Matplotlib

The Dongle Problem

MATLAB Graphics API

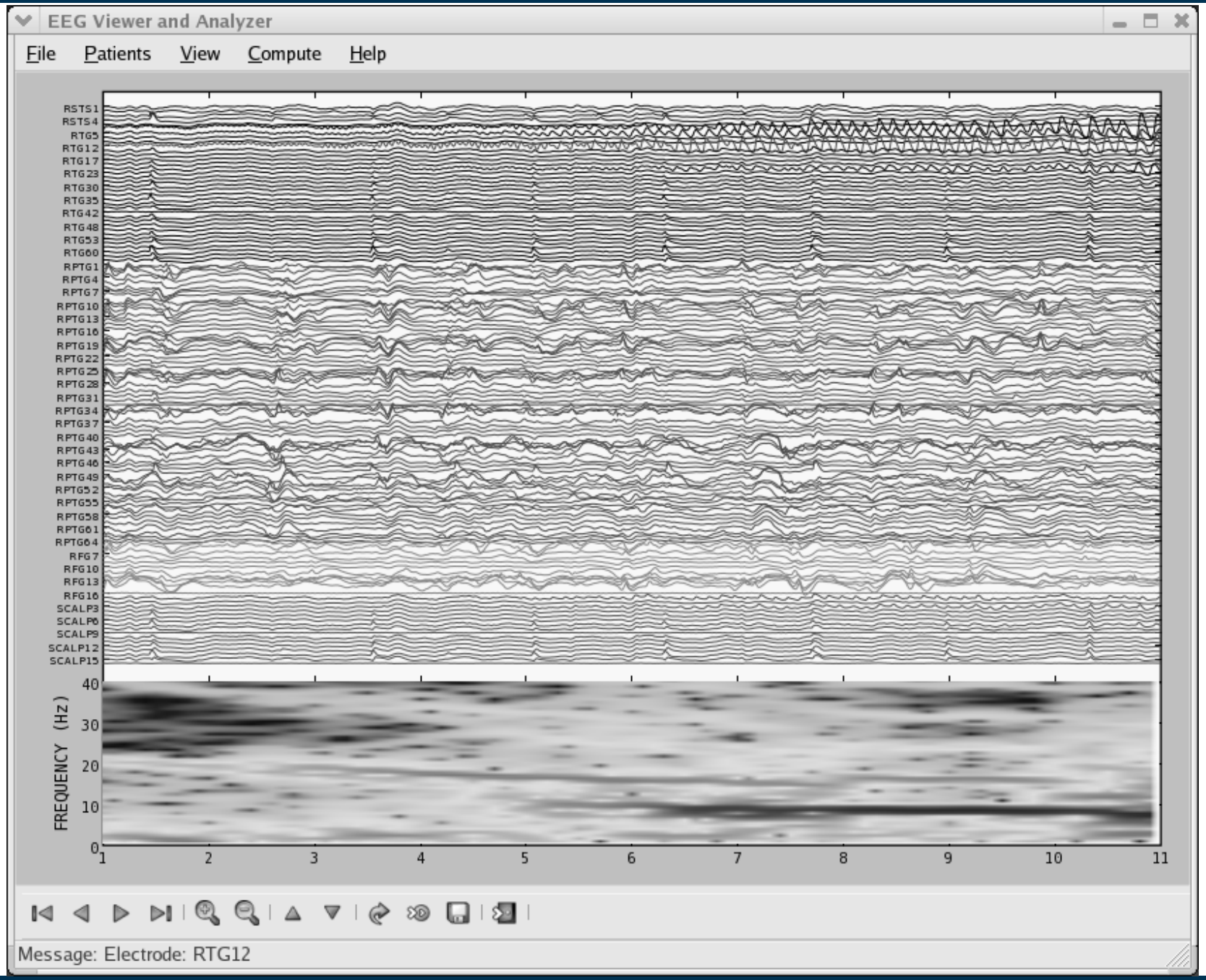
Simple, convenient plot commands: `plot()`

