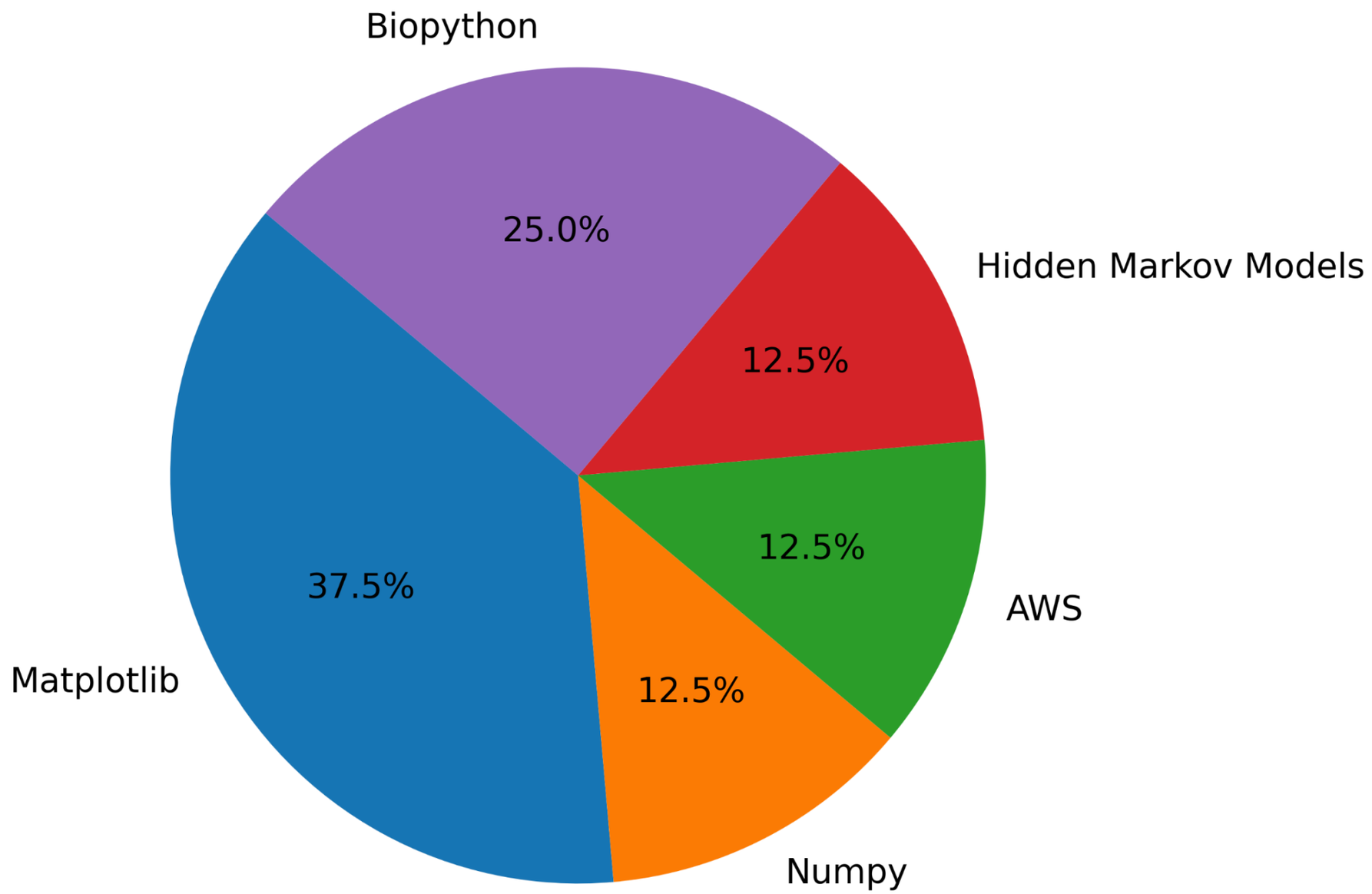

matplotlib

Poll

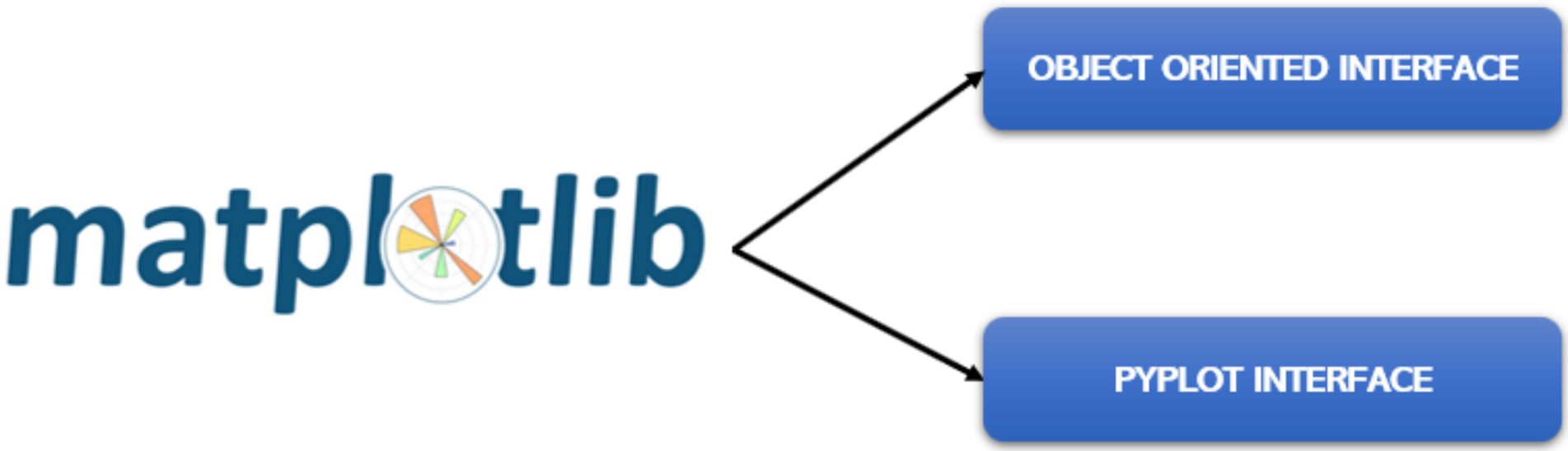

Biopython
25.0%
Hidden Markov Models
12.5%
AWS
12.5%
Numpy
12.5%
Matplotlib
37.5%

