Matplotlib v3.4.2 Cheat Sheet

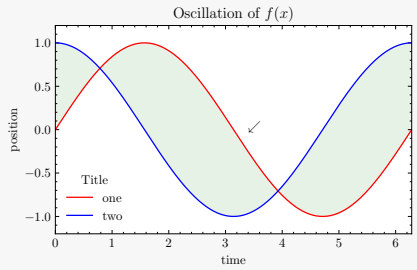
```
import numpy as np; import matplotlib.pyplot as plt; plt.style.use('science')
```

Preparing the Data

```
> t = np.linspace(0, 2*np.pi, 500)
> x, y = np.sin(t), np.cos(t)
> data1 = [10, 20, 40, 80]
> data2 = [12, 18, 48, 66]
> labels = ['A', 'B', 'C', 'D']
> data3 = np.random.randn(300)
> data4 = np.random.randn(300) * 2
> X, Y = np.meshgrid(t, t)
> Z = (np.sin(X) + np.cos(Y))**2
```

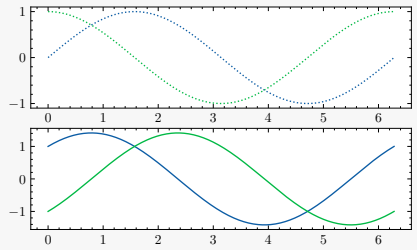
Single 2D Line Plot

```
> fig, ax = plt.subplots(figsize=(5,3))
> ax.set_xlim(t[0], t[-1])
> ax.set_ylim(-1.2, 1.2)
> ax.set_xlabel(r'time')
> ax.set_ylabel(r'position')
> ax.plot(t, x, c='r', label=r'one')
> ax.fill_between(t, x, y, alpha=0.1,
.                   color='g')
> ax.plot(t, y, c='b', label=r'two')
> ax.legend(title='Title')
> ax.set_title(r'Oscillation of $f(x)$')
> ax.annotate(r'$\swarrow$', (3.4,0))
> plt.savefig('figure.pdf')
```



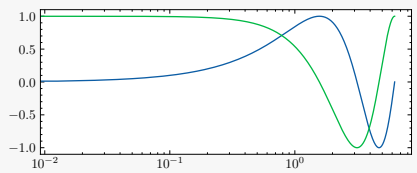
Multiple 2D Plots

```
> fig, axes = plt.subplots(2, 1,
.                          figsize=(5,3))
> ax = axes[0]
> ax.plot(t, x, t, y, linestyle=":")
> ax = axes[1]
> ax.plot(t, x+y, t, x-y)
```

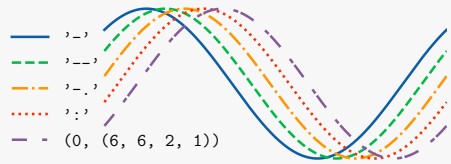


Logarithmic Plots

```
> fig, ax = plt.subplots(figsize=(5,2))
> ax.set_xscale('log')
> ax.plot(t, x, t, y)
```



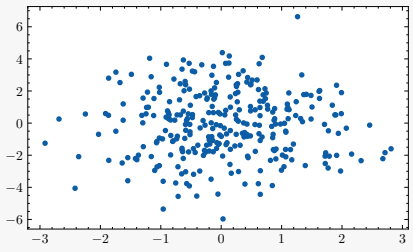
Line Styles



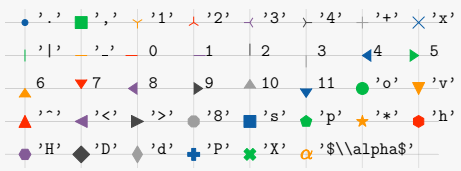
(0, (6, 6, 2, 1)): 0pt offset, 6pt line, 6pt space, 2pt line, 1pt space, and then repeat.

