Module 6: Intro to Jupyter, Matplotlib, and Pandas

October 17, 2025

Content covered:

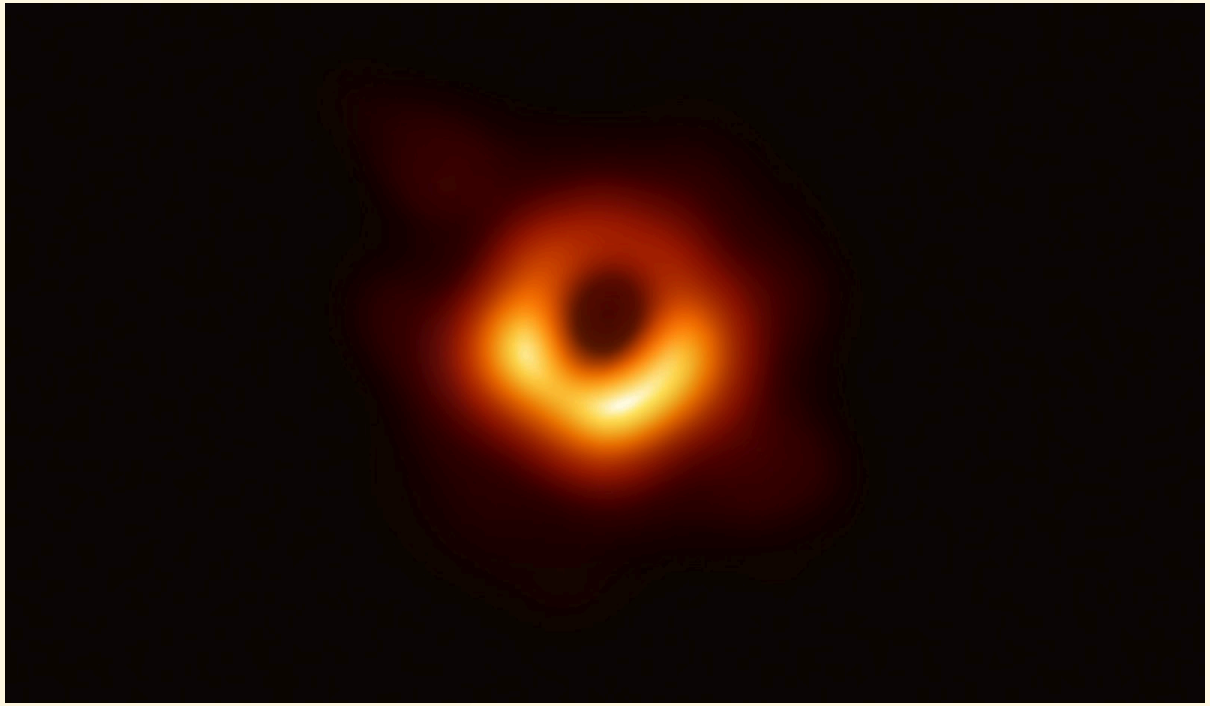
H1

1. Jupyter Lab
2. Intro to Matplotlib
    1. Pyplot
3. Intro to Pandas

Jupyter Notebooks

Matplotlib is the main plotting library in Python.

It's been used to plot generate the first image of a black hole!



The most common submodule in `matplotlib` that you will use is `pyplot`. A submodule is a collection of scripts that belong to a module/package.

A common Python pattern for interacting with this submodule is:

```
import matplotlib.pyplot as plt
```

The keyword `as` works as an alias, which lets you map an import to another variable name.
We saw this when we imported `numpy`:

```
 import numpy as np
```