Poll

```
>>> import matplotlib.pyplot as plt
...
... # Data for the pie chart
... labels = ['Matplotlib', 'Numpy', 'AWS', 'Hidden Markov Models', 'Biopython']
... sizes = [37.5, 12.5, 12.5, 12.5, 25]
...
... # Creating the pie chart
... plt.figure(figsize=(6, 4))
... plt.pie(sizes, labels=labels, autopct='%1.1f%%', startangle=140)
... plt.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle.
...
... # Show the chart
... plt.show()
```


matplotlib

Interfaces



https://miro.medium.com/v2/resize:fit:1400/1*9hl9HkkvXgQ3orS7c6D8dA.png

PYLAB

Implicit vs explicit interface

implicit

```
>>> import matplotlib.pyplot as plt
...
... # Sample data
... x = [1, 2, 3, 4, 5]
... y = [2, 3, 5, 7, 11]
...
... # Creating the plot using pyplot
... plt.plot(x, y)
... plt.xlabel("X-axis")
... plt.ylabel("Y-axis")
...
... # Show plot
... plt.show()
```

explicit

```
>>> import matplotlib.pyplot as plt
...
... # Sample data
... x = [1, 2, 3, 4, 5]
... y = [2, 3, 5, 7, 11]
...
... # Creating the figure and axis objects
... fig, ax = plt.subplots()
...
... # Creating the plot using OO interface
... ax.plot(x, y)
... ax.set_xlabel("x-axis")
... ax.set_ylabel("y-axis")
...
... # Show plot
... fig.show()
```

Handling figures

implicit

```
>>> plt.gcf().canvas.get_supported_filetypes()
{'eps': 'Encapsulated Postscript', 'jpg': 'Joint Photographic Experts Group',
>>> plt.savefig('test.png', transparent=False, dpi=400, bbox_inches="tight")
>>> plt.show()
>>> plt.close()
>>> plt.clf()
```

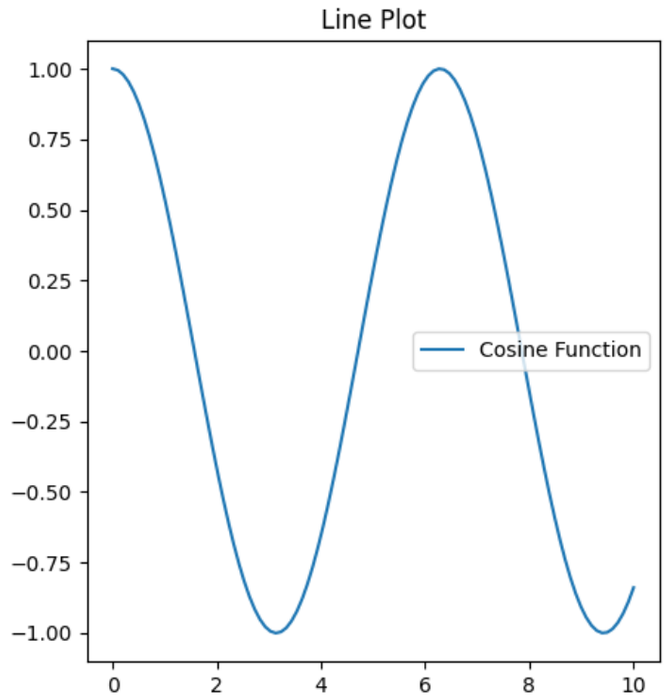
explicit

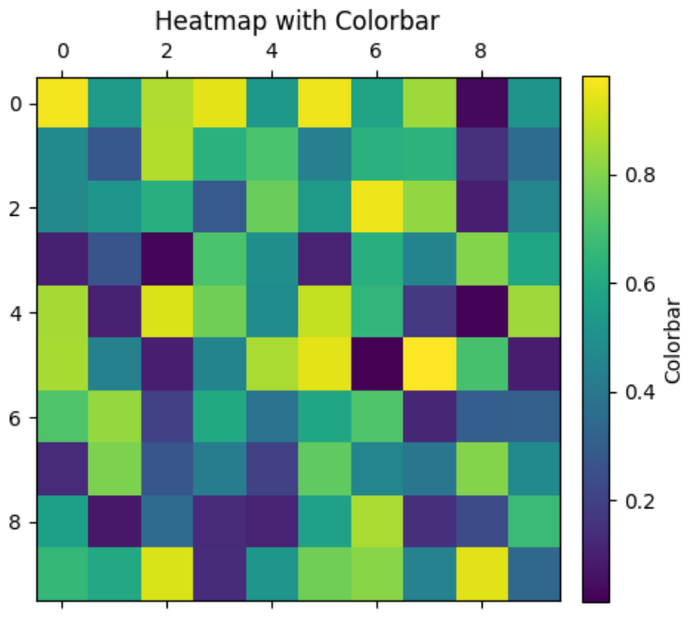
```
>>> fig.canvas.get_supported_filetypes()
{'eps': 'Encapsulated Postscript', 'jpg': 'Joint Photographic Experts Group',
>>> fig.savefig('test.png', transparent=False, dpi=400, bbox_inches="tight")
>>> fig.show()
>>> plt.close(fig)
```