Arguments for simple customization: `'r--'`

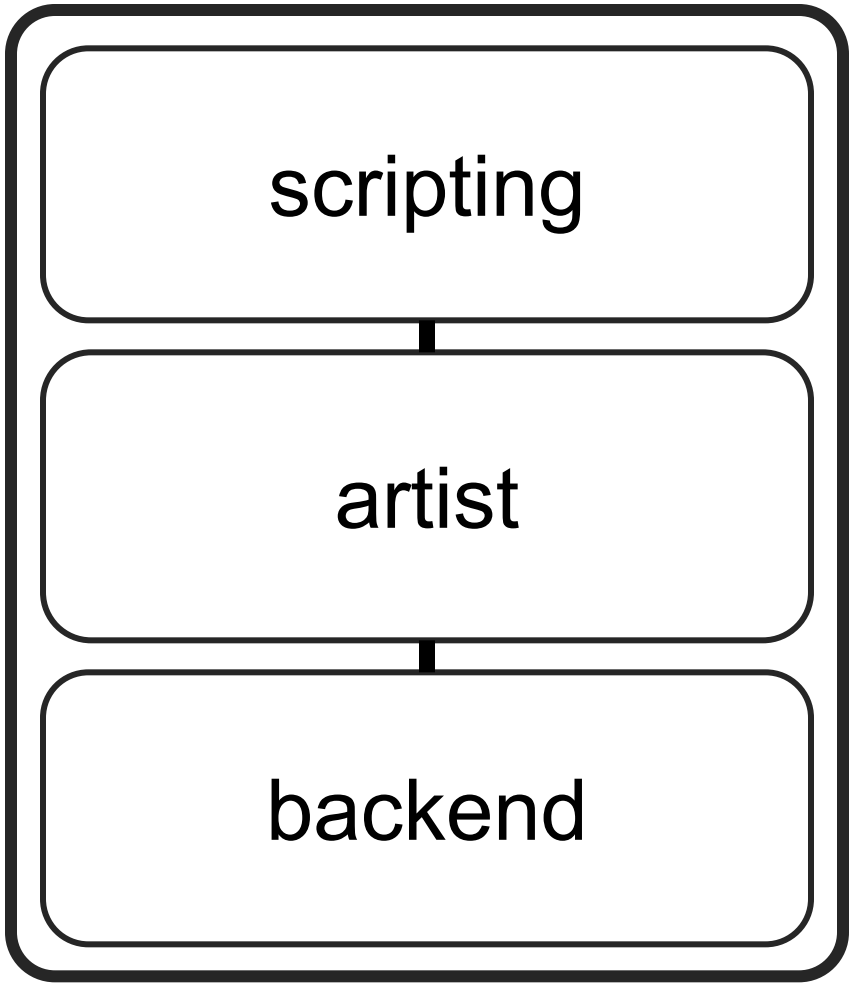
Rich, structured API to access lower-level functionality (axes, tick generators, etc.)

Great!

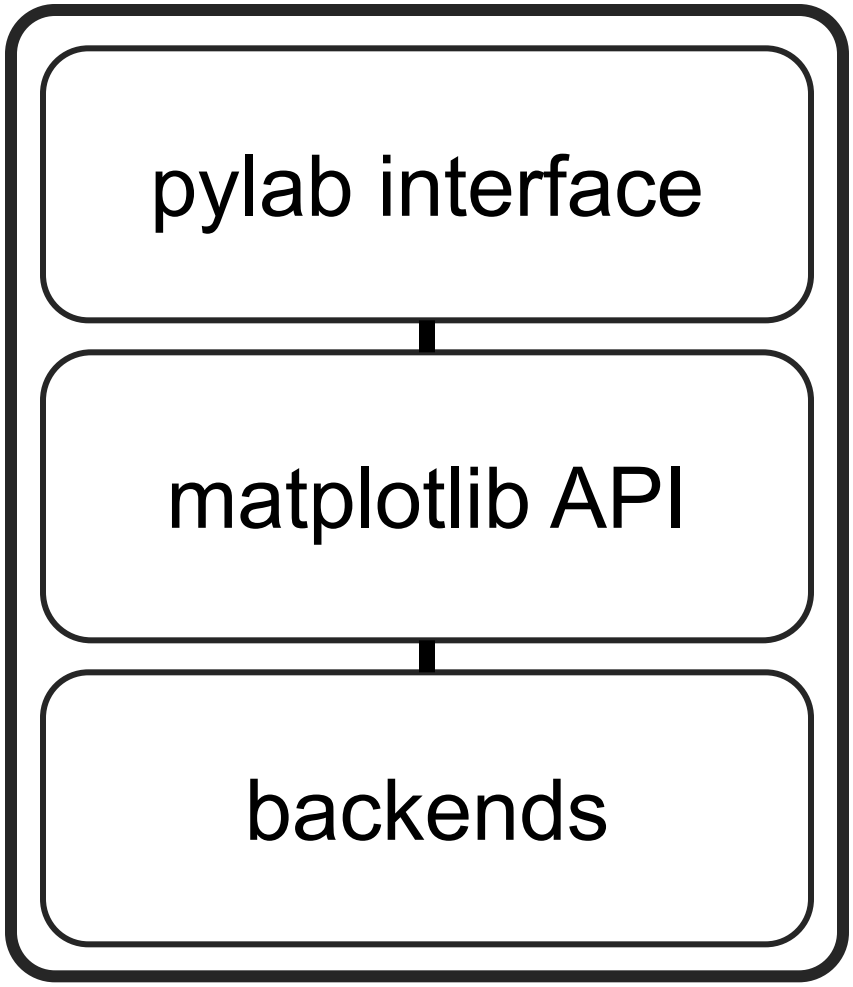
MATrix LABoratory for large applications…


EEG Viewer and Analyzer
File
Patients
View
Compute
Help
RSTS1
RSTS4
RTG5
RTG12
RTG17
RTG23
RTG30
RTG35
RTG42
RTG48
RTG53
RTG60
RPTG1
RPTG4
RPTG7
RPTG10
RPTG13
RPTG16
RPTG19
RPTG22
RPTG25
RPTG28
RPTG31
RPTG34
RPTG37
RPTG40
RPTG43
RPTG46
RPTG49
RPTG52
RPTG55
RPTG58
RPTG61
RPTG64
RFG7
RFG10
RFG13
RFG16
SCALP3
SCALP6
SCALP9
SCALP12
SCALP15
FREQUENCY (Hz)
40
30
20
10
0
1
2
3
4
5
6
7
8
9
10
11
Message: Electrode: RTG12

Matplotlib: Today



Matplotlib: Today



Backends

PS
SVG
AGG
GTK
GTKAgg
PDF
wxWidgets
…etc.

