Matplotlib Intermediate Cheat Sheet

1. Advanced Plot Types

Visualize more complex data structures.

Heatmaps with ‘imshow‘

Display a 2D matrix as a color-coded grid.

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

data = np.random.rand(10, 10)
fig, ax = plt.subplots()
im = ax.imshow(data, cmap='viridis')

# Add a colorbar to show the scale
fig.colorbar(im)
```

Contour Plots

Display 3D data in 2D by plotting lines of constant value (isolevels).

```
x = np.linspace(-3, 3, 100)
y = np.linspace(-3, 3, 100)
X, Y = np.meshgrid(x, y)
Z = np.sin(X**2 + Y**2)

fig, ax = plt.subplots()
ax.contourf(X, Y, Z, levels=15, cmap='plasma')
```

Error Bars

Show uncertainty in your data points.

```
x = np.arange(5)
y = x**2
y_err = x * 0.5

fig, ax = plt.subplots()
ax.errorbar(x, y, yerr=y_err, fmt='o-', capsize=5)
```

2. Ticks, Spines, & Scales

Fine-grained control over the look of your axes.

Logarithmic Scales

Useful for data spanning several orders of magnitude.

```
fig, ax = plt.subplots()
x = np.linspace(1, 1000, 100)
ax.plot(x, x**2)
ax.set_yscale('log')
ax.set_xscale('log')
```

Controlling Ticks

Change tick locations and labels.

```
from matplotlib.ticker import MultipleLocator,
    FormatStrFormatter

fig, ax = plt.subplots()
ax.plot(range(10))

ax.xaxis.set_major_locator(MultipleLocator(2))
ax.yaxis.set_major_formatter(FormatStrFormatter('%.1
    f'))
```

Manipulating Spines

Spines are the lines connecting the axis tick marks.

```
fig, ax = plt.subplots()
ax.plot(range(5))

ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)
ax.spines['bottom'].set_position(('data', 0))
```

3. Text & Annotations

Add explanatory text directly to your plots.

Adding Text

Place text at arbitrary coordinates.

```
fig, ax = plt.subplots()
ax.axis([0, 10, 0, 10])
ax.text(2, 8, 'My Text',
        fontsize=12,
        bbox=dict(boxstyle='round', facecolor='wheat
    '))
```

Annotating Points

Create an arrow pointing from text to a data point.

```
fig, ax = plt.subplots()
x = np.arange(0, 5)
y = x**2
ax.plot(x, y)

ax.annotate('Maximum value',
            xy=(4, 16),
            xytext=(1, 15),
            arrowprops=dict(facecolor='black',
    shrink=0.05))
```

4. Stylesheets & rcParams

Easily change the overall look and feel of your plots.

Using Stylesheets

Apply a pre-defined style to all subsequent plots.

```
plt.style.use('seaborn-v0_8-whitegrid')
fig, ax = plt.subplots()
ax.plot(np.sin(np.linspace(0, 10)))
```

Customizing with ‘rcParams‘

Modify individual default settings.

```
plt.rcParams['axes.titleweight'] = 'bold'
plt.rcParams['figure.figsize'] = (10, 6)
```

5. Advanced Layouts

Go beyond simple grids with `subplot_mosaic`.

```
fig = plt.figure(tight_layout=True)
ax_dict = fig.subplot_mosaic(
    """
    AAB
    A.C
    """
```

Matplotlib Intermediate Cheat Sheet

```
)
x = np.linspace(0, 10, 100)
ax_dict['A'].plot(x, np.sin(x))
ax_dict['B'].hist(np.random.randn(100))
ax_dict['C'].scatter(x, np.random.rand(100))
```

For more complex programmatic layouts, use `matplotlib.gridspec.GridSpec`.

6. Plotting Directly from Pandas

Pandas DataFrames have a built-in `.plot()` accessor that uses Matplotlib.

Basic Pandas Plotting

```
import pandas as pd
df = pd.DataFrame(np.random.randn(10, 4),
                  columns=['A', 'B', 'C', 'D'])
df.plot(kind='bar')
```

Integrating with OO Workflow

Pass an `Axes` object to the Pandas plot call to have full control.

```
fig, ax = plt.subplots()
df['A'].plot(kind='hist', ax=ax, bins=10)
ax.set_title('Histogram of Column A')
ax.set_xlabel('Value')
```

7. Introduction to 3D Plotting

Matplotlib can create basic 3D plots.

```
from mpl_toolkits.mplot3d import Axes3D

theta = np.linspace(-4 * np.pi, 4 * np.pi, 100)
z = np.linspace(-2, 2, 100)
r = z**2 + 1
x = r * np.sin(theta)
y = r * np.cos(theta)

fig, ax = plt.subplots(subplot_kw={"projection": "3d
    "})
ax.plot(x, y, z, label='3D Spiral')
ax.legend()
plt.show()
```