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

Figure and Axes

```python
import matplotlib.pyplot as plt
import numpy as np

# Sample data
x = np.linspace(0, 10, 100)
y = np.cos(x)
data = np.random.rand(10, 10)

# Creating a figure and two subplots
fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(5, 12))

# First subplot - Line Plot
ax1.plot(x, y, label='Cosine Function')
ax1.set_title("Line Plot")
ax1.legend()

# Second subplot - Heatmap
cax = ax2.matshow(data, cmap='viridis')
cbar = fig.colorbar(cax, ax=ax2, fraction=0.046, pad=0.04)
cbar.set_label('Colorbar')
ax2.set_title("Heatmap with Colorbar")

# Show the plot
fig.show()
```





Figure

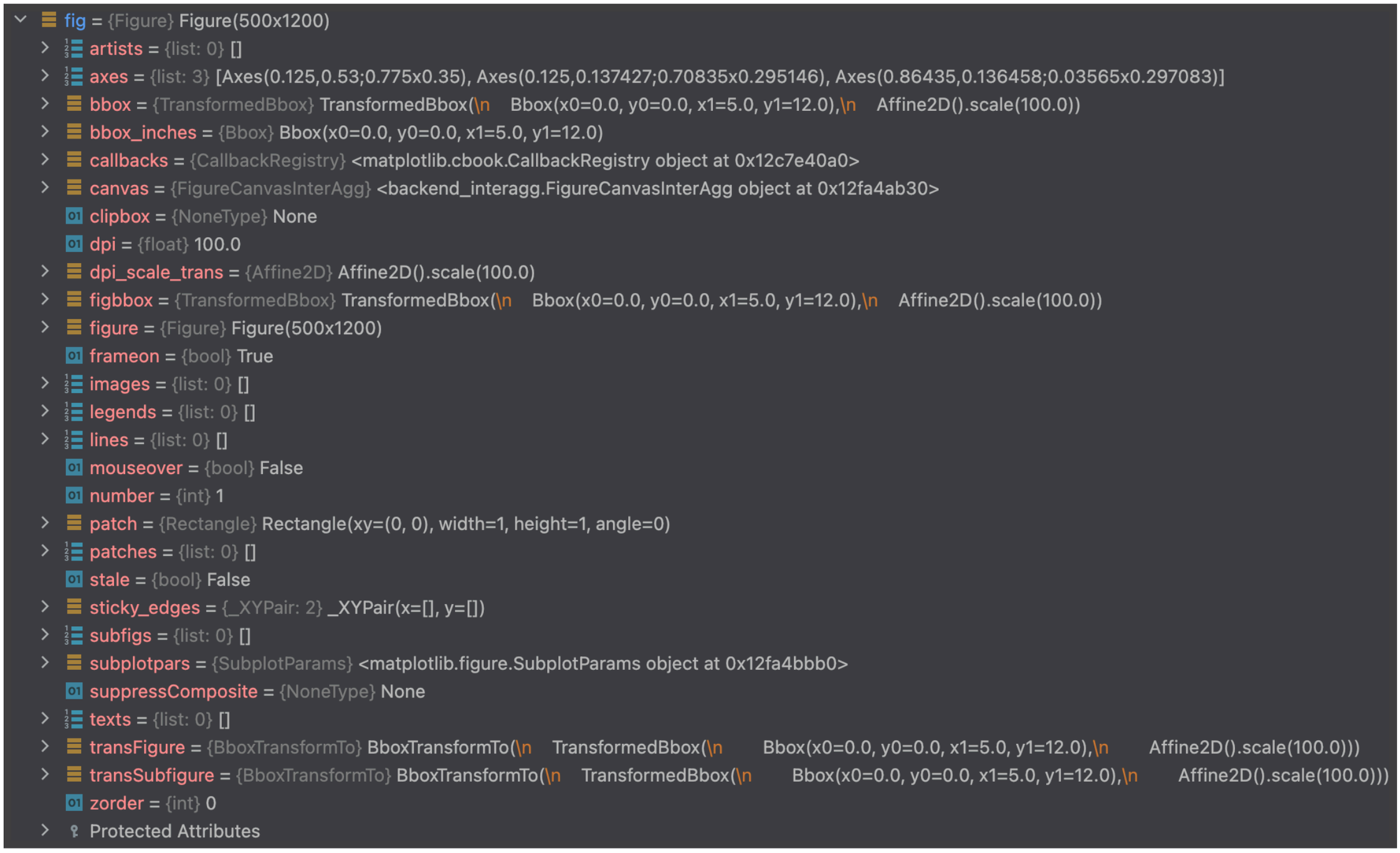
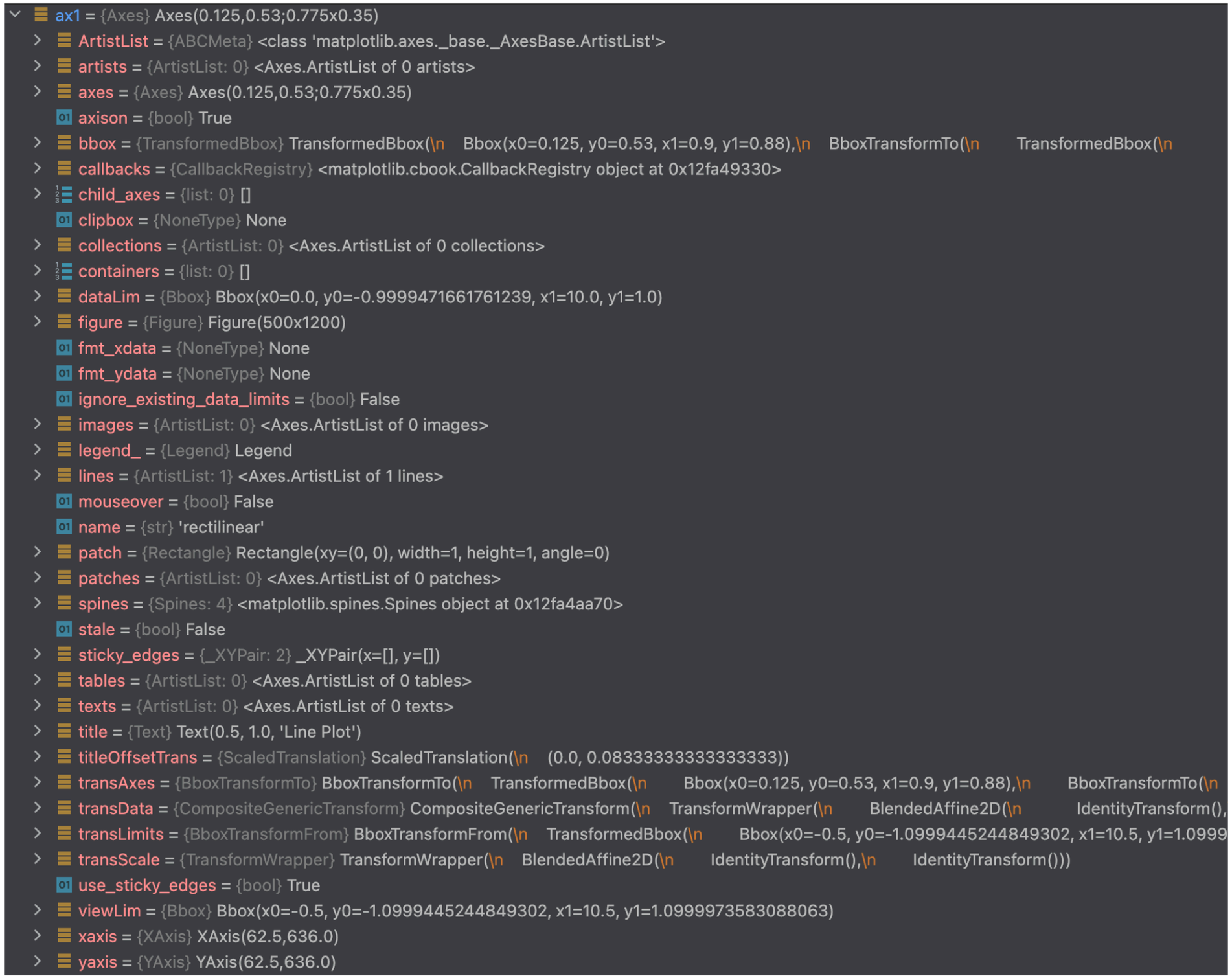