Selecting a Backend

Depends on your OS…
(Default: AGG)

In [1]:

```
import matplotlib
matplotlib.use('PDF')
import matplotlib.pyplot as plt
import numpy as np
```

* Select before loading pyplot to have effect

Inside Jupyter…

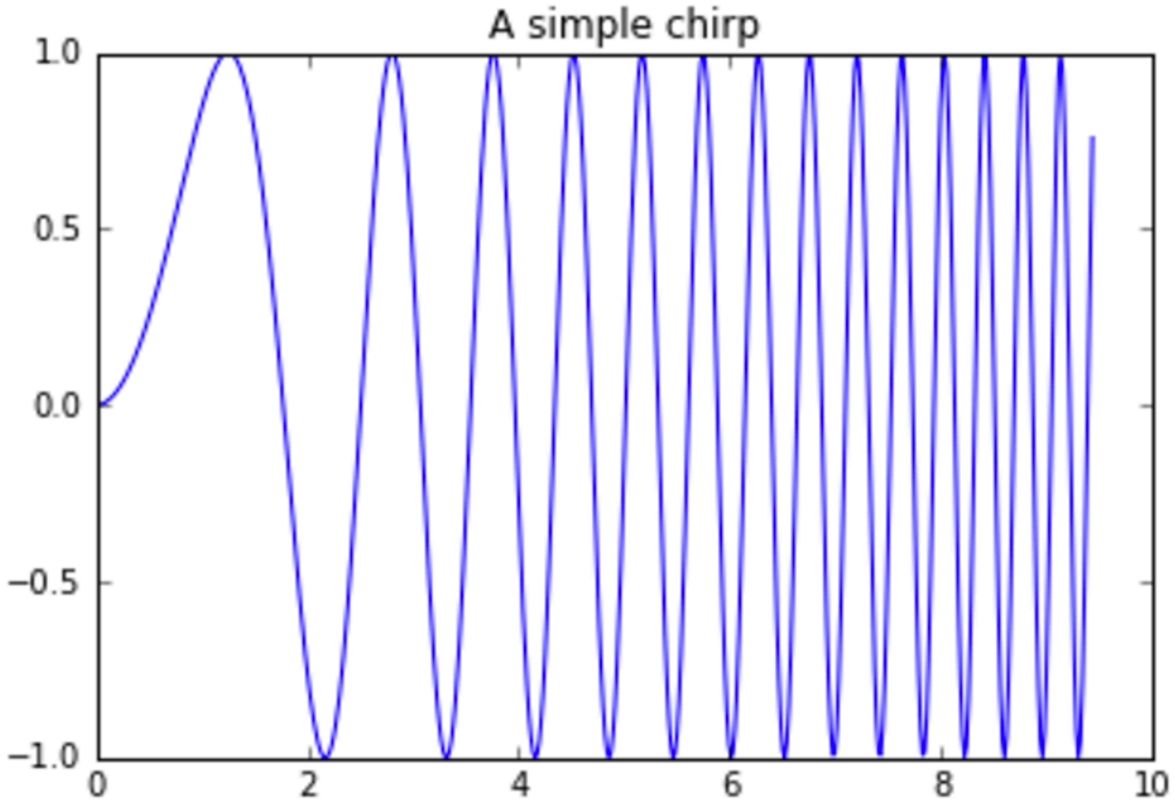
There's a twist:

In [1]:

