ENGR 21:
Computer Engineering Fundamentals

Lecture 10
Thursday, October 02, 2025

More about Arrays in NumPy

Initialize a random 3-D array of size (2,3,4)

```
import numpy as np
np.random.rand(2,3,4)
```

3rd dimension

Length of array in 3rd dim. is 2.

rows