fig = {Figure} Figure(500x1200)
artists = {list: 0} []
axes = {list: 3} [Axes(0.125,0.53;0.775x0.35), Axes(0.125,0.137427;0.70835x0.295146), Axes(0.86435,0.136458;0.03565x0.297083)]
bbox = {TransformedBbox} TransformedBbox(\n Bbox(x0=0.0, y0=0.0, x1=5.0, y1=12.0),\n Affine2D().scale(100.0))
bbox_inches = {Bbox} Bbox(x0=0.0, y0=0.0, x1=5.0, y1=12.0)
callbacks = {CallbackRegistry} <matplotlib.cbook.CallbackRegistry object at 0x12c7e40a0>
canvas = {FigureCanvasInterAgg} <backend_interagg.FigureCanvasInterAgg object at 0x12fa4ab30>
clipbox = {NoneType} None
dpi = {float} 100.0
dpi_scale_trans = {Affine2D} Affine2D().scale(100.0)
figbbox = {TransformedBbox} TransformedBbox(\n Bbox(x0=0.0, y0=0.0, x1=5.0, y1=12.0),\n Affine2D().scale(100.0))
figure = {Figure} Figure(500x1200)
frameon = {bool} True
images = {list: 0} []
legends = {list: 0} []
lines = {list: 0} []
mouseover = {bool} False
number = {int} 1
patch = {Rectangle} Rectangle(xy=(0, 0), width=1, height=1, angle=0)
patches = {list: 0} []
stale = {bool} False
sticky_edges = {_XYPair: 2} _XYPair(x=[], y=[])
subfigs = {list: 0} []
subplotpars = {SubplotParams} <matplotlib.figure.SubplotParams object at 0x12fa4bbb0>
suppressComposite = {NoneType} None
texts = {list: 0} []
transFigure = {BboxTransformTo} BboxTransformTo(\n TransformedBbox(\n Bbox(x0=0.0, y0=0.0, x1=5.0, y1=12.0),\n Affine2D().scale(100.0)))
transSubfigure = {BboxTransformTo} BboxTransformTo(\n TransformedBbox(\n Bbox(x0=0.0, y0=0.0, x1=5.0, y1=12.0),\n Affine2D().scale(100.0)))
zorder = {int} 0
Protected Attributes