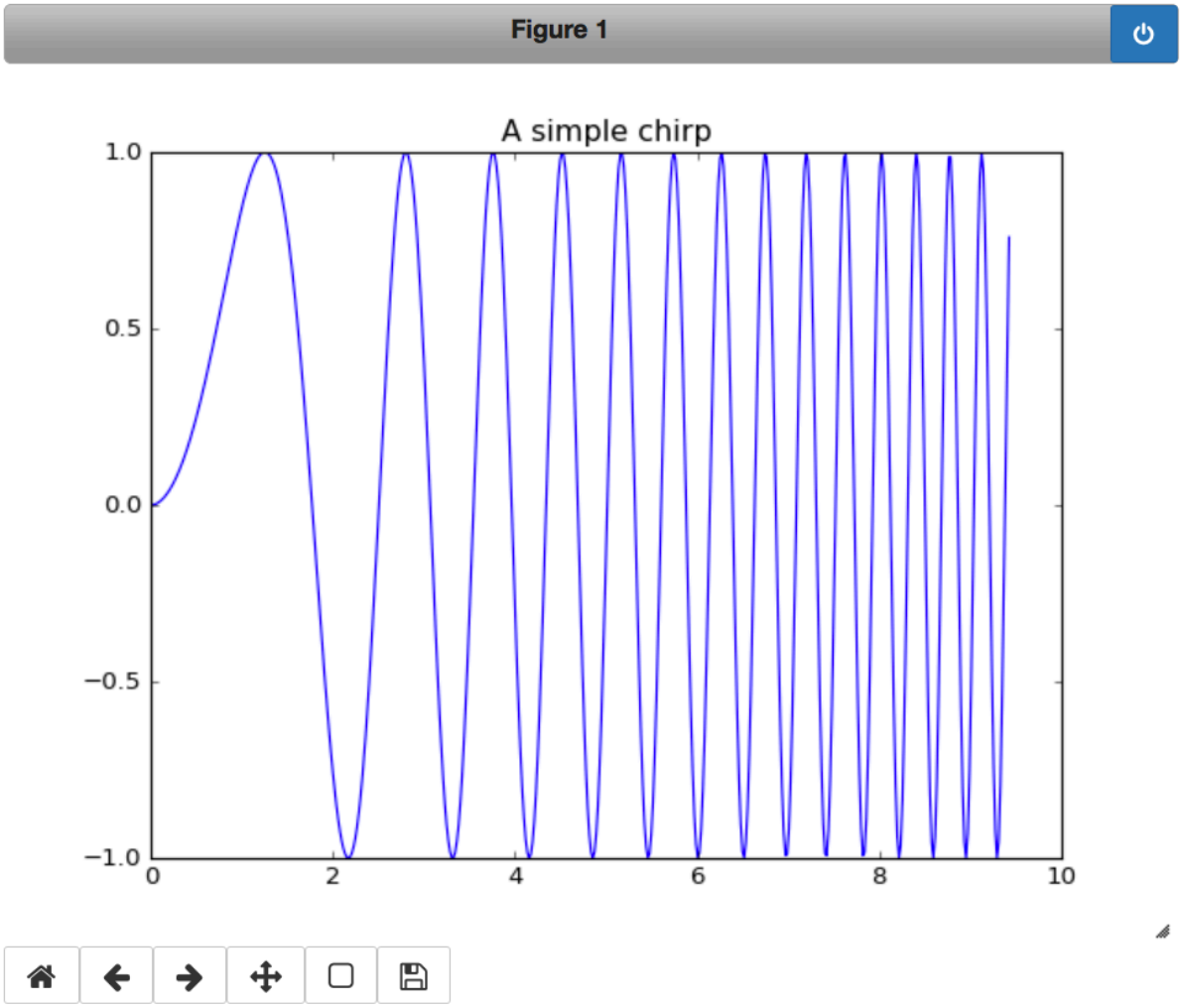
```
import matplotlib
# choose one or the other
%matplotlib inline
%matplotlib notebook
import matplotlib.pyplot as plt
import numpy as np
```

Inline

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



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



* Works only if notebook run locally, uses Qt

Wrapping It Back to Jupyter

Matplotlib API (Artist)

Three pieces!

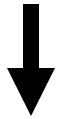
Matplotlib API (Artist)

`matplotlib.backend_bases.FigureCanvas`
area onto which the figure is drawn

Matplotlib API (Artist)

`matplotlib.backend_bases.FigureCanvas`

area onto which the figure is drawn

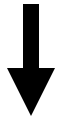


`matplotlib.backend_bases.Renderer`

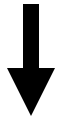
knows how to draw on the FigureCanvas

Matplotlib API (Artist)

