Object Oriented Programming (Using Python)

UNIT- III

Python Libraries:

- matplotlib
  - Introduction
  - Installation
  - A simple example
  - Parts of a Figure
  - Types of inputs to plotting functions

Prof. R. MADANA MOHANA
Professor, Artificial Intelligence & Data Science
http://rmadanamohana.com/

matplotlib - Quick start guide

https://matplotlib.org/stable/tutorials/introductory/quick_start.html

Matplotlib

- This is a standard data science library that helps to generate data visualizations such as two-dimensional diagrams and graphs (histograms, scatterplots, non-Cartesian coordinates graphs).
- Matplotlib is one of those plotting libraries that are really useful in data science projects - it provides an object-oriented API for embedding plots into applications.
- Python can compete with scientific tools like MatLab or Mathematica.
- However, developers need to write more code than usual while using this library for generating advanced visualizations.
- Note that popular plotting libraries work seamlessly with Matplotlib.

Matplotlib

- Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python.
- Matplotlib makes easy things easy and hard things possible.
  - Create publication quality plots.
  - Make interactive figures that can zoom, pan, update.
  - Customize visual style and layout.
  - Export to many file formats.
  - Embed in JupyterLab and Graphical User Interfaces.
  - Use a rich array of third-party packages built on Matplotlib.

Installing matplotlib

If you already have Python, you can install matplotlib using pip - PyPI (Python Package Index):

```
pip install matplotlib
```

`pip` is a package manager for Python packages, or modules

or

Install using conda:

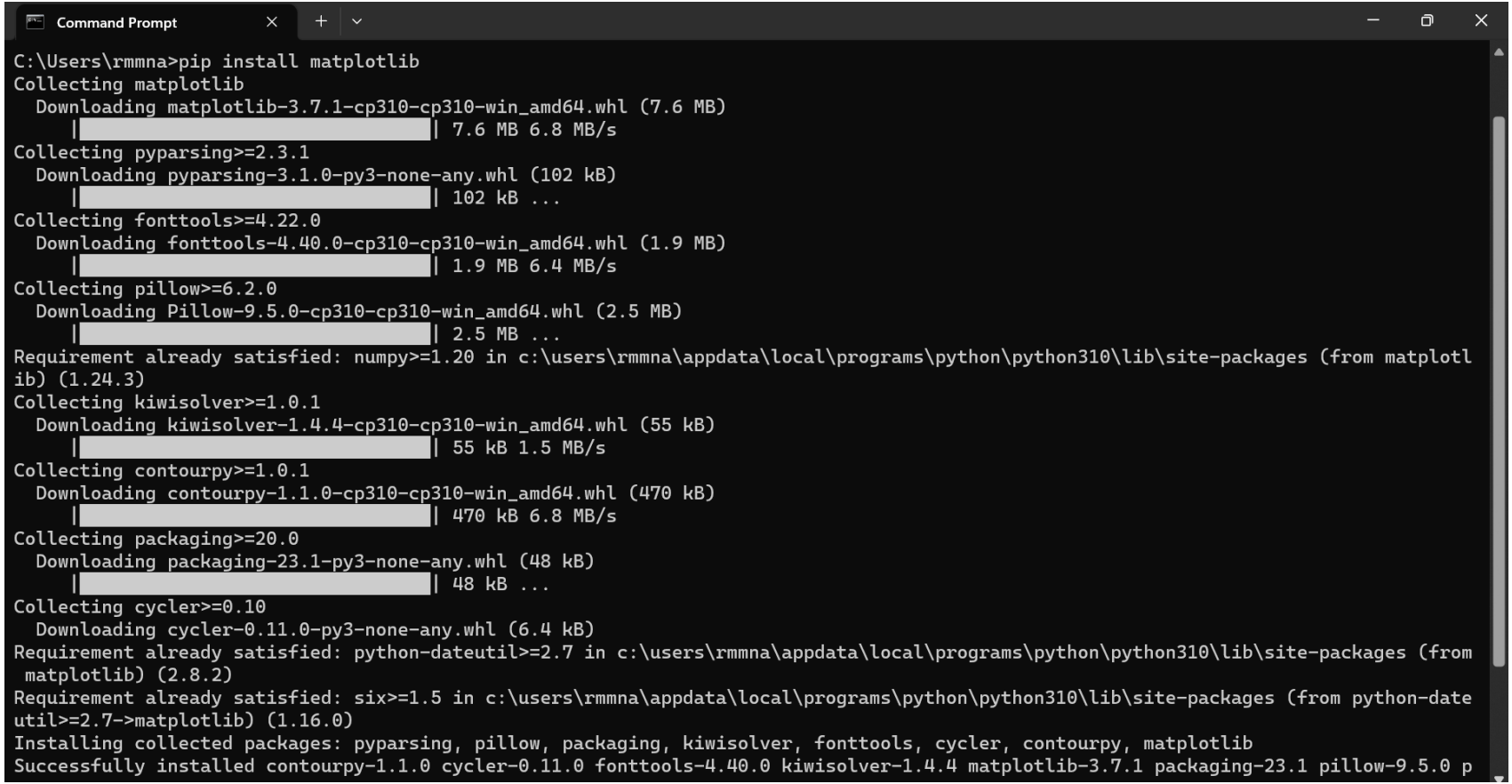
```
conda install -c conda-forge matplotlib
```