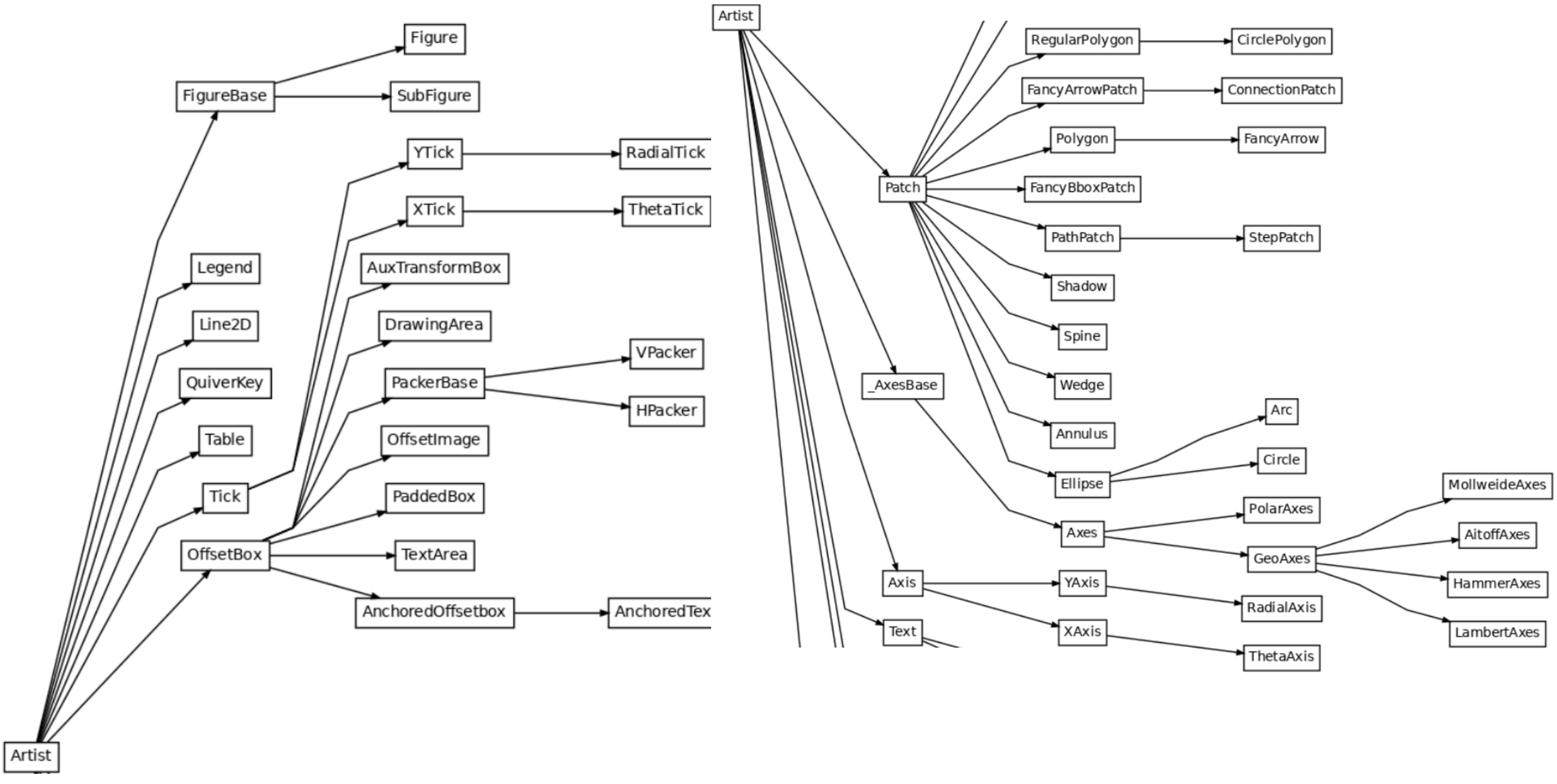
Axes


ax1 = {Axes} Axes(0.125,0.53;0.775x0.35)
ArtistList = {ABCMeta} <class 'matplotlib.axes._base._AxesBase.ArtistList'>
artists = {ArtistList: 0} <Axes.ArtistList of 0 artists>
axes = {Axes} Axes(0.125,0.53;0.775x0.35)
axison = {bool} True
bbox = {TransformedBbox} TransformedBbox(\n Bbox(x0=0.125, y0=0.53, x1=0.9, y1=0.88),\n BboxTransformTo(\n TransformedBbox(\n
callbacks = {CallbackRegistry} <matplotlib.cbook.CallbackRegistry object at 0x12fa49330>
child_axes = {list: 0} []
clipbox = {NoneType} None
collections = {ArtistList: 0} <Axes.ArtistList of 0 collections>
containers = {list: 0} []
dataLim = {Bbox} Bbox(x0=0.0, y0=-0.9999471661761239, x1=10.0, y1=1.0)
figure = {Figure} Figure(500x1200)
fmt_xdata = {NoneType} None
fmt_ydata = {NoneType} None
ignore_existing_data_limits = {bool} False
images = {ArtistList: 0} <Axes.ArtistList of 0 images>
legend_ = {Legend} Legend
lines = {ArtistList: 1} <Axes.ArtistList of 1 lines>
mouseover = {bool} False
name = {str} 'rectilinear'
patch = {Rectangle} Rectangle(xy=(0, 0), width=1, height=1, angle=0)
patches = {ArtistList: 0} <Axes.ArtistList of 0 patches>
spines = {Spines: 4} <matplotlib.spines.Spines object at 0x12fa4aa70>
stale = {bool} False
sticky_edges = {_XYPair: 2} _XYPair(x=[], y=[])
tables = {ArtistList: 0} <Axes.ArtistList of 0 tables>
texts = {ArtistList: 0} <Axes.ArtistList of 0 texts>
title = {Text} Text(0.5, 1.0, 'Line Plot')
titleOffsetTrans = {ScaledTranslation} ScaledTranslation(\n (0.0, 0.08333333333333333))
transAxes = {BboxTransformTo} BboxTransformTo(\n TransformedBbox(\n Bbox(x0=0.125, y0=0.53, x1=0.9, y1=0.88),\n BboxTransformTo(\n
transData = {CompositeGenericTransform} CompositeGenericTransform(\n TransformWrapper(\n BlendedAffine2D(\n IdentityTransform(),
transLimits = {BboxTransformFrom} BboxTransformFrom(\n TransformedBbox(\n Bbox(x0=-0.5, y0=-1.0999445244849302, x1=10.5, y1=1.0999
transScale = {TransformWrapper} TransformWrapper(\n BlendedAffine2D(\n IdentityTransform(),\n IdentityTransform()))
use_sticky_edges = {bool} True
viewLim = {Bbox} Bbox(x0=-0.5, y0=-1.0999445244849302, x1=10.5, y1=1.0999973583088063)
xaxis = {XAxis} XAxis(62.5,636.0)
yaxis = {YAxis} YAxis(62.5,636.0)