Scatter Plot

```
> fig, ax = plt.subplots(figsize=(5,3))
> ax.scatter(data3, data4, marker='.')
```

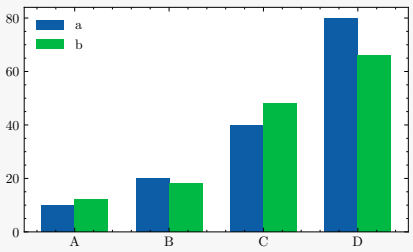


Scatter Markers



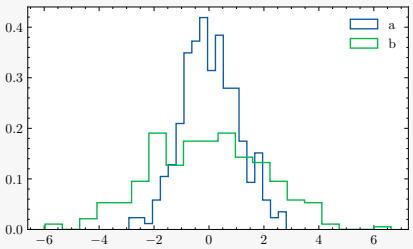
Bar Chart

```
> l, w = np.arange(len(labels)), 0.35
> fig, ax = plt.subplots(figsize=(5,3))
> ax.bar(l - w/2, data1, w, label='a')
> ax.bar(l + w/2, data2, w, label='b')
> ax.set_xticks(l)
> ax.set_xticklabels(labels)
> ax.legend()
```



Histogram

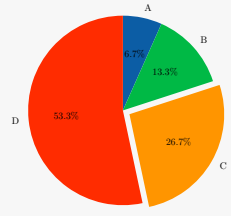
```
> fig, ax = plt.subplots(figsize=(5,3))
> ax.hist(data3, bins=20, density=True,
.         histtype='step', label='a')
> ax.hist(data4, bins=20, density=True,
.         histtype='step', label='b')
> ax.legend()
```



density=True: area under histogram = 1.

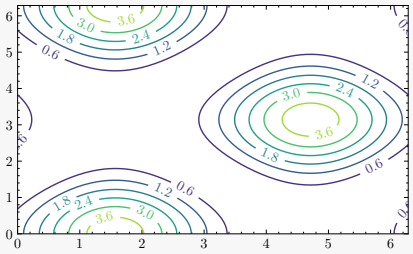
Pie Chart

```
> fig, ax = plt.subplots(figsize=(4,4))
> ax.pie(data1,
.       explode=[0, 0, .08, 0],
.       labels=labels,
.       autopct='%1.1f\%%',
.       startangle = 90,
.       counterclock = False)
> ax.axis('equal')
```



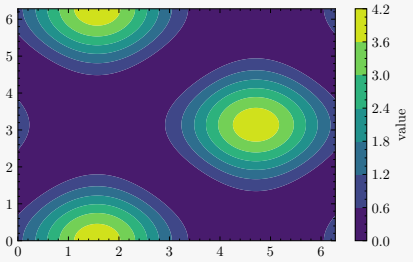
Contour Plot

```
> fig, ax = plt.subplots(figsize=(5,3))
> c = ax.contour(X, Y, Z, levels=6)
> ax.clabel(c, inline=True, fmt='%1.1f')
```



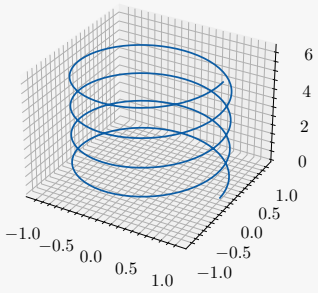
Filled Contour Plot

```
> fig, ax = plt.subplots(figsize=(5,3))
> c = ax.contourf(X, Y, Z, levels=6)
> cbar = fig.colorbar(c)
> cbar.ax.set_ylabel('value')
```



3D Parametric Plot

```
> fig, ax = plt.subplots(figsize=(3,3),
.   subplot_kw={"projection": "3d"})
> ax.plot(np.cos(4*t), np.sin(4*t), t)
> plt.savefig('a.pdf', transparent=True)
```



3D Surface Plot

```
> from matplotlib import cm
> fig, axes = plt.subplots(1, 2,
.       subplot_kw={"projection": "3d"},
.       figsize=(8,4))
> ax = axes[0]
> ax.plot_surface(X, Y, Z,
.                 cmap=cm.coolwarm)
> ax = axes[1]
> ax.plot_surface(X, Y, Z, alpha=0.3)
> ax.contour(X, Y, Z, zdir='x',
.            offset=X.min(),
.            cmap=cm.coolwarm)
> ax.contour(X, Y, Z, zdir='y',
.            offset=Y.max(),
.            cmap=cm.coolwarm)
> ax.contour(X, Y, Z, zdir='z',
.            offset=Z.min(),
.            cmap=cm.coolwarm)
```