

# Matplotlib

## Overview

As we delve deeper into our Macroeconomics course, understanding and visualizing economic data becomes increasingly crucial. To aid us in this endeavor, I would like to introduce you to a powerful tool in Python: Matplotlib.

Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter, wxPython, Qt, or GTK. Developed by John D. Hunter in 2003, Matplotlib has since become an essential part of the Python ecosystem for data analysis and visualization.

## Why Use Matplotlib?

1. **Versatility:** It supports a wide range of graph types, from simple line plots to complex 3D plots.
2. **Customization:** You can easily customize plots with various styles, colors, and annotations.
3. **Integration:** It integrates seamlessly with other Python libraries like Pandas, making data analysis more efficient.
4. **Documentation:** It has extensive documentation and a supportive community, which makes learning and troubleshooting easier.

## Matplotlib's Split Personality

Matplotlib is unusual in that it offers two different interfaces to plotting.

One is a simple MATLAB-style API (Application Programming Interface) that was written to help MATLAB refugees find a ready home.

The other is a more "Pythonic" object-oriented API.

**For reasons described below, we recommend that you use the second API.**

But first, let's discuss the difference.

## The APIs

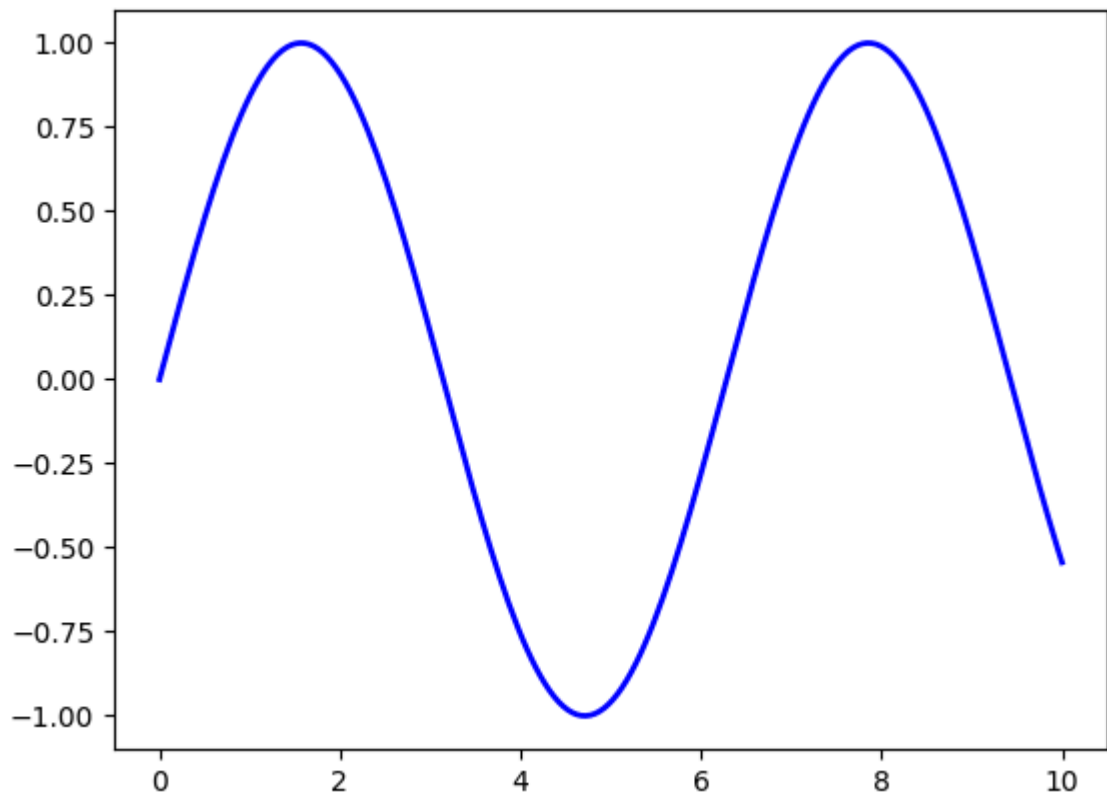
### The MATLAB-style API

Here's the kind of easy example you might find in introductory treatments

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

x = np.linspace(0, 10, 200)
y = np.sin(x)

plt.plot(x, y, 'b-', linewidth=2)
plt.show()
```



This is simple and convenient, but also somewhat limited and un-Pythonic.

For example, in the function calls, a lot of objects get created and passed around without making themselves known to the programmer.

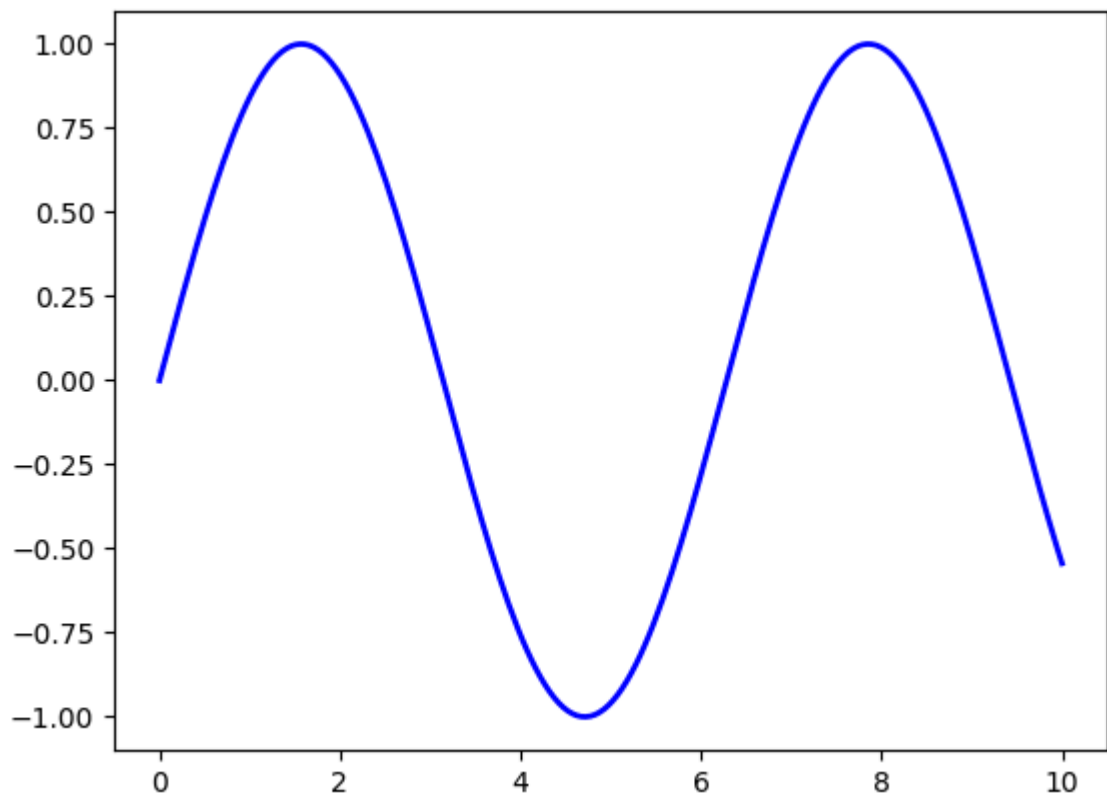
Python programmers tend to prefer a more explicit style of programming (run `import this` in a code block and look at the second line).

This leads us to the alternative, object-oriented Matplotlib API.

## The Object-Oriented API

Here's the code corresponding to the preceding figure using the object-oriented API