Using Artist interface

```
>>> import matplotlib.pyplot as plt
... from matplotlib.lines import Line2D
... from matplotlib.text import Text
...
... # Sample data
... x = [1, 2, 3, 4, 5]
... y = [2, 3, 5, 7, 11]
...
... # Create a figure and an axes
... fig = plt.figure()
... ax = fig.add_subplot()
...
... # Create a line object and add it to the axes
... line = Line2D(x, y)
... ax.add_line(line)
...
... # Set the limits
... ax.set_xlim(min(x), max(x))
... ax.set_ylim(min(y), max(y))
...
... # Add title and labels using Text artists
... ax._add_text(Text(x=0.5, y=1.05, text="Line Plot using Pyplot", transform=ax.transAxes, ha='center', va='bottom'))
... ax._add_text(Text(x=0.5, y=-0.05, text="x-axis", transform=ax.transAxes, ha='center', va='top'))
... ax._add_text(Text(x=-0.05, y=0.5, text="y-axis", transform=ax.transAxes, ha='right', va='center', rotation='vertical'))
...
... fig.show()
```

Artist class



Artist class

`Artist` class

class **matplotlib.artist.Artist** **[source]**

Abstract base class for objects that render into a FigureCanvas.

Typically, all visible elements in a figure are subclasses of Artist.

Interactive

Artist.add_callback	Add a callback function that will be called whenever one of the **Artist**'s properties changes.
Artist.remove_callback	Remove a callback based on its observer id.
Artist.pchanged	Call all of the registered callbacks.
Artist.get_cursor_data	Return the cursor data for a given event.
Artist.format_cursor_data	Return a string representation of *data*.
Artist.set_mouseover	Set whether this artist is queried for custom context information when the mouse cursor moves over it.