`matplotlib.backend_bases.FigureCanvas`
area onto which the figure is drawn



`matplotlib.backend_bases.Renderer`
knows how to draw on the FigureCanvas



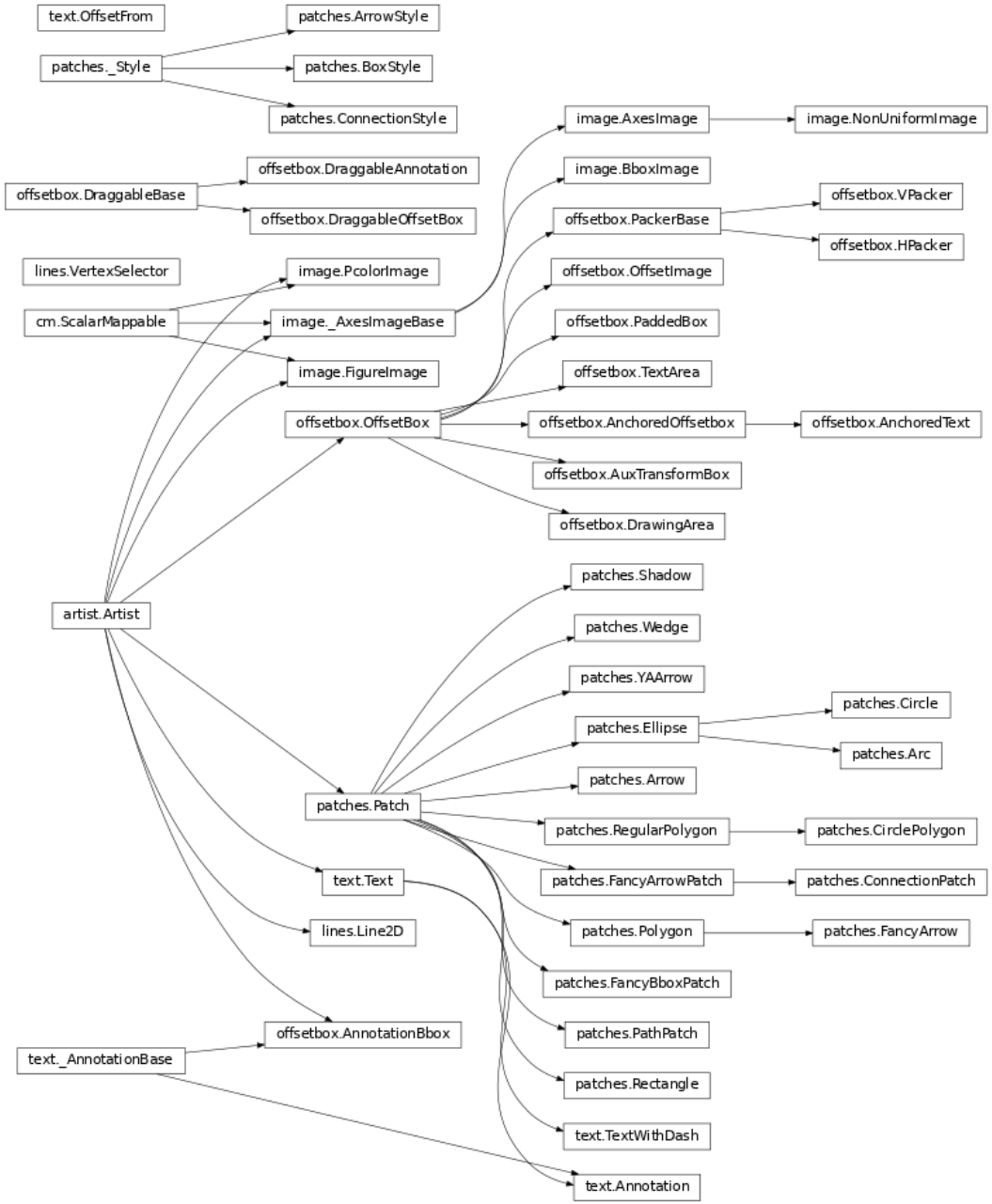
`matplotlib.artist.Artist`
knows how to use a renderer to paint onto the canvas

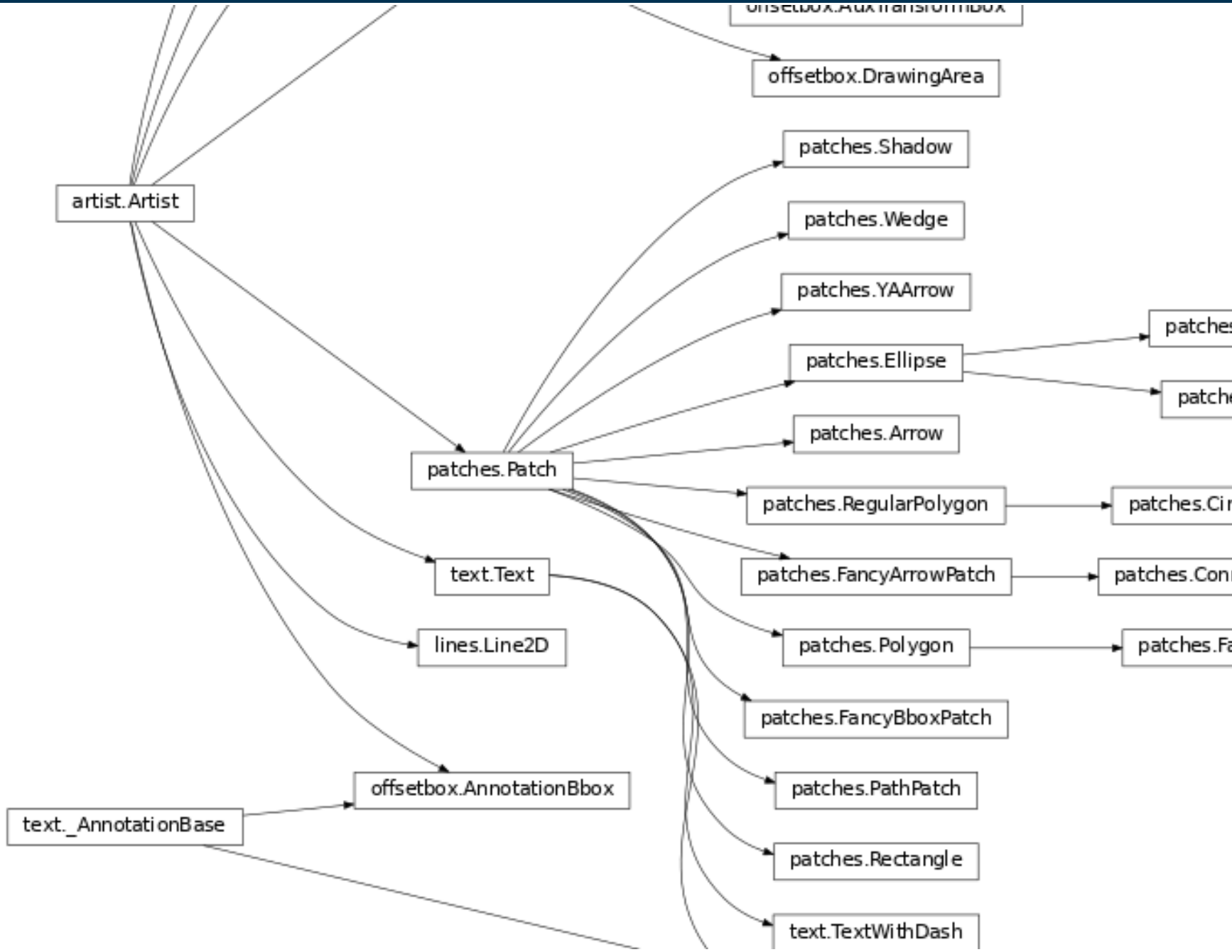
Matplotlib API (Artist)