```
In [ ]: fig, ax = plt.subplots()
ax.plot(x, y, 'b-', linewidth=2)
plt.show()
```



Here the call `fig, ax = plt.subplots()` returns a pair, where

- `fig` is a `Figure` instance—like a blank canvas.
- `ax` is an `AxesSubplot` instance—think of a frame for plotting in.

The `plot()` function is actually a method of `ax`.

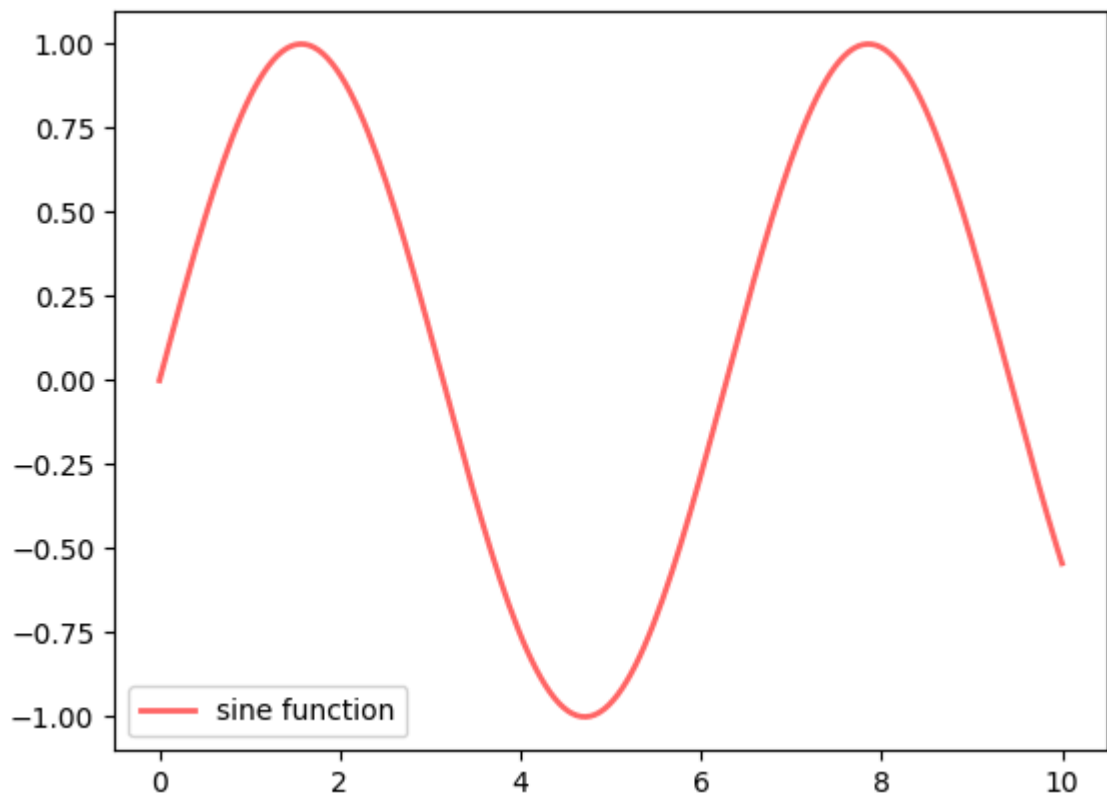
While there's a bit more typing, the more explicit use of objects gives us better control.

This will become more clear as we go along.

## Tweaks

Here we've changed the line to red and added a legend

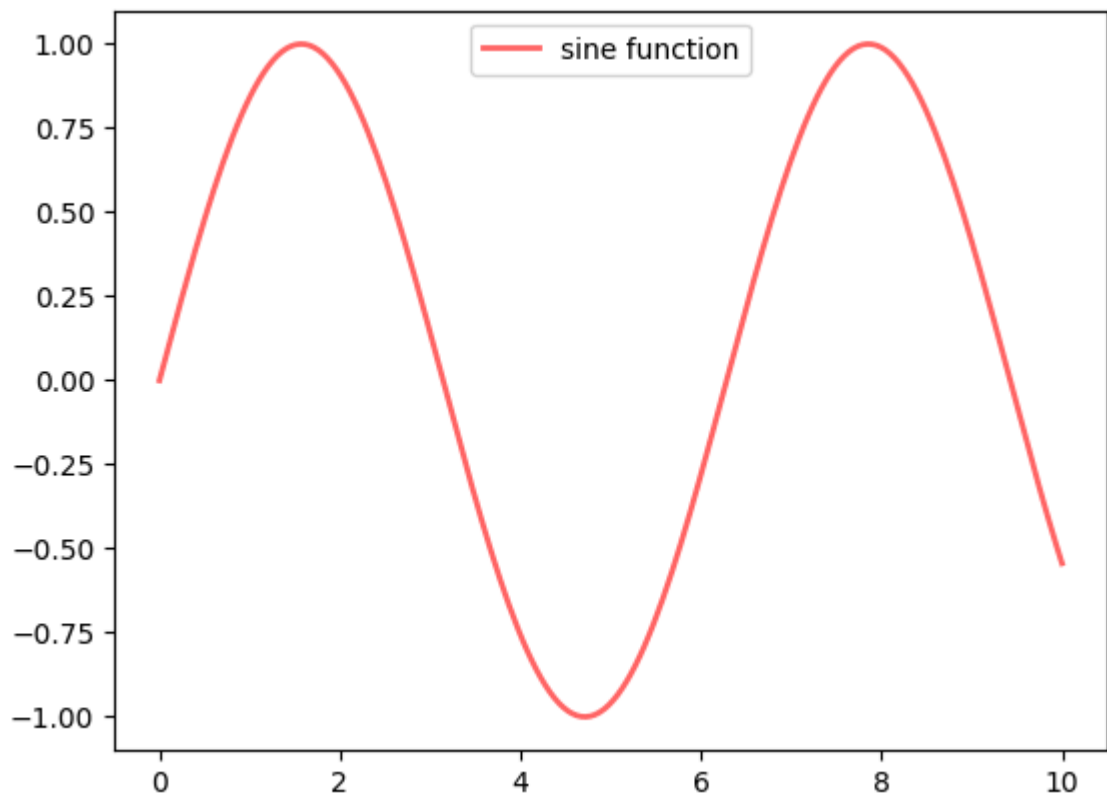
```
In [ ]: fig, ax = plt.subplots()
ax.plot(x, y, 'r-', linewidth=2, label='sine function', alpha=0.6)
ax.legend()
plt.show()
```



We've also used `alpha` to make the line slightly transparent—which makes it look smoother.

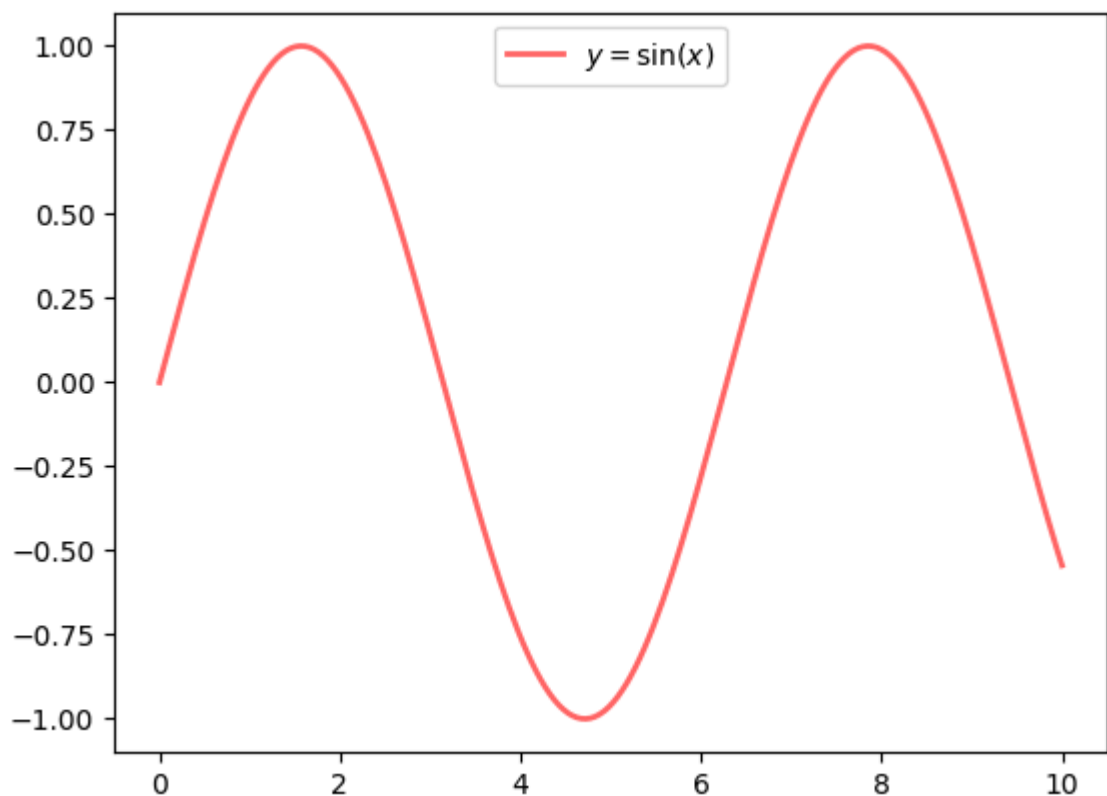
The location of the legend can be changed by replacing `ax.legend()` with `ax.legend(loc='upper center')`.