Layers of matplotlib API

`matplotlib.artist.Artist`

`matplotlib.backend_bases.Renderer`

`matplotlib.backend_bases.FigureCanvas`

Figure

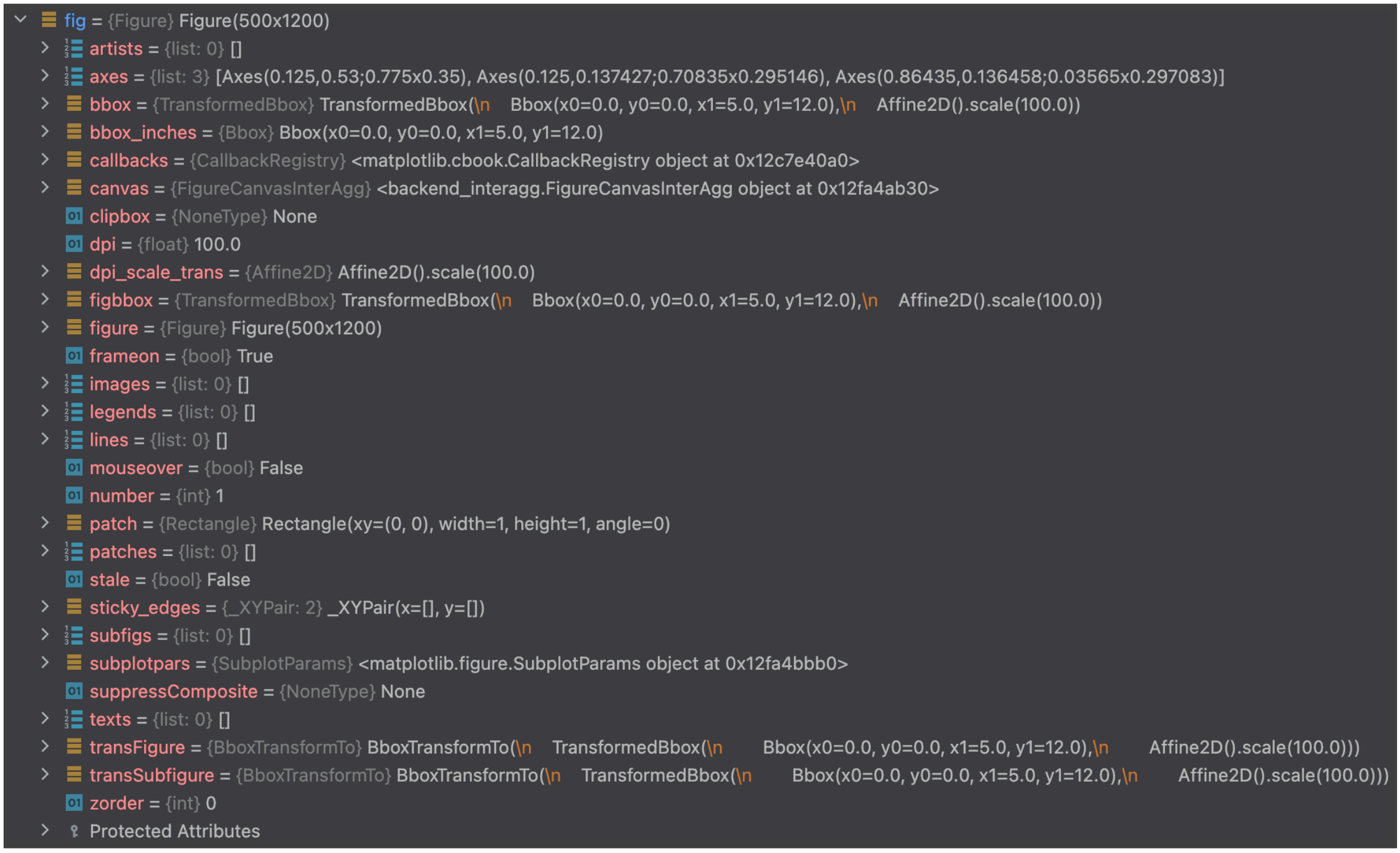


fig = {Figure} Figure(500x1200)
artists = {list: 0} []
axes = {list: 3} [Axes(0.125,0.53;0.775x0.35), Axes(0.125,0.137427;0.70835x0.295146), Axes(0.86435,0.136458;0.03565x0.297083)]
bbox = {TransformedBbox} TransformedBbox(\n Bbox(x0=0.0, y0=0.0, x1=5.0, y1=12.0),\n Affine2D().scale(100.0))
bbox_inches = {Bbox} Bbox(x0=0.0, y0=0.0, x1=5.0, y1=12.0)
callbacks = {CallbackRegistry} <matplotlib.cbook.CallbackRegistry object at 0x12c7e40a0>
canvas = {FigureCanvasInterAgg} <backend_interagg.FigureCanvasInterAgg object at 0x12fa4ab30>
clipbox = {NoneType} None
dpi = {float} 100.0
dpi_scale_trans = {Affine2D} Affine2D().scale(100.0)
figbbox = {TransformedBbox} TransformedBbox(\n Bbox(x0=0.0, y0=0.0, x1=5.0, y1=12.0),\n Affine2D().scale(100.0))
figure = {Figure} Figure(500x1200)
frameon = {bool} True
images = {list: 0} []
legends = {list: 0} []
lines = {list: 0} []
mouseover = {bool} False
number = {int} 1
patch = {Rectangle} Rectangle(xy=(0, 0), width=1, height=1, angle=0)
patches = {list: 0} []
stale = {bool} False
sticky_edges = {_XYPair: 2} _XYPair(x=[], y=[])
subfigs = {list: 0} []
subplotpars = {SubplotParams} <matplotlib.figure.SubplotParams object at 0x12fa4bbb0>
suppressComposite = {NoneType} None
texts = {list: 0} []
transFigure = {BboxTransformTo} BboxTransformTo(\n TransformedBbox(\n Bbox(x0=0.0, y0=0.0, x1=5.0, y1=12.0),\n Affine2D().scale(100.0)))
transSubfigure = {BboxTransformTo} BboxTransformTo(\n TransformedBbox(\n Bbox(x0=0.0, y0=0.0, x1=5.0, y1=12.0),\n Affine2D().scale(100.0)))
zorder = {int} 0
Protected Attributes

Comparison

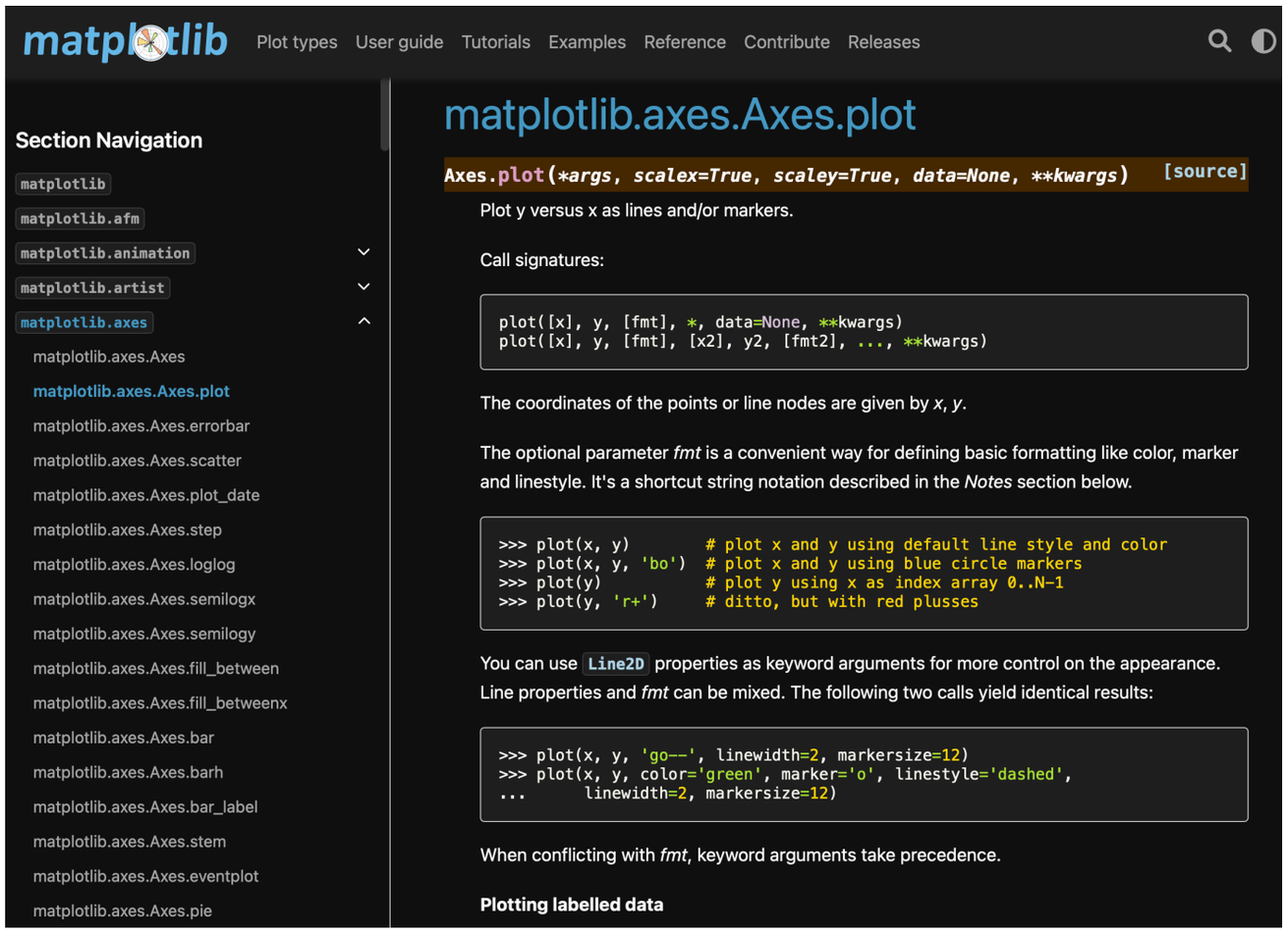
implicit pyplot interface

```
>>> import matplotlib.pyplot as plt
...
... # Sample data
... x = [1, 2, 3, 4, 5]
... y = [2, 3, 5, 7, 11]
...
... # Creating the plot using pyplot
... plt.plot(x, y)
... plt.xlabel("X-axis")
... plt.ylabel("Y-axis")
...
... # Show plot
... plt.show()
```