`matplotlib.backend_bases.FigureCanvas`
area onto which the figure is drawn

`matplotlib.backend_bases.Renderer`
knows how to draw on the FigureCanvas

`matplotlib.artist.Artist`
knows how to use a renderer to paint onto the canvas


text.OffsetFrom
patches.ArrowStyle
patches._Style
patches.BoxStyle
patches.ConnectionStyle
image.AxesImage
image.NonUniformImage
offsetbox.DraggableAnnotation
offsetbox.DraggableBase
offsetbox.DraggableOffsetBox
image.BboxImage
offsetbox.PackerBase
offsetbox.VPacker
offsetbox.HPacker
lines.VertexSelector
image.PcolorImage
offsetbox.OffsetImage
cm.ScalarMappable
image._AxesImageBase
offsetbox.PaddedBox
image.FigureImage
offsetbox.TextArea
offsetbox.OffsetBox
offsetbox.AnchoredOffsetbox
offsetbox.AnchoredText
offsetbox.AuxTransformBox
offsetbox.DrawingArea
patches.Shadow
artist.Artist
patches.Wedge
patches.YAArrow
patches.Circle
patches.Ellipse
patches.Arc
patches.Arrow
patches.Patch
patches.RegularPolygon
patches.CirclePolygon
patches.FancyArrowPatch
patches.ConnectionPatch
text.Text
patches.Polygon
patches.FancyArrow
lines.Line2D
patches.FancyBboxPatch
patches.PathPatch
offsetbox.AnnotationBbox
text._AnnotationBase
patches.Rectangle
text.TextWithDash
text.Annotation