```
In [ ]: fig, ax = plt.subplots()
ax.plot(x, y, 'r-', linewidth=2, label='sine function', alpha=0.6)
ax.legend(loc='upper center')
plt.show()
```



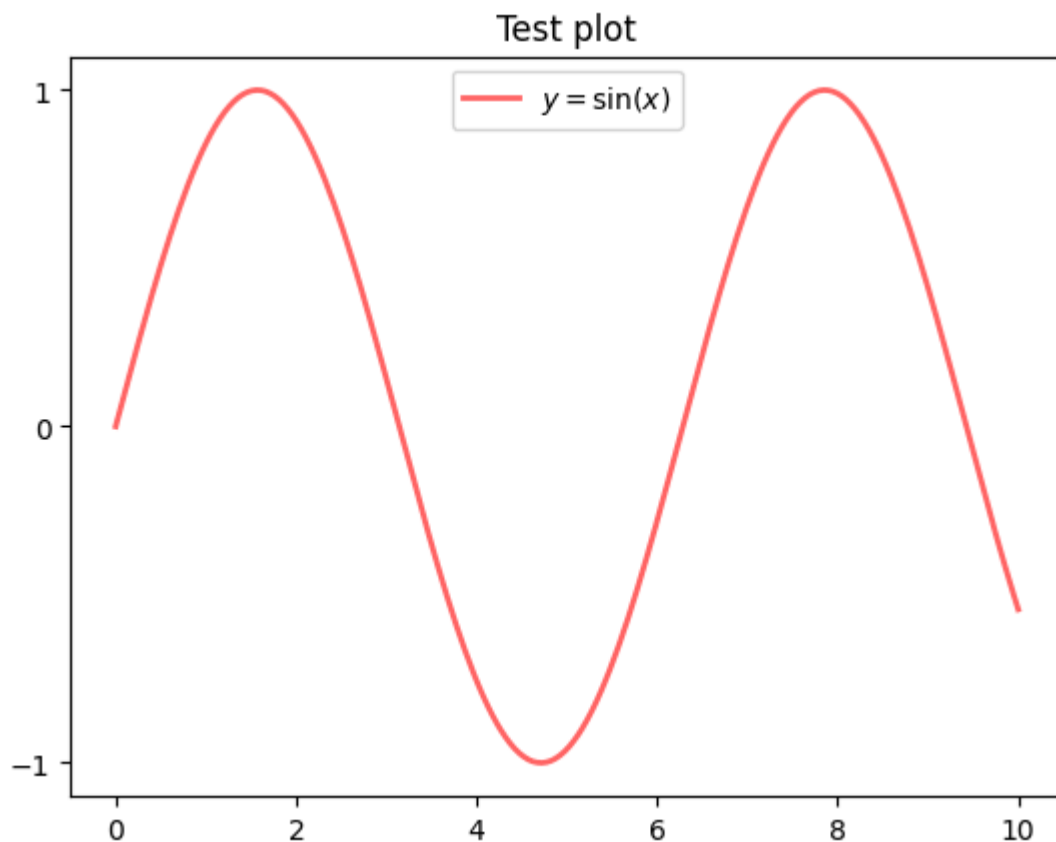
If everything is properly configured, then adding LaTeX is trivial

```
In [ ]: fig, ax = plt.subplots()
ax.plot(x, y, 'r-', linewidth=2, label='$y=\sin(x)$', alpha=0.6)
ax.legend(loc='upper center')
plt.show()
```



Controlling the ticks, adding titles and so on is also straightforward

```
In [ ]: fig, ax = plt.subplots()
ax.plot(x, y, 'r-', linewidth=2, label='$y=\sin(x)$', alpha=0.6)
ax.legend(loc='upper center')
ax.set_yticks([-1, 0, 1])
ax.set_title('Test plot')
plt.show()
```



## More Features

Matplotlib has a huge array of functions and features, which you can discover over time as you have need for them.

We mention just a few.

## Multiple Plots on One Axis

It's straightforward to generate multiple plots on the same axes.

Here's an example that randomly generates three normal densities and adds a label with their mean

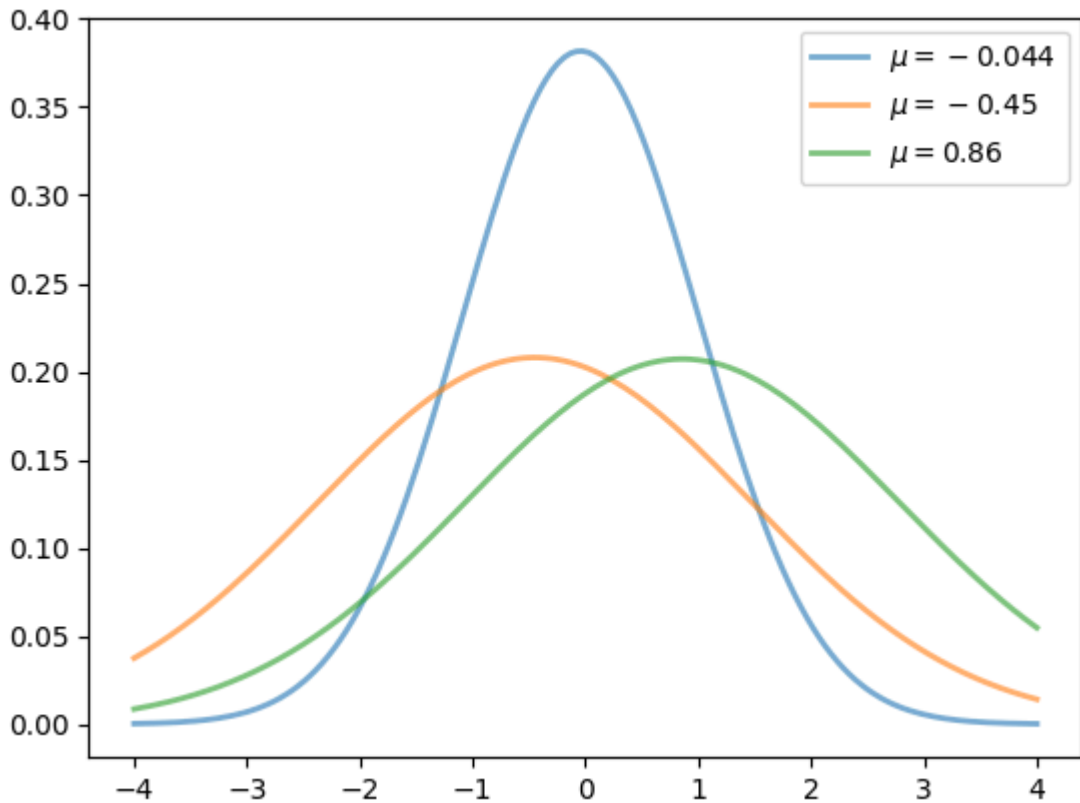
```
In [ ]: from scipy.stats import norm
from random import uniform

fig, ax = plt.subplots()
x = np.linspace(-4, 4, 150)
for i in range(3):
    m, s = uniform(-1, 1), uniform(1, 2)
    y = norm.pdf(x, loc=m, scale=s)
```

