W autoresolve:162] autoresolving conflict at /cells/0/execution_count with clear
[W autoresolve:162] autoresolving conflict at /cells/1/execution_count with clear
[W nbmergeapp:47] Conflicts occured during merge operation.
[I nbmergeapp:60] Merge result written to merged.ipynb
```

git mergedriver

- Associate ipynb files with our merger
- Fewer conflict than default merger

Web applications: Diff



Loading Matplotlib demos with %load

Cell added

```
IPython's `%load` magic can be used to load any Matplotlib demo by its URL:
```

In [4]:

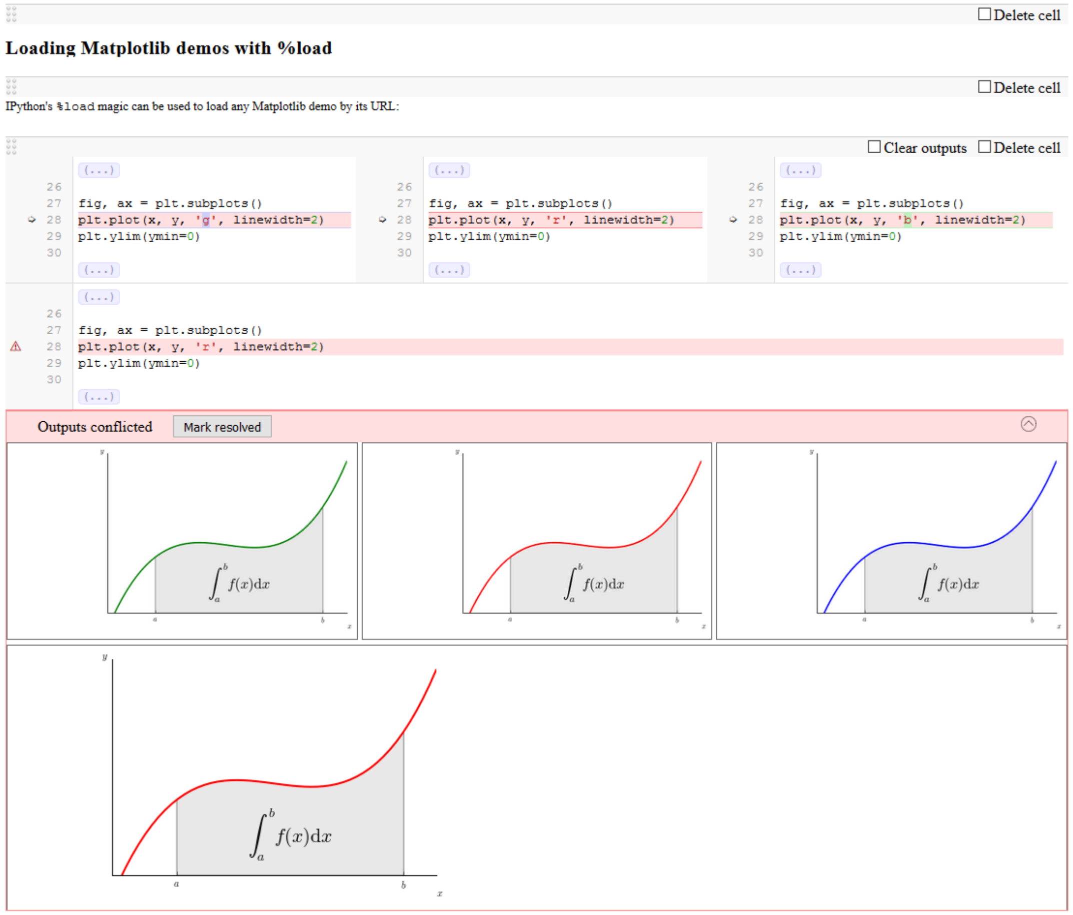
```
(...)
iy = func(ix)
verts = [(a, 0)] + list(zip(ix, iy)) + [(b, 0)
poly = Polygon(verts, facecolor='0.9', edgecol
ax.add_patch(poly)

(...)
```

In [4]:

```
(...)
iy = func(ix)
verts = [(a, 0)] + list(zip(ix, iy)) + [(b, 0)
poly = Polygon(verts, facecolor='0.6', edgecol
ax.add_patch(poly)

(...)
```

Outputs changed

Web applications: Merge



Loading Matplotlib demos with %load

IPython's `%load` magic can be used to load any Matplotlib demo by its URL:

Clear outputs
Delete cell

```
(...)

fig, ax = plt.subplots()
plt.plot(x, y, 'g', linewidth=2)
plt.ylim(ymin=0)