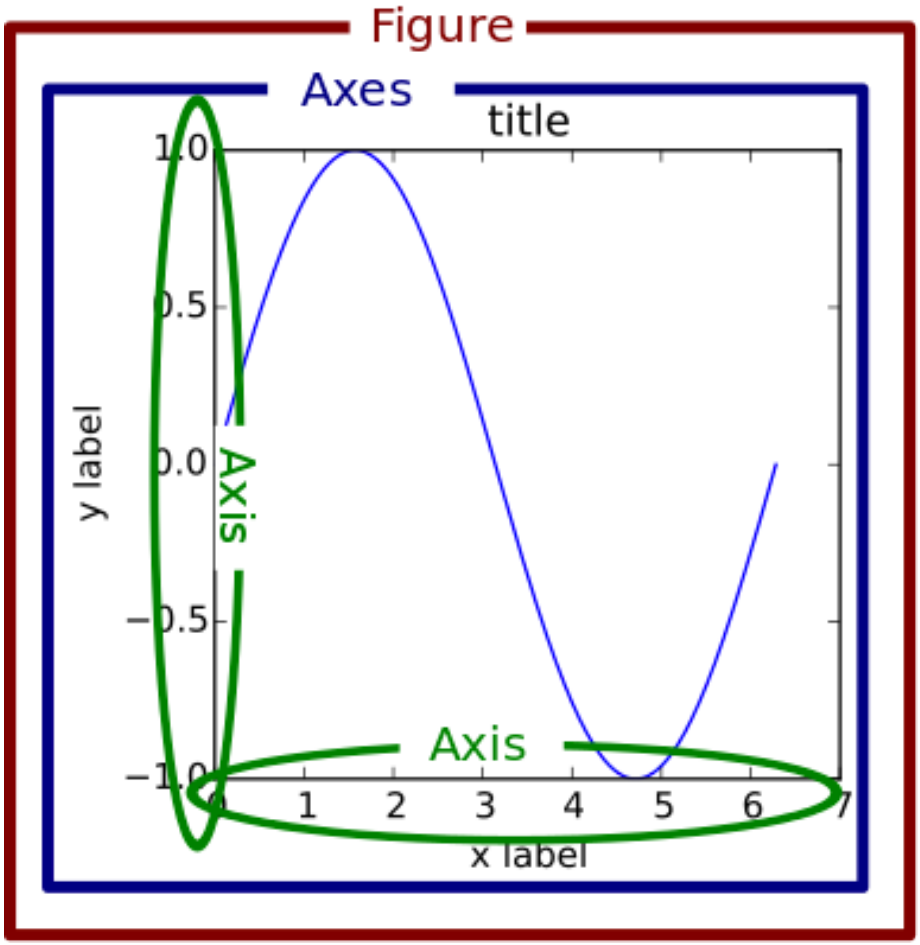

offsetbox.DrawingArea
artist.Artist
patches.Shadow
patches.Wedge
patches.YAArrow
patches.Ellipse
patches.Arrow
patches.Patch
patches.RegularPolygon
text.Text
patches.FancyArrowPatch
lines.Line2D
patches.Polygon
patches.FancyBboxPatch
offsetbox.AnnotationBbox
patches.PathPatch
text._AnnotationBase
patches.Rectangle
text.TextWithDash

Interacting With the Artist

This is up at the scripting (pyplot) level:

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

Where the Artists Live



Where the Artists Live

```
x = np.linspace(0, 3*np.pi, 500)
line = plt.plot(x, np.sin(x**2))
title = plt.title('A simple chirp')
```

Where the Artists Live

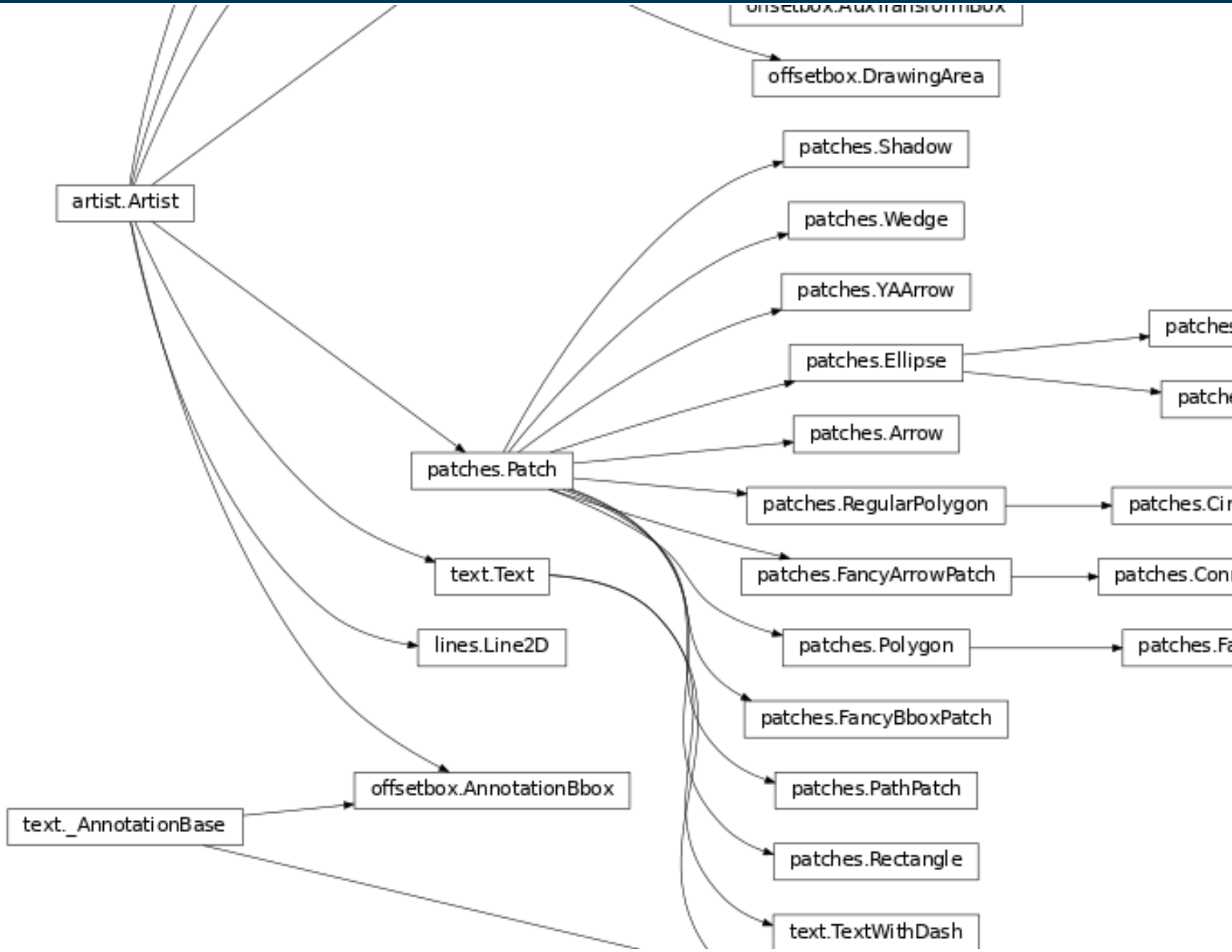
```
x = np.linspace(0, 3*np.pi, 500)
line = plt.plot(x, np.sin(x**2))
title = plt.title('A simple chirp')
```