`Conda` is a cross platform package and environment manager that installs and manages conda packages from the Anaconda repository as well as from the Anaconda Cloud.

How to install matplotlib using official python IDLE?

- Press the Windows key on your keyboard.
- Type CMD and open Command Prompt. A black terminal should open up.
- Type pip install matplotlib and hit enter.
- It should start the installation. After you see the "Successfully Installed" message, go back to your IDLE and try importing matplotlib, it should work.

How to install matplotlib using official python IDLE?



How to import matplotlib

- To access matplotlib and its functions import it in your Python code like this:

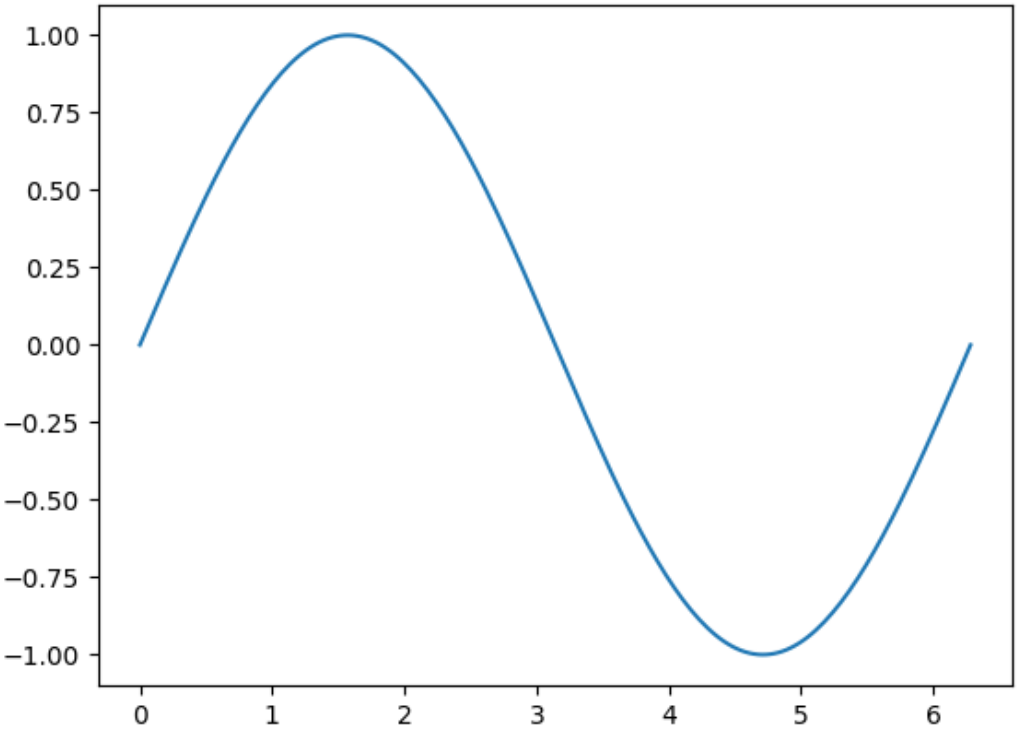
```
import matplotlib as mpl
```

- We shorten the imported name to mpl for better readability of code using matplotlib. This is a widely adopted convention that you should follow so that anyone working with your code can easily understand it.

Draw a first plot

```
>>> 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)
[<matplotlib.lines.Line2D object at 0x000001A0E5021780>]
>>> plt.show()
```

Draw a first plot



Quick start guide

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

Pyplot:

- `matplotlib.pyplot` is a collection of functions that make matplotlib work like MATLAB.
- Each pyplot function makes some change to a figure: e.g., creates a figure, creates a plotting area in a figure, plots some lines in a plotting area, decorates the plot with labels, etc.