```

current_label = f'$\mu = {m:.2}$'
ax.plot(x, y, linewidth=2, alpha=0.6, label=current_label)
ax.legend()
plt.show()

```



## Multiple Subplots

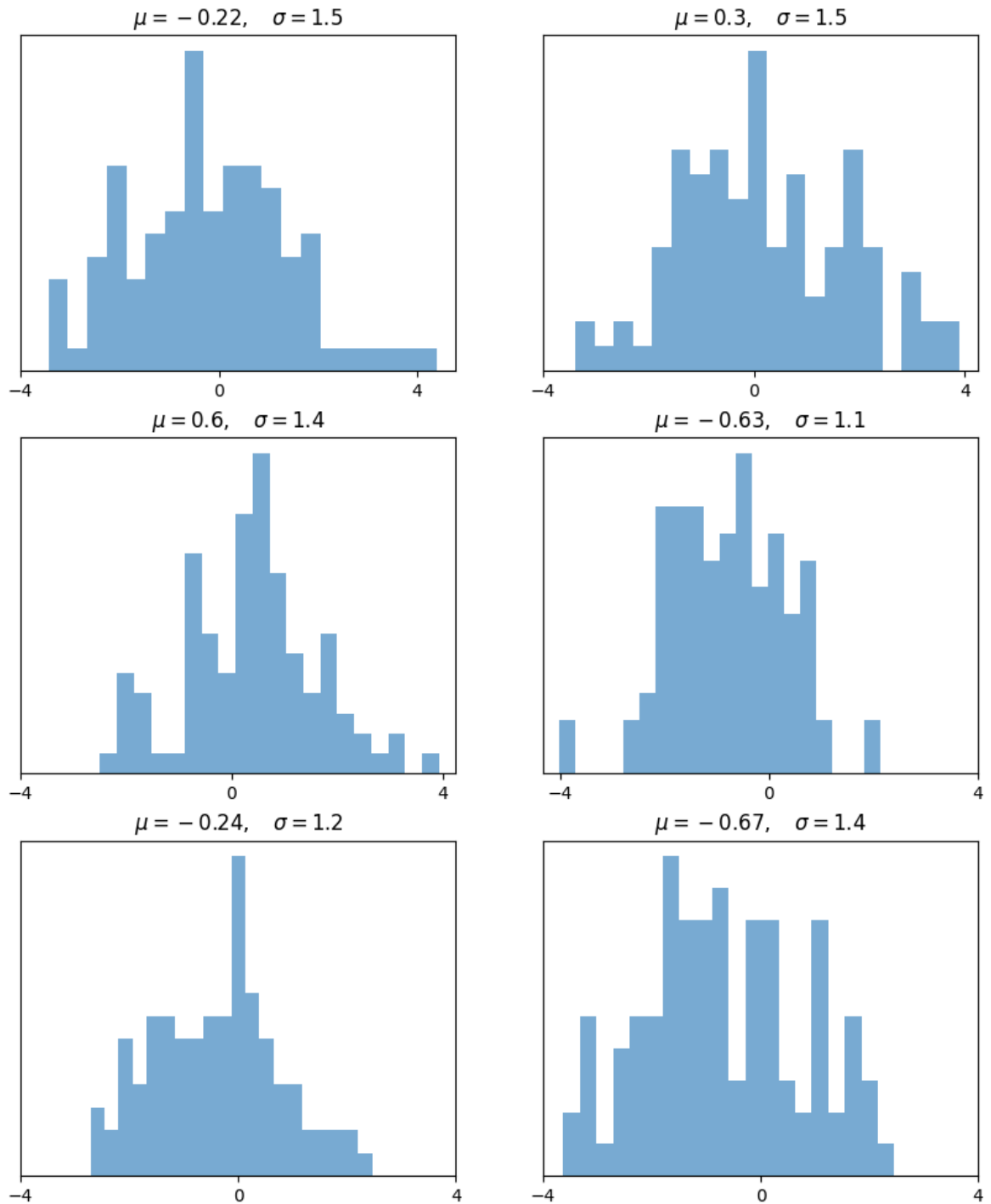
Sometimes we want multiple subplots in one figure.

Here's an example that generates 6 histograms

```

In [ ]: num_rows, num_cols = 3, 2
fig, axes = plt.subplots(num_rows, num_cols, figsize=(10, 12))
for i in range(num_rows):
    for j in range(num_cols):
        m, s = uniform(-1, 1), uniform(1, 2)
        x = norm.rvs(loc=m, scale=s, size=100)
        axes[i, j].hist(x, alpha=0.6, bins=20)
        t = f'$\mu = {m:.2}$, \quad \sigma = {s:.2}$'
        axes[i, j].set(title=t, xticks=[-4, 0, 4], yticks=[])
plt.show()

```



## 3D Plots

Matplotlib does a nice job of 3D plots — here is one example

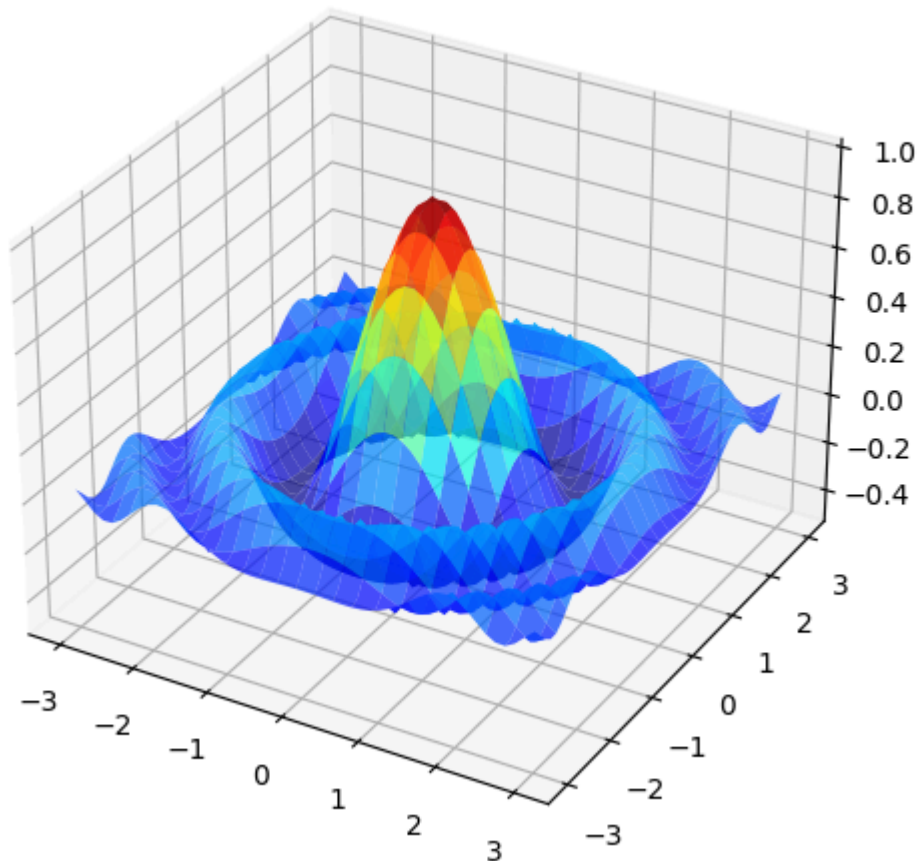
```
In [ ]: from mpl_toolkits.mplot3d.axes3d import Axes3D
        from matplotlib import cm

        def f(x, y):
            return np.cos(x**2 + y**2) / (1 + x**2 + y**2)

        xgrid = np.linspace(-3, 3, 50)
        ygrid = xgrid
        x, y = np.meshgrid(xgrid, ygrid)
```



```
fig = plt.figure(figsize=(10, 6))
ax = fig.add_subplot(111, projection='3d')
ax.plot_surface(x,
                y,
                f(x, y),
                rstride=2, cstride=2,
                cmap=cm.jet,
                alpha=0.7,
                linewidth=0.25)
ax.set_zlim(-0.5, 1.0)
plt.show()
```



## A Customizing Function

Perhaps you will find a set of customizations that you regularly use.

Suppose we usually prefer our axes to go through the origin, and to have a grid.

Here's a nice example from [Matthew Doty](#) of how the object-oriented API can be used to build a custom `subplots` function that implements these changes.

Read carefully through the code and see if you can follow what's going on

```
In [ ]: def subplots():
        "Custom subplots with axes through the origin"
        fig, ax = plt.subplots()

        # Set the axes through the origin
        for spine in ['left', 'bottom']:
```

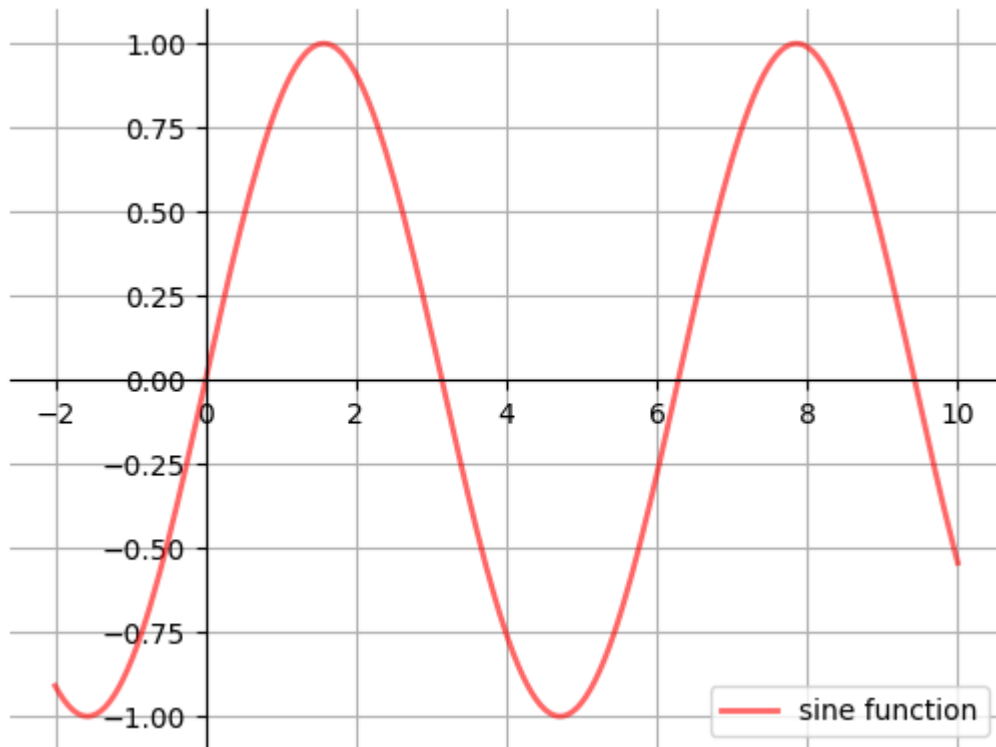
```