In [5]:

```
title
```

Out[5]:

```
<matplotlib.text.Text at 0x
110650eb8>
```


offsetbox.DrawingArea
artist.Artist
patches.Shadow
patches.Wedge
patches.YAArrow
patches.Ellipse
patches.Arrow
patches.Patch
patches.RegularPolygon
text.Text
patches.FancyArrowPatch
lines.Line2D
patches.Polygon
patches.FancyBboxPatch
offsetbox.AnnotationBbox
patches.PathPatch
text._AnnotationBase
patches.Rectangle
text.TextWithDash

How to Check?

In [11]:

```
type(title).__bases__
```

Out[11]:

```
(matplotlib.artist.Artist,)
```

What About the Figure, Axes?

```
x = np.linspace(0, 3*np.pi, 500)
line = plt.plot(x, np.sin(x**2))
title = plt.title('A simple chirp')
fig = plt.gcf()
ax = plt.gca()
```

Not in Reverse…

```
x = np.linspace(0, 3*np.pi, 500)
fig = plt.figure()
# choose one... (default is the first)
ax = fig.add_subplot(1,1,1)
ax = fig.add_axes([.2,.2,.7,.7])
line = ax.plot(x, np.sin(x**2))
title = ax.set_title('A simple chirp')
```

Now We Can Get More Specific

```
x = np.linspace(0, 3*np.pi, 500)
fig = plt.figure(figsize=(8,5))
ax = fig.add_axes([.2,.2,.7,.7])
line = ax.plot(x, np.sin(x**2))
title = ax.set_title('A simple chirp')
ax.set_ylim([-1.1,1.1])
```

Every Artist Inherits These

Property	Description
alpha	The transparency - a scalar from 0-1
animated	A boolean that is used to facilitate animated drawing
axes	The axes that the Artist lives in, possibly None
clip_box	The bounding box that clips the Artist
clip_on	Whether clipping is enabled
clip_path	The path the artist is clipped to
contains	A picking function to test whether the artist contains the pick point
figure	The figure instance the artist lives in, possibly None
label	A text label (e.g., for auto-labeling)
picker	A python object that controls object picking
transform	The transformation
visible	A boolean whether the artist should be drawn
zorder	A number which determines the drawing order
rasterized	Boolean; Turns vectors into rastergraphics: (for compression & eps transparency)

Why Axes?

"The matplotlib.axes.Axes is the center of the matplotlib universe"
—The Docs

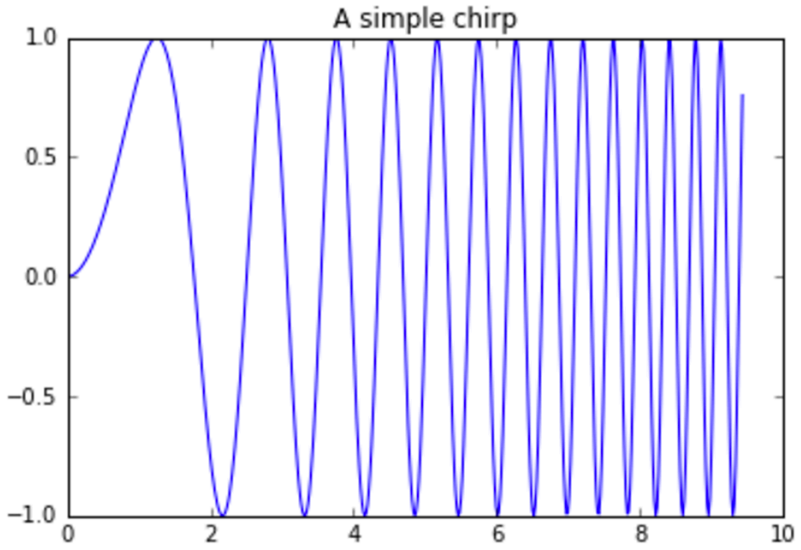
Tips

Use `figsize` wisely: always work at final size.

Tips

Use `figsize` wisely: always work at final size.

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



Tips

Use `figsize` wisely: always work at final size.

Work with `axes` objects to build custom plots.

Tips

Use `figsize` wisely: always work at final size.

Work with `axes` objects to build custom plots.

Keep track of styles in a stylesheet.

Berkeley SCHOOL OF INFORMATION