Artist interface

```
>>> import matplotlib.pyplot as plt
... from matplotlib.lines import Line2D
... from matplotlib.text import Text
...
... # Sample data
... x = [1, 2, 3, 4, 5]
... y = [2, 3, 5, 7, 11]
...
... # Create a figure and an axes
... fig = plt.figure()
... ax = fig.add_subplot()
...
... # Create a line object and add it to the axes
... line = Line2D(x, y)
... ax.add_line(line)
...
... # Set the limits
... ax.set_xlim(min(x), max(x))
... ax.set_ylim(min(y), max(y))
...
... # Add title and labels using Text artists
... ax._add_text(Text(x=0.5, y=1.05, text="Line Plot
... ax._add_text(Text(x=0.5, y=-0.05, text="x-axis",
... ax._add_text(Text(x=-0.05, y=0.5, text="y-axis",
...
... fig.show()
```

Resources



Link

Create same plot using the pyplot interface

```
>>> import matplotlib.pyplot as plt
... import numpy as np
... from matplotlib.lines import Line2D
... from matplotlib.patches import Rectangle
...
... x = np.linspace(0, 10, 100)
... y1 = np.sin(x)
... y2 = np.cos(x)
... data = np.random.normal(size=1000)
...
... fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(8, 6))
...
... ax1.add_line(Line2D(x, y1, label='sin(x)', color='blue'))
... ax1.add_line(Line2D(x, y2, label='cos(x)', color='green'))
... ax1.set_xlim(0, 10)
... ax1.set_ylim(-1, 1)
... ax1.set_xlabel('X-axis')
... ax1.set_ylabel('Y-axis')
... ax1.legend()
... ax1.set_title('Line Plots')
...
... n, bins = np.histogram(data, bins=30)
... for left, height in zip(bins[:-1], n):
...     rect = Rectangle((left, 0), bins[1] - bins[0], height, color='red', alpha=0.3)
...     ax2.add_patch(rect)
... ax2.set_xlim(min(bins), max(bins))
... ax2.set_ylim(min(n), max(n))
... ax2.set_xlabel('Value')
... ax2.set_ylabel('Frequency')
... ax2.set_title('Histogram')
...
... fig.suptitle('Line Plots and Histogram - Artist Interface', fontsize=16)
... fig.tight_layout(rect=[0, 0, 1, 0.95])
... fig.show()
```

Using Seaborn

```
>>> import matplotlib.pyplot as plt
... import seaborn as sns
... import numpy as np
... import pandas as pd
...
... # Create a dataframes
... df_lines = pd.DataFrame({
...     'X': np.linspace(0, 10, 100),
...     'sin(X)': np.sin(x),
...     'cos(X)': np.cos(x)
... })
...
... df_hist = pd.DataFrame({'Data': np.random.normal(size=1000)})
...
... # Create a figure and two subplots
... fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(8, 6))
...
... # First subplot - Line Plots using Seaborn
... sns.lineplot(x='X', y='sin(X)', data=df_lines, ax=ax1, label='sin(x)', color='blue')
... sns.lineplot(x='X', y='cos(X)', data=df_lines, ax=ax1, label='cos(x)', color='green')
... ax1.set_title('Line Plots')
...
... # Second subplot - Histogram using Seaborn
... sns.histplot(data=df_hist, x='Data', bins=30, color='red', ax=ax2, alpha=0.3)
... ax2.set_title('Histogram')
...
... fig.suptitle('Line Plots and Histogram - Seaborn', fontsize=16)
... plt.tight_layout(rect=[0, 0, 1, 0.95])
... plt.show()
```