    ax.spines[spine].set_position('zero')
    for spine in ['right', 'top']:
        ax.spines[spine].set_color('none')

    ax.grid()
    return fig, ax

fig, ax = subplots() # Call the local version, not plt.subplots()
x = np.linspace(-2, 10, 200)
y = np.sin(x)
ax.plot(x, y, 'r-', linewidth=2, label='sine function', alpha=0.6)
ax.legend(loc='lower right')
plt.show()

```



The custom `subplots` function

1. calls the standard `plt.subplots` function internally to generate the `fig, ax` pair,
2. makes the desired customizations to `ax`, and
3. passes the `fig, ax` pair back to the calling code.

## Style Sheets

Another useful feature in Matplotlib is [style sheets](#).

We can use style sheets to create plots with uniform styles.

We can find a list of available styles by printing the attribute `plt.style.available`

```
In [ ]: print(plt.style.available)
```

```
['Solarize_Light2', '_classic_test_patch', '_mpl-gallery', '_mpl-gallery-nogrid', 'bmh', 'classic', 'dark_background', 'fast', 'fivethirtyeight', 'ggplot', 'grayscale', 'seaborn-v0_8', 'seaborn-v0_8-bright', 'seaborn-v0_8-colorblind', 'seaborn-v0_8-dark', 'seaborn-v0_8-dark-palette', 'seaborn-v0_8-darkgrid', 'seaborn-v0_8-deep', 'seaborn-v0_8-muted', 'seaborn-v0_8-notbook', 'seaborn-v0_8-paper', 'seaborn-v0_8-pastel', 'seaborn-v0_8-poster', 'seaborn-v0_8-talk', 'seaborn-v0_8-ticks', 'seaborn-v0_8-white', 'seaborn-v0_8-whitegrid', 'tableau-colorblind10']
```

We can now use the `plt.style.use()` method to set the style sheet.

Let's write a function that takes the name of a style sheet and draws different plots with the style

```
In [ ]: def draw_graphs(style='default'):

    # Setting a style sheet
    plt.style.use(style)

    fig, axes = plt.subplots(nrows=1, ncols=4, figsize=(10, 3))
    x = np.linspace(-13, 13, 150)

    # Set seed values to replicate results of random draws
    np.random.seed(9)

    for i in range(3):

        # Draw mean and standard deviation from uniform distributions
        m, s = np.random.uniform(-8, 8), np.random.uniform(2, 2.5)

        # Generate a normal density plot
        y = norm.pdf(x, loc=m, scale=s)
        axes[0].plot(x, y, linewidth=3, alpha=0.7)

        # Create a scatter plot with random X and Y values
        # from normal distributions
        rnormX = norm.rvs(loc=m, scale=s, size=150)
        rnormY = norm.rvs(loc=m, scale=s, size=150)
        axes[1].plot(rnormX, rnormY, ls='none', marker='o', alpha=0.7)

        # Create a histogram with random X values
        axes[2].hist(rnormX, alpha=0.7)

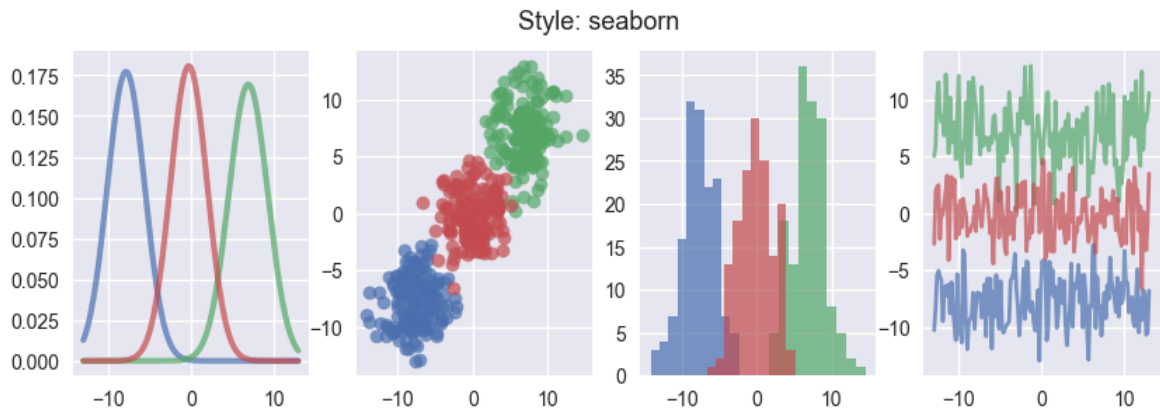
        # and a line graph with random Y values
        axes[3].plot(x, rnormY, linewidth=2, alpha=0.7)

    style_name = style.split('-')[0]
    plt.suptitle(f'Style: {style_name}', fontsize=13)
    plt.show()
```

Let's see what some of the styles look like.

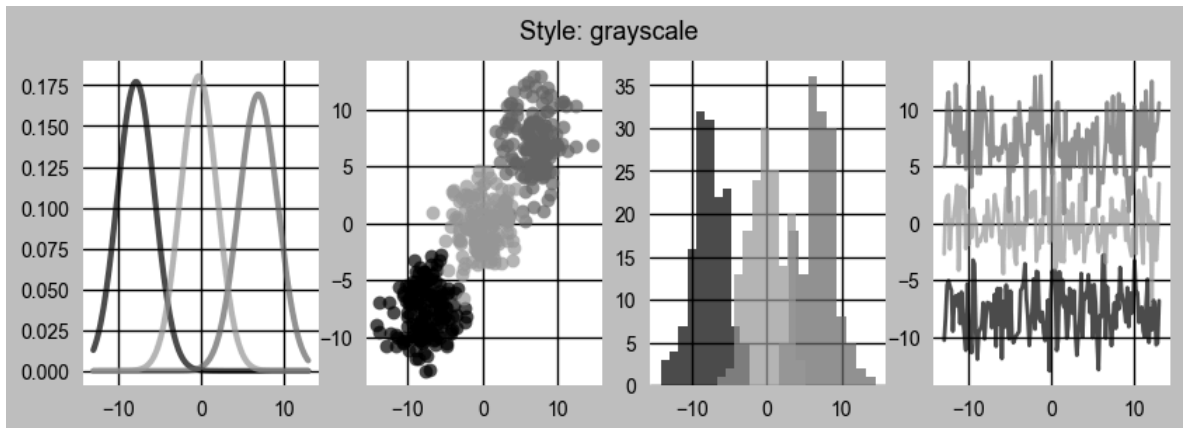
First, we draw graphs with the style sheet `seaborn`

```
In [ ]: draw_graphs(style='seaborn-v0_8')
```



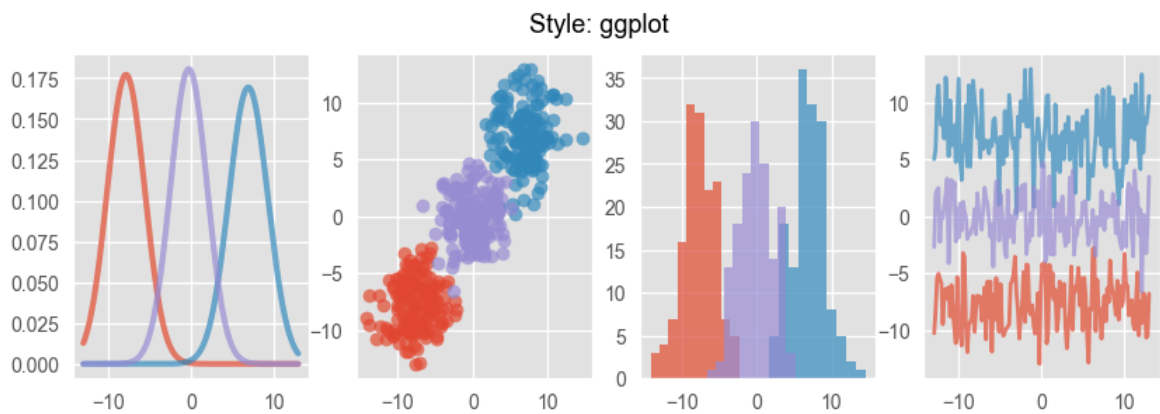
We can use `grayscale` to remove colors in plots