(...)
```

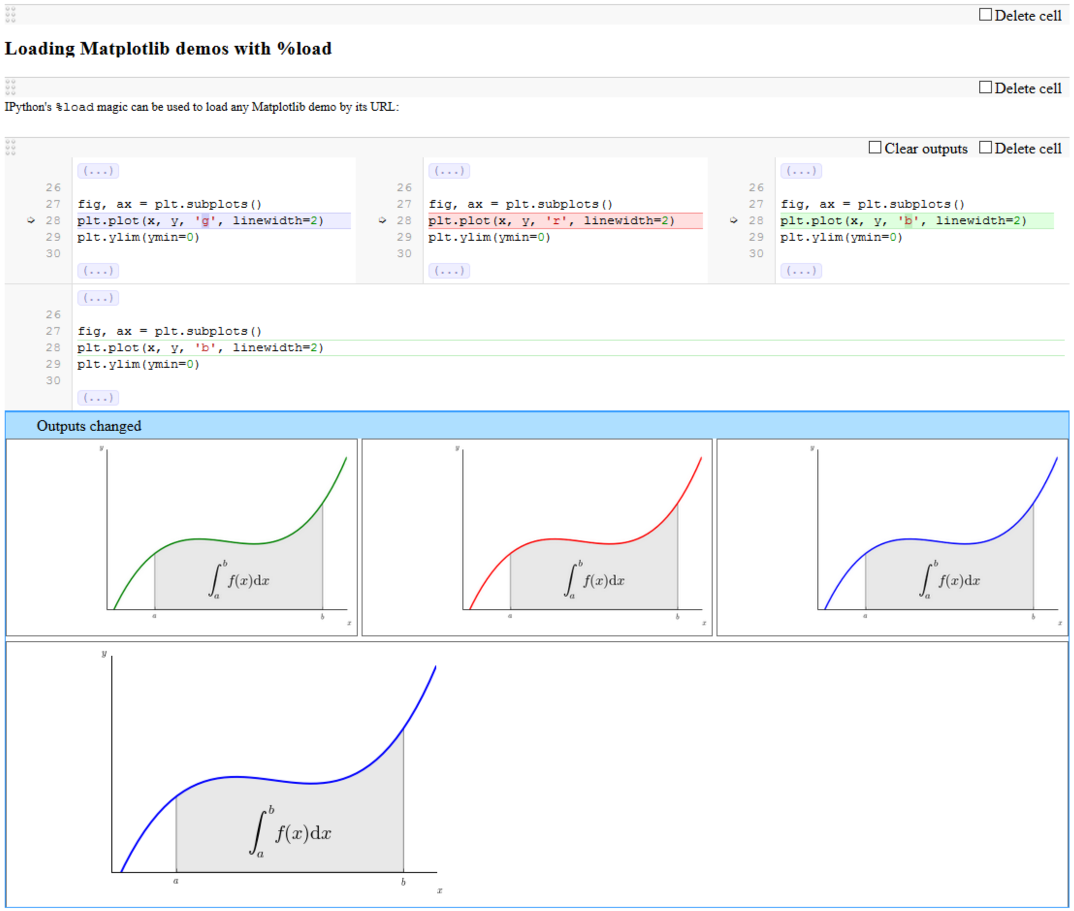
```
(...)

fig, ax = plt.subplots()
plt.plot(x, y, 'r', linewidth=2)
plt.ylim(ymin=0)

(...)
```

```
(...)

fig, ax = plt.subplots()
plt.plot(x, y, 'b', linewidth=2)
plt.ylim(ymin=0)

(...)
```

```
(...)

fig, ax = plt.subplots()
plt.plot(x, y, 'r', linewidth=2)
plt.ylim(ymin=0)

(...)
```

Outputs conflicted
Mark resolved
$\int_a^b f(x)\mathrm{d}x$
$\int_a^b f(x)\mathrm{d}x$
$\int_a^b f(x)\mathrm{d}x$
$\int_a^b f(x)\mathrm{d}x$

Web applications: Merge


☐Delete cell

Loading Matplotlib demos with %load

☐Delete cell

IPython's `%load` magic can be used to load any Matplotlib demo by its URL:

☐Clear outputs ☐Delete cell

```
(...)

fig, ax = plt.subplots()
plt.plot(x, y, 'g', linewidth=2)
plt.ylim(ymin=0)

(...)
```

```
(...)

fig, ax = plt.subplots()
plt.plot(x, y, 'r', linewidth=2)
plt.ylim(ymin=0)

(...)
```

```
(...)

fig, ax = plt.subplots()
plt.plot(x, y, 'b', linewidth=2)
plt.ylim(ymin=0)

(...)
```

```
(...)

fig, ax = plt.subplots()
plt.plot(x, y, 'b', linewidth=2)
plt.ylim(ymin=0)

(...)
```

Outputs changed

simula

jupyter

Demo

Getting it

- Release pre-release last night
  - Try it out, find the bugs
- Full release with a week or two

```
> pip install --pre nbdime
```

Summary

- Nbdime: Custom diff/merge because of structured format
  - Also allows us to make informed merges
- Integrates with git
- Rich rendering allows for better oversight

OPEN
DREAMKIT

jupyter

Thanks!

European
Commission

simula

jupyter