```
%md
## Introduction
In this tutorial we will go through some of the basic features of Zeppelin's built-in matplotlib integration.

### Prerequisites
`matplotlib` must be installed to your local python installation. (use `pip install matplotlib` or `conda install matplotlib` if you have `conda`). Additionally, you will need Zeppelin's matplotlib backe

### Interpreters
Most of the examples shown in this tutorial can be used interchangeably with either the `python` or `pyspark` interpreters. Iterative plotting using the Angular Display System is currently only available

### macOS
Make sure locale is set, to avoid `ValueError: unknown locale: UTF-8`

### virtualenv
In case you want to use virtualenv or conda env:
 - configure python interpreter property -> `absolute/path/to/venv/bin/python`
 - see *Working with Matplotlib in Virtual environments* in the [Matplotlib FAQ](http://matplotlib.org/faq/virtualenv_faq.html)

### A simple example
Let's start by making a very simple line plot:

<div class="markdown-body">
<h2>Introduction</h2>
<p>In this tutorial we will go through some of the basic features of Zeppelin&rsquo;s built-in matplotlib integration. </p>
<h3>Prerequisites</h3>
<p><code>matplotlib</code> must be installed to your local python installation. (use <code>pip install matplotlib</code> or <code>conda install matplotlib</code> if you have <code>conda</code>). Addition
<h3>Interpreters</h3>
<p>Most of the examples shown in this tutorial can be used interchangeably with either the <code>python</code> or <code>pyspark</code> interpreters. Iterative plotting using the Angular Display System is
<h3>macOS</h3>
<p>Make sure locale is set, to avoid <code>ValueError: unknown locale: UTF-8</code></p>
<h3>virtualenv</h3>
<p>In case you want to use virtualenv or conda env:<br/> - configure python interpreter property -&gt; <code>absolute/path/to/venv/bin/python</code><br/> - see <em>Working with Matplotlib in Virtual envi
<h3>A simple example</h3>
<p>Let&rsquo;s start by making a very simple line plot:</p>
</div>

%python
import matplotlib.pyplot as plt
plt.plot([1, 2, 3])

<div style='width:auto;height:auto'><img src=

%md
Notice how an explicit call to `show()` is not necessary. This is accomplished via a post-execute hook which tells Zeppelin to plot all currently open matplotlib figures after executing the rest of the p
### Plotting multiple figures
We can easily plot multiple figures at once too:

<p>Notice how an explicit call to <code>show()</code> is not necessary. This is accomplished via a post-execute hook which tells Zeppelin to plot all currently open matplotlib figures after executing the
<h3>Plotting multiple figures</h3>
<p>We can easily plot multiple figures at once too:</p>

%python
# Figure 1
plt.plot([1, 2, 3])

# Figure 2
plt.figure()
plt.plot([3, 2, 1])

<div style='width:auto;height:auto'><img src=
<div style='width:auto;height:auto'><img src=ÜFuVRg

%md
### Changing the default inline plotting behavior
Both the `python` and `pyspark` include a built-in function for changing some default inline plotting behavior. For example, we can change the default size of each figure in pixels to 400x300 in svg form

<h3>Changing the default inline plotting behavior</h3>
<p>Both the <code>python</code> and <code>pyspark</code> include a built-in function for changing some default inline plotting behavior. For example, we can change the default size of each figure in pixe

%python
z.configure_mpl(width=400, height=300, fmt='svg')
plt.plot([1, 2, 3])

<div style='width:auto;height:auto'><img src=

%md
### Iteratively updating a plot
#### (a) Using multiple plots
Now let's show an example where we update each element of the plot in a separate paragraph. However, you may have noticed that each matplotlib figure instance gets closed immediately after its shown. To

<h3>Iteratively updating a plot</h3>
<h4>(a) Using multiple plots</h4>
<p>Now let's show an example where we update each element of the plot in a separate paragraph. However, you may have noticed that each matplotlib figure instance gets closed immediately after its shown.

%python
plt.close() # Added here to reset the first plot when rerunning the paragraph
z.configure_mpl(width=600, height=400, fmt='png', close=False)
plt.plot([1, 2, 3], label=r'$y=x$')

<div style='width:auto;height:auto'><img src=

%python
plt.plot([3, 2, 1], label=r'$y=3-x$')

<div style='width:auto;height:auto'><img src=

%python
plt.xlabel(r'$x$', fontsize=20)
plt.ylabel(r'$y$', fontsize=20)

<div style='width:auto;height:auto'><img src=

%python
plt.legend(loc='upper center', fontsize=20)

<div style='width:auto;height:auto'><img src=

%python
plt.title('Inline plotting example', fontsize=20)

<div style='width:auto;height:auto'><img src=

%md
####(b) Using a single plot
To iteratively update a single plot, we can leverage Zeppelin's built-in Angular Display System. Currently this feature is only available for the `pyspark` interpreter for raster (png and jpg) formats. T

<h4>(b) Using a single plot</h4>
<p>To iteratively update a single plot, we can leverage Zeppelin's built-in Angular Display System. Currently this feature is only available for the <code>pyspark</code> interpreter for raster (png and j

%pyspark
import matplotlib.pyplot as plt
plt.close() # Added here to reset the plot when rerunning the paragraph
z.configure_mpl(angular=True, close=False)
plt.plot([1, 2, 3], label=r'$y=x$')

<div style='width:auto;height:auto'><img src=

%pyspark
plt.plot([3, 2, 1], label=r'$y=3-x$')

%pyspark
plt.xlabel(r'$x$', fontsize=20)
plt.ylabel(r'$y$', fontsize=20)

%pyspark
plt.legend(loc='upper center', fontsize=20)

%pyspark
plt.title('Inline plotting example', fontsize=20)
```