```
In [ ]: draw_graphs(style='grayscale')
```



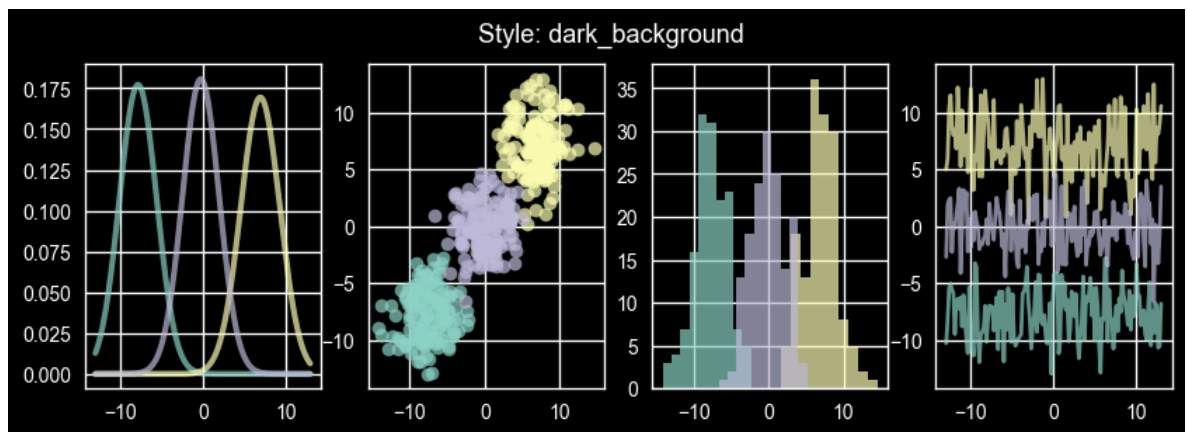
Here is what `ggplot` looks like

```
In [ ]: draw_graphs(style='ggplot')
```



We can also use the style `dark_background`

```
In [ ]: draw_graphs(style='dark_background')
```



You can use the function to experiment with other styles in the list.

If you are interested, you can even create your own style sheets.

Parameters for your style sheets are stored in a dictionary-like variable

`plt.rcParams`

```
In [ ]: print(plt.rcParams.keys())
```

```

KeysView(RcParams({'_internal.classic_mode': False,
    'agg.path.chunksize': 0,
    'animation.bitrate': -1,
    'animation.codec': 'h264',
    'animation.convert_args': ['-layers', 'OptimizePlus'],
    'animation.convert_path': 'convert',
    'animation.embed_limit': 20.0,
    'animation.ffmpeg_args': [],
    'animation.ffmpeg_path': 'ffmpeg',
    'animation.frame_format': 'png',
    'animation.html': 'none',
    'animation.writer': 'ffmpeg',
    'axes.autolimit_mode': 'data',
    'axes.axisbelow': True,
    'axes.edgecolor': 'white',
    'axes.facecolor': 'black',
    'axes.formatter.limits': [-5, 6],
    'axes.formatter.min_exponent': 0,
    'axes.formatter.offset_threshold': 4,
    'axes.formatter.use_locale': False,
    'axes.formatter.use_mathtext': False,
    'axes.formatter.useoffset': True,
    'axes.grid': True,
    'axes.grid.axis': 'both',
    'axes.grid.which': 'major',
    'axes.labelcolor': 'white',
    'axes.labelpad': 4.0,
    'axes.labelsize': 'large',
    'axes.labelweight': 'normal',
    'axes.linewidth': 1.0,
    'axes.prop_cycle': cycler('color', ['#8dd3c7', '#feffb3', '#bfbb
d9', '#fa8174', '#81b1d2', '#fdb462', '#b3de69', '#bc82bd', '#ccebc4', '#f
fed6f'])),
    'axes.spines.bottom': True,
    'axes.spines.left': True,
    'axes.spines.right': True,
    'axes.spines.top': True,
    'axes.titlecolor': 'auto',
    'axes.titlelocation': 'center',
    'axes.titlepad': 6.0,
    'axes.titlesize': 'x-large',
    'axes.titleweight': 'normal',
    'axes.titley': None,
    'axes.unicode_minus': True,
    'axes.xmargin': 0.05,
    'axes.ymargin': 0.05,
    'axes.zmargin': 0.05,
    'axes3d.grid': True,
    'axes3d.xaxis.panecolor': (0.95, 0.95, 0.95, 0.5),
    'axes3d.yaxis.panecolor': (0.9, 0.9, 0.9, 0.5),
    'axes3d.zaxis.panecolor': (0.925, 0.925, 0.925, 0.5),
    'backend': 'module://matplotlib_inline.backend_inline',
    'backend_fallback': True,
    'boxplot.bootstrap': None,
    'boxplot.boxprops.color': 'white',
    'boxplot.boxprops.linestyle': '-',
    'boxplot.boxprops.linewidth': 1.0,
    'boxplot.capprops.color': 'white',
    'boxplot.capprops.linestyle': '-',
    'boxplot.capprops.linewidth': 1.0,

```

```
'boxplot.flierprops.color': 'white',
'boxplot.flierprops.linestyle': 'none',
'boxplot.flierprops.linewidth': 1.0,
'boxplot.flierprops.marker': 'o',
'boxplot.flierprops.markeredgecolor': 'white',
'boxplot.flierprops.markeredgewidth': 1.0,
'boxplot.flierprops.markerfacecolor': 'none',
'boxplot.flierprops.markersize': 6.0,
'boxplot.meanline': False,
'boxplot.meanprops.color': 'C2',
'boxplot.meanprops.linestyle': '--',
'boxplot.meanprops.linewidth': 1.0,
'boxplot.meanprops.marker': '^',
'boxplot.meanprops.markeredgecolor': 'C2',
'boxplot.meanprops.markerfacecolor': 'C2',
'boxplot.meanprops.markersize': 6.0,
'boxplot.medianprops.color': 'C1',
'boxplot.medianprops.linestyle': '-',
'boxplot.medianprops.linewidth': 1.0,
'boxplot.notch': False,
'boxplot.patchartist': False,
'boxplot.showbox': True,
'boxplot.showcaps': True,
'boxplot.showfliers': True,
'boxplot.showmeans': False,
'boxplot.vertical': True,
'boxplot.whiskerprops.color': 'white',
'boxplot.whiskerprops.linestyle': '-',
'boxplot.whiskerprops.linewidth': 1.0,
'boxplot.whiskers': 1.5,
'contour.algorithm': 'mpl2014',
'contour.corner_mask': True,
'contour.linewidth': None,
'contour.negative_linestyle': 'dashed',
'date.autoformatter.day': '%Y-%m-%d',
'date.autoformatter.hour': '%m-%d %H',
'date.autoformatter.microsecond': '%M:%S.%f',
'date.autoformatter.minute': '%d %H:%M',
'date.autoformatter.month': '%Y-%m',
'date.autoformatter.second': '%H:%M:%S',
'date.autoformatter.year': '%Y',
'date.converter': 'auto',
'date.epoch': '1970-01-01T00:00:00',
'date.interval_multiples': True,
'docstring.hardcopy': False,
'errorbar.capsize': 0.0,
'figure.autolayout': False,
'figure.constrained_layout.h_pad': 0.04167,
'figure.constrained_layout.hspace': 0.02,
'figure.constrained_layout.use': False,
'figure.constrained_layout.w_pad': 0.04167,
'figure.constrained_layout.wspace': 0.02,
'figure.dpi': 100.0,
'figure.edgecolor': 'black',
'figure.facecolor': 'black',
'figure.figsize': [8.0, 5.5],
'figure.frameon': True,
'figure.hooks': [],
'figure.labelsize': 'large',
'figure.labelweight': 'normal',
```