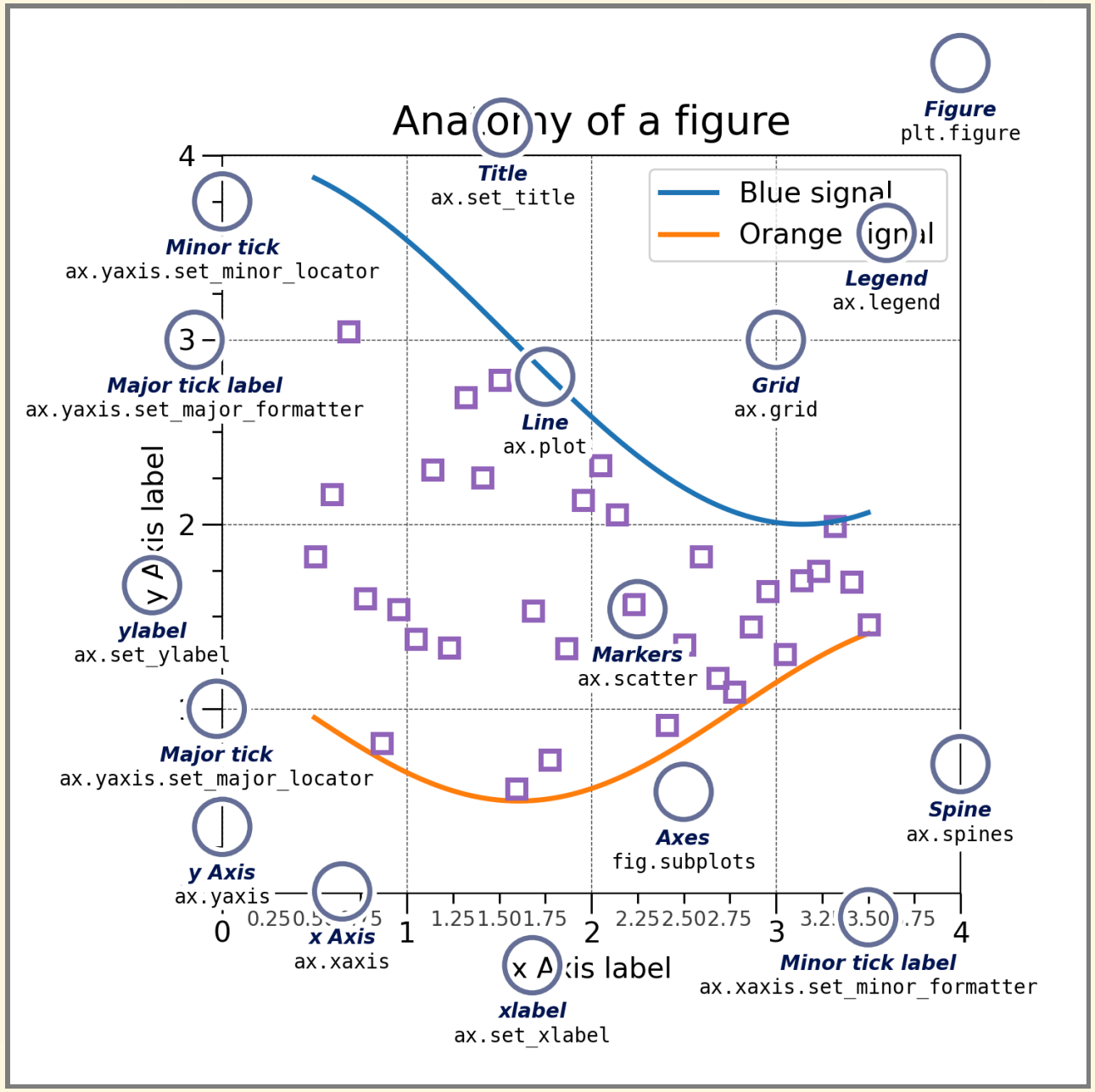
Let's actually bring together what we learned with `numpy` with some new functionality with `matplotlib`.

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

x = np.linspace(0, 2 * np.pi, 200)
y = np.sin(x)

fig, ax = plt.subplots()
ax.plot(x, y)
plt.show()
```

Anatomy of Plotting

```
plt.plot([2,4,6,8])
```

If only a single array is specified, `pyplot` defaults to the input array being the y-values and the x-values will be default to a `range(len(input_array))`, i.e. `[0, 1, 2, 3]`.


Anatomy of a figure
Figure
plt.figure
Title
ax.set_title
Minor tick
ax.yaxis.set_minor_locator
Blue signal
Orange signal
Legend
ax.legend
Major tick label
ax.yaxis.set_major_formatter
Grid
ax.grid
Line
ax.plot
y Axis label
ylabel
ax.set_ylabel
Markers
ax.scatter
Major tick
ax.yaxis.set_major_locator
Spine
ax.spines
y Axis
ax.yaxis
Axes
fig.subplots
x Axis
ax.xaxis
x Axis label
xlabel
ax.set_xlabel
Minor tick label
ax.xaxis.set_minor_formatter

Most Common Elements to Modify

- Axes
  - `.plot()`
  - `.set_title()`
  - `.set_xlabel()`
  - `.set_ylabel()`
  - `.legend()`

Building a Plot, the Object-oriented Way

```
x = np.linspace(0, 2, 100)  # Sample data.

# Note that even in the OO-style, we use `.pyplot.figure` to create the Figure.
fig, ax = plt.subplots(figsize=(5, 2.7), layout='constrained')
ax.plot(x, x, label='linear')  # Plot some data on the Axes.
ax.plot(x, x**2, label='quadratic')  # Plot more data on the Axes...
ax.plot(x, x**3, label='cubic')  # ... and some more.
ax.set_xlabel('x label')  # Add an x-label to the Axes.
ax.set_ylabel('y label')  # Add a y-label to the Axes.
ax.set_title("Simple Plot")  # Add a title to the Axes.
ax.legend()  # Add a legend.
```

There's also a simplier way to plot but it's less explicit using the `pyplot` directly.

You might come across examples using it but for now, we will use the explicit way so that we have as much control over our plots.

Ref: Simple Plot

Generating Data from a Normal Distribution

In previous classes, we used the random number generator to return a value that was randomly sampled from 0 to 1.

Let's use another random generator method to randomly sample from a standard Normal distribution:

```
import numpy as np


rng = np.random.default_rng()
data1, data2, data3, data4 = rng.standard_normal((4, 100))
```

Styling

We can also style the Artists, the output of the plotting function.

Here, we are going to define the color, width, and style of different line plots.

```
fig, ax = plt.subplots(figsize=(5, 2.7))
x = np.arange(len(data1))
ax.plot(x, np.cumsum(data1), color='blue', linewidth=3, linestyle='--')
l, = ax.plot(x, np.cumsum(data2), color='orange', linewidth=2)
l.set_linestyle(':')
```

Other Useful Random Number Functions

There are a host of different distributions you can generate stimulated data from.

- `rng.normal()` lets you set the mean (loc), and the spread, or standard deviation (scale)

Some other distributions to highligh are `beta`, `binomial`, `chisquare`, `f`, `poisson`, `wald`.

There are many more with the random number generator object.

Exercise

Create a figure with two subplots.

Create a lineplot in each of the subplots:

For the lineplot, give it a unique color and linestyle.

Pandas

Pandas is a dataframe library used to load, manipulate, and analyze data.

You can think of it as a way to program a spreadsheet.

Creating our first DataFrame

```
import pandas as pd

my_zoo = {
    'animal': ['panda', 'panda', 'lion', 'tiger', 'lion'],
    'sex': ['f', 'm', 'f', 'f', 'm'],
    'age': [1, 5, 2.5, 3, 3]
}

df = pd.DataFrame(my_zoo)
```

Dataframes are a collection of Series. Series are a collection of values of all the same data type.

You can think of a Series conceptually as a generalized version of a `numpy.array`.

A Series can be a collection of strings, a collection of floats, a collection of DateTime, and a host of other data types.

You can index the contents of a DataFrame in the same way we indexed values from an array.

Instead of using the index for selecting the data, we can use the column name to return the values.

This is similar to how a dictionary works.

These Series have built-in methods to make sense of the data

```
df['age'].max()

df['age'].mean()
```

Group By

If you want to find a breakdown of a given level within your dataframe, you can use the groupby method to specify what column you're interested in grouping by.

This lets you compute summary statistics for the given grouping:

```
df.groupby('animal')['age'].mean()
```

Groupby, cont.

You can also group by multiple columns:

```
df.groupby(['sex', 'age']).count()
```