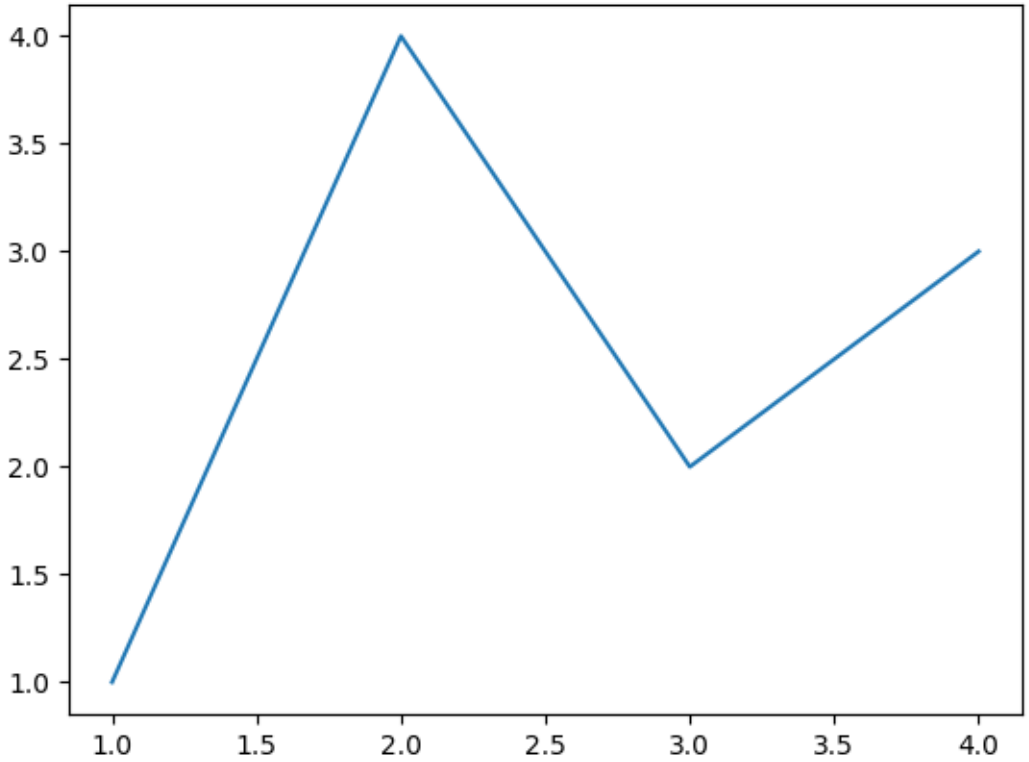
A simple example

- Matplotlib graphs your data on `Figure`s (e.g., windows, Jupyter widgets, etc.), each of which can contain one or more `Axes`, an area where points can be specified in terms of x-y coordinates (or theta-r in a polar plot, x-y-z in a 3D plot, etc.).
- The simplest way of creating a Figure with an Axes is using `pyplot.subplots`.
- We can then use `Axes.plot` to draw some data on the Axes:

A simple example

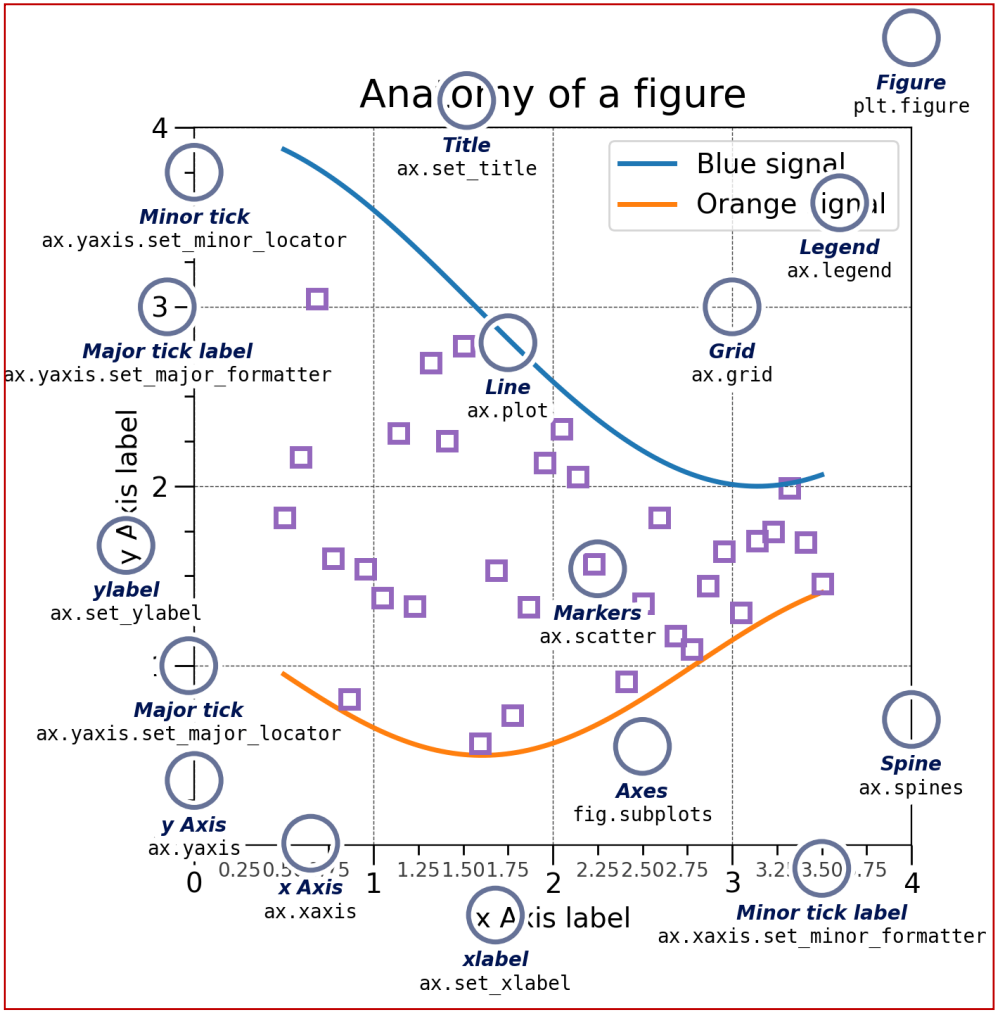
```
>>> fig, ax = plt.subplots() #Create a figure containing a single axes.
>>> ax.plot([1, 2, 3, 4], [1, 4, 2, 3]) # Plot some data on the axes.
[<matplotlib.lines.Line2D object at 0x000002191A694F10>]
>>> plt.show()
```

A simple example



Parts of a Figure

- Here are the components of a Matplotlib Figure.



Parts of a Figure

Figure

- The whole figure.
- The Figure keeps track of all the child `Axes`, a group of 'special' Artists (titles, figure legends, colorbars, etc), and even nested subfigures.
- The easiest way to create a new Figure is with pyplot:

Parts of a Figure

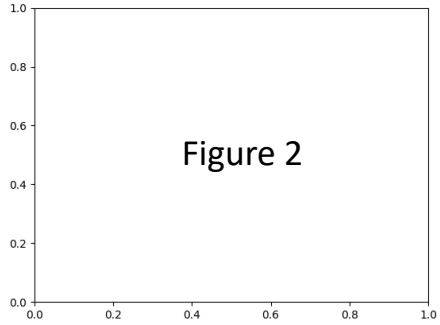
Figure

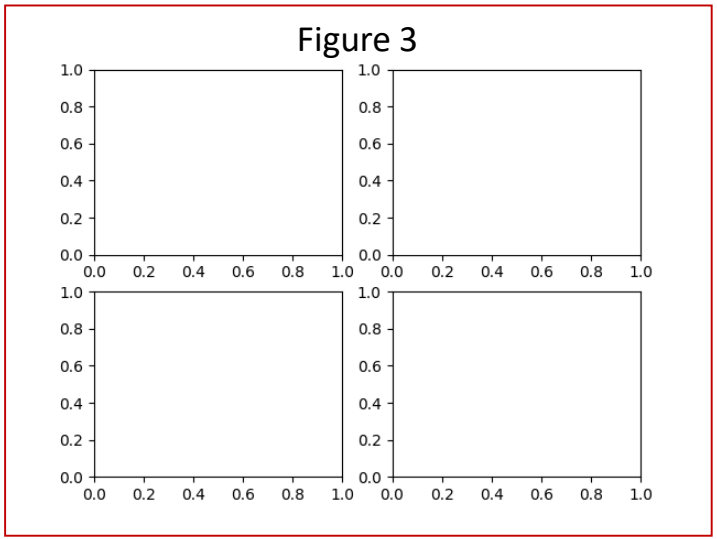
- The easiest way to create a new Figure is with pyplot:

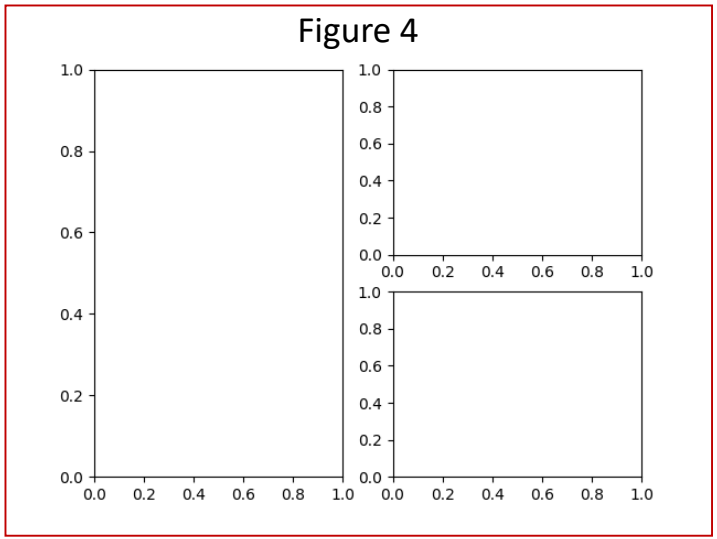
```
>>> fig = plt.figure()  # an empty figure with no Axes
>>> fig, ax = plt.subplots()  # a figure with a single Axes
>>> fig, axs = plt.subplots(2, 2)  # a figure with a 2x2 grid of Axes
# a figure with one axes on the left, and two on the right:
>>> fig, axs = plt.subplot_mosaic([['left', 'right-top'],
                                   ['left', 'right_bottom']])
>>> plt.show()
```

Parts of a Figure

Figure 1







Parts of a Figure

Axes

- An Axes is an Artist attached to a Figure that contains a region for plotting data, and usually includes two (or three in the case of 3D) `Axis` objects (be aware of the difference between Axes and Axis) that provide ticks and tick labels to provide scales for the data in the Axes.
- Each `Axes` also has a title (set via `set_title()`), an x-label (set via `set_xlabel()`), and a y-label set via `set_ylabel()`).
- The `Axes` class and its member functions are the primary entry point to working with the OOP interface, and have most of the plotting methods defined on them (e.g. `ax.plot()`, shown above, uses the `plot` method)

Parts of a Figure

Axis

- These objects set the scale and limits and generate ticks (the marks on the Axis) and ticklabels (strings labeling the ticks).
- The location of the ticks is determined by a `Locator` object and the ticklabel strings are formatted by a `Formatter`.
- The combination of the correct `Locator` and `Formatter` gives very fine control over the tick locations and labels.

Parts of a Figure

Artist

- Basically, everything visible on the Figure is an Artist (even `Figure`, `Axes`, and `Axis` objects).
- This includes `Text` objects, `Line2D` objects, `collections` objects, `Patch` objects, etc.
- When the Figure is rendered, all of the Artists are drawn to the canvas.
- Most Artists are tied to an Axes; such an Artist cannot be shared by multiple Axes, or moved from one to another.

Types of inputs to plotting functions

- Plotting functions expect `numpy.array` or `numpy.ma.masked_array` as input, or objects that can be passed to `numpy.asarray`.
- Classes that are similar to arrays ('array-like') such as `pandas` data objects and `numpy.matrix` may not work as intended.
- Common convention is to convert these to `numpy.array` objects prior to plotting.
- For example, to convert a `numpy.matrix`

Types of inputs to plotting functions

For example, to convert a numpy.matrix

```
>>> import numpy as np
>>> b = np.matrix([[1, 2], [3, 4]])
>>> b_asarray = np.asarray(b)
>>> b_asarray
array([[1, 2],
       [3, 4]])
>>> print(b_asarray)
[[1 2]
 [3 4]]
```

Types of inputs to plotting functions

- Most methods will also parse an addressable object like a *dict*, a `numpy.recarray`, or a `pandas.DataFrame`.
- Matplotlib allows you to provide the `data` keyword argument and generate plots passing the strings corresponding to the *x* and *y* variables.

Types of inputs to plotting functions

```
>>> import numpy as np
>>> import matplotlib.pyplot as plt
>>> np.random.seed(19680801)
>>> data = {'a': np.arange(50),
        'c': np.random.randint(0, 50, 50),
        'd': np.random.randn(50)}
>>> data['b'] = data['a'] + 10 * np.random.randn(50)
>>> data['d'] = np.abs(data['d']) * 100
>>>    fig,    ax    =    plt.subplots(figsize=(5,    2.7),
layout='constrained')
```

Types of inputs to plotting functions

```
>>> ax.scatter('a', 'b', c='c', s='d', data=data)
<matplotlib.collections.PathCollection object at 0x000001C94970E380>
>>> ax.set_xlabel('entry a')
Text(0.5, 0, 'entry a')
>>> ax.set_ylabel('entry b')
Text(0, 0.5, 'entry b')
```

Types of inputs to plotting functions

```
>>> plt.show()
```