```
'figure.max_open_warning': 20,  
'figure.raise_window': True,  
'figure.subplot.bottom': 0.11,  
'figure.subplot.hspace': 0.2,  
'figure.subplot.left': 0.125,  
'figure.subplot.right': 0.9,  
'figure.subplot.top': 0.88,  
'figure.subplot.wspace': 0.2,  
'figure.titlesize': 'large',  
'figure.titleweight': 'normal',  
'font.cursive': ['Apple Chancery',  
                 'Textile',  
                 'Zapf Chancery',  
                 'Sand',  
                 'Script MT',  
                 'Felipa',  
                 'Comic Neue',  
                 'Comic Sans MS',  
                 'cursive'],  
'font.family': ['sans-serif'],  
'font.fantasy': ['Chicago',  
                 'Charcoal',  
                 'Impact',  
                 'Western',  
                 'Humor Sans',  
                 'xkcd',  
                 'fantasy'],  
'font.monospace': ['DejaVu Sans Mono',  
                   'Bitstream Vera Sans Mono',  
                   'Computer Modern Typewriter',  
                   'Andale Mono',  
                   'Nimbus Mono L',  
                   'Courier New',  
                   'Courier',  
                   'Fixed',  
                   'Terminal',  
                   'monospace'],  
'font.sans-serif': ['Arial',  
                   'Liberation Sans',  
                   'DejaVu Sans',  
                   'Bitstream Vera Sans',  
                   'sans-serif'],  
'font.serif': ['DejaVu Serif',  
               'Bitstream Vera Serif',  
               'Computer Modern Roman',  
               'New Century Schoolbook',  
               'Century Schoolbook L',  
               'Utopia',  
               'ITC Bookman',  
               'Bookman',  
               'Nimbus Roman No9 L',  
               'Times New Roman',  
               'Times',  
               'Palatino',  
               'Charter',  
               'serif'],  
'font.size': 10.0,  
'font.stretch': 'normal',  
'font.style': 'normal',  
'font.variant': 'normal',
```



```
'font.weight': 'normal',
'grid.alpha': 1.0,
'grid.color': 'white',
'grid.linestyle': '-',
'grid.linewidth': 1.0,
'hatch.color': 'black',
'hatch.linewidth': 1.0,
'hist.bins': 10,
'image.aspect': 'equal',
'image.cmap': 'gray',
'image.composite_image': True,
'image.interpolation': 'antialiased',
'image.lut': 256,
'image.origin': 'upper',
'image.resample': True,
'interactive': True,
'keymap.back': ['left', 'c', 'backspace', 'MouseButton.BACK'],
'keymap.copy': ['ctrl+c', 'cmd+c'],
'keymap.forward': ['right', 'v', 'MouseButton.FORWARD'],
'keymap.fullscreen': ['f', 'ctrl+f'],
'keymap.grid': ['g'],
'keymap.grid_minor': ['G'],
'keymap.help': ['f1'],
'keymap.home': ['h', 'r', 'home'],
'keymap.pan': ['p'],
'keymap.quit': ['ctrl+w', 'cmd+w', 'q'],
'keymap.quit_all': [],
'keymap.save': ['s', 'ctrl+s'],
'keymap.xscale': ['k', 'L'],
'keymap.yscale': ['l'],
'keymap.zoom': ['o'],
'legend.borderaxespad': 0.5,
'legend.borderpad': 0.4,
'legend.columnspacing': 2.0,
'legend.edgecolor': '0.8',
'legend.facecolor': 'inherit',
'legend.fancybox': True,
'legend.fontsize': 10.0,
'legend.framealpha': 0.8,
'legend.frameon': False,
'legend.handleheight': 0.7,
'legend.handlelength': 2.0,
'legend.handletextpad': 0.8,
'legend.labelcolor': 'None',
'legend.labelspacing': 0.5,
'legend.loc': 'best',
'legend.markerscale': 1.0,
'legend.numpoints': 1,
'legend.scatterpoints': 1,
'legend.shadow': False,
'legend.title_fontsize': None,
'lines.antialiased': True,
'lines.color': 'white',
'lines.dash_capstyle': <CapStyle.butt: 'butt'>,
'lines.dash_joinstyle': <JoinStyle.round: 'round'>,
'lines.dashdot_pattern': [6.4, 1.6, 1.0, 1.6],
'lines.dashed_pattern': [3.7, 1.6],
'lines.dotted_pattern': [1.0, 1.65],
'lines.linestyle': '-',
'lines.linewidth': 1.75,
```

```

'lines.marker': 'None',
'lines.markeredgewidth': 0.0,
'lines.markeredgewidth': 0.0,
'lines.markerfacecolor': 'auto',
'lines.markersize': 7.0,
'lines.scale_dashes': True,
'lines.solid_capstyle': <CapStyle.round: 'round'>,
'lines.solid_joinstyle': <JoinStyle.round: 'round'>,
'markers.fillstyle': 'full',
'mathtext.bf': 'sans:bold',
'mathtext.cal': 'cursive',
'mathtext.default': 'it',
'mathtext.fallback': 'cm',
'mathtext.fontset': 'dejavusans',
'mathtext.it': 'sans:italic',
'mathtext.rm': 'sans',
'mathtext.sf': 'sans',
'mathtext.tt': 'monospace',
'patch.antialiased': True,
'patch.edgecolor': 'white',
'patch.facecolor': '#348ABD',
'patch.force_edgecolor': False,
'patch.linewidth': 0.5,
'path.effects': [],
'path.simplify': True,
'path.simplify_threshold': 0.111111111111,
'path.sketch': None,
'path.snap': True,
'pcolor.shading': 'auto',
'pcolormesh.snap': True,
'pdf.compression': 6,
'pdf.fonttype': 3,
'pdf.inheritcolor': False,
'pdf.use14corefonts': False,
'pgf.preamble': '',
'pgf.rcfonts': True,
'pgf.texsystem': 'xelatex',
'polaraxes.grid': True,
'ps.distiller.res': 6000,
'ps.fonttype': 3,
'ps.papersize': 'letter',
'ps.useafm': False,
'ps.usedistiller': None,
'savefig.bbox': None,
'savefig.directory': '~',
'savefig.dpi': 'figure',
'savefig.edgecolor': 'black',
'savefig.facecolor': 'black',
'savefig.format': 'png',
'savefig.orientation': 'portrait',
'savefig.pad_inches': 0.1,
'savefig.transparent': False,
'scatter.edgecolors': 'face',
'scatter.marker': 'o',
'svg.fonttype': 'path',
'svg.hashsalt': None,
'svg.image_inline': True,
'text.antialiased': True,
'text.color': 'white',
'text.hinting': 'force_autohint',

```

```

'text.hinting_factor': 8,
'text.kerning_factor': 0,
'text.latex.preamble': '',
'text.parse_math': True,
'text.usetex': False,
'timezone': 'UTC',
'tk.window_focus': False,
'toolbar': 'toolbar2',
'webagg.address': '127.0.0.1',
'webagg.open_in_browser': True,
'webagg.port': 8988,
'webagg.port_retries': 50,
'xaxis.labellocation': 'center',
'xtick.alignment': 'center',
'xtick.bottom': True,
'xtick.color': 'white',
'xtick.direction': 'out',
'xtick.labelbottom': True,
'xtick.labelcolor': 'inherit',
'xtick.labelsize': 10.0,
'xtick.labeltop': False,
'xtick.major.bottom': True,
'xtick.major.pad': 7.0,
'xtick.major.size': 0.0,
'xtick.major.top': True,
'xtick.major.width': 1.0,
'xtick.minor.bottom': True,
'xtick.minor.pad': 3.4,
'xtick.minor.size': 0.0,
'xtick.minor.top': True,
'xtick.minor.visible': False,
'xtick.minor.width': 0.5,
'xtick.top': False,
'yaxis.labellocation': 'center',
'ytick.alignment': 'center_baseline',
'ytick.color': 'white',
'ytick.direction': 'out',
'ytick.labelcolor': 'inherit',
'ytick.labelleft': True,
'ytick.labelright': False,
'ytick.labelsize': 10.0,
'ytick.left': True,
'ytick.major.left': True,
'ytick.major.pad': 7.0,
'ytick.major.right': True,
'ytick.major.size': 0.0,
'ytick.major.width': 1.0,
'ytick.minor.left': True,
'ytick.minor.pad': 3.4,
'ytick.minor.right': True,
'ytick.minor.size': 0.0,
'ytick.minor.visible': False,
'ytick.minor.width': 0.5,
'ytick.right': False}))

```

There are many parameters you could set for your style sheets.

Set parameters for your style sheet by:

1. creating your own `matplotlibrc` [file](#), or

2. updating values stored in the dictionary-like variable `plt.rcParams`

Let's change the style of our overlaid density lines using the second method

```
In [ ]: from cycler import cycler

# set to the default style sheet
plt.style.use('default')

# You can update single values using keys:

# Set the font style to italic
plt.rcParams['font.style'] = 'italic'

# Update linewidth
plt.rcParams['lines.linewidth'] = 2

# You can also update many values at once using the update() method:

parameters = {

    # Change default figure size
    'figure.figsize': (5, 4),

    # Add horizontal grid lines
    'axes.grid': True,
    'axes.grid.axis': 'y',

    # Update colors for density lines
    'axes.prop_cycle': cycler('color',
                              ['dimgray', 'slategrey', 'darkgray'])
}

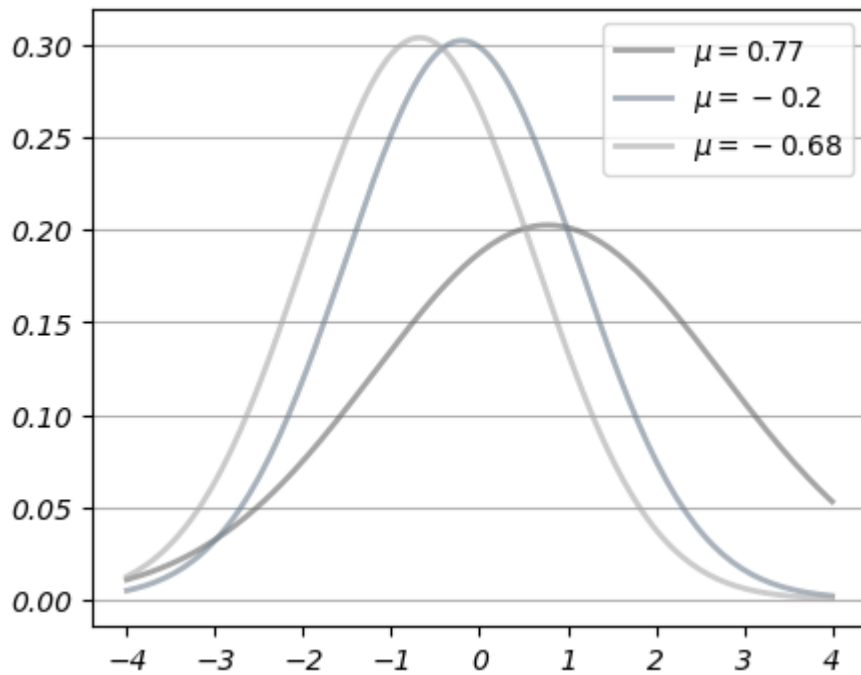
plt.rcParams.update(parameters)
```

#### Note

These settings are `global`.

Any plot generated after changing parameters in `.rcParams` will be affected by the setting.

```
In [ ]: fig, ax = plt.subplots()
x = np.linspace(-4, 4, 150)
for i in range(3):
    m, s = uniform(-1, 1), uniform(1, 2)
    y = norm.pdf(x, loc=m, scale=s)
    current_label = f'$\mu = {m:.2}$'
    ax.plot(x, y, linewidth=2, alpha=0.6, label=current_label)
ax.legend()
plt.show()
```



Apply the `default` style sheet again to change your style back to default

```
In [ ]: plt.style.use('default')

# Reset default figure size
plt.rcParams['figure.figsize'] = (10, 6)
```

## Further Reading

- The [Matplotlib gallery](#) provides many examples.
- A nice [Matplotlib tutorial](#) by Nicolas Rougier, Mike Muller and Gael Varoquaux.
- [mpltools](#) allows easy switching between plot styles.
- [Seaborn](#) facilitates common statistics plots in Matplotlib.

## Exercises

### Exercise 12.1

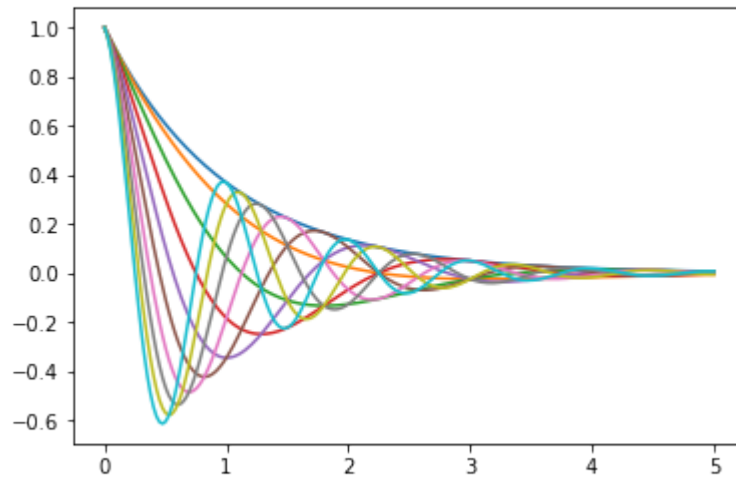
Plot the function

$$f(x) = \cos(\pi\theta x) \exp(-x)$$

over the interval  $[0, 5]$  for each  $\theta$  in `np.linspace(0, 2, 10)`.

Place all the curves in the same figure.

The output should look like this



## Solution to Exercise 12.1

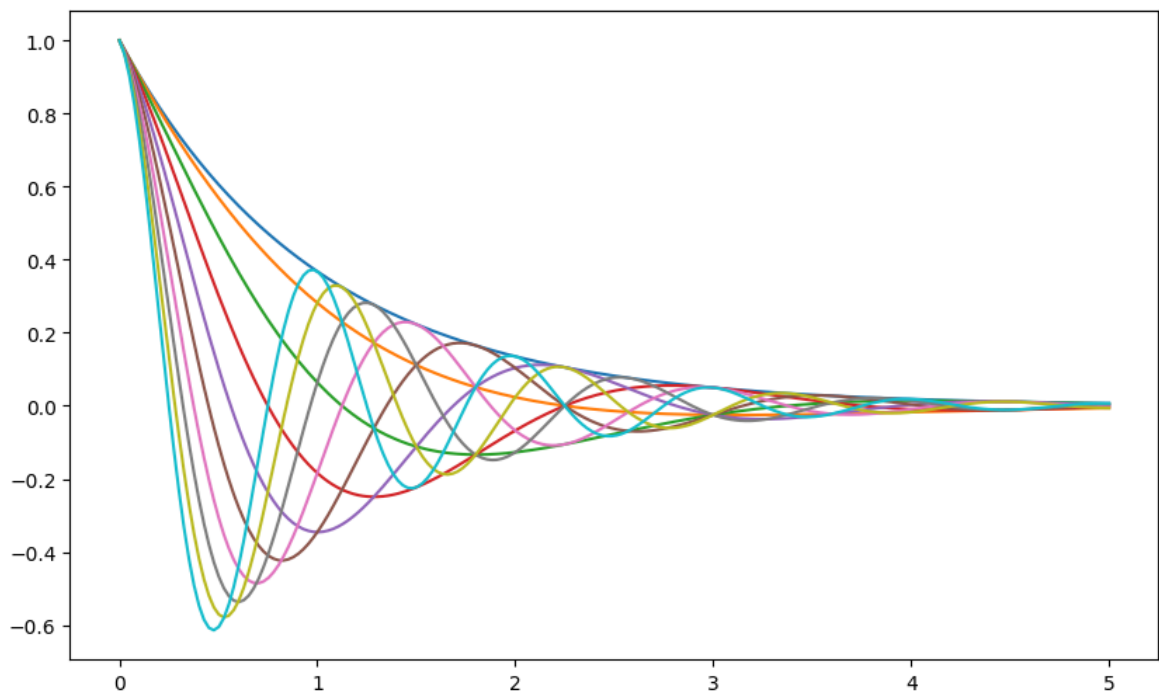
Here's one solution

```
In [ ]: def f(x, theta):
        return np.cos(np.pi * theta * x) * np.exp(-x)

        theta_vals = np.linspace(0, 2, 10)
        x = np.linspace(0, 5, 200)
        fig, ax = plt.subplots()

        for theta in theta_vals:
            ax.plot(x, f(x, theta))

        plt.show()
```



This work is based in part on material under the **Creative Commons Attribution-ShareAlike 4.0 International License**.

- **Original author:** Thomas J. Sargent and John Stachurski

- **Title of work:** Python Programming for Economics and Finance
- **License link:** [Creative Commons Attribution-ShareAlike 4.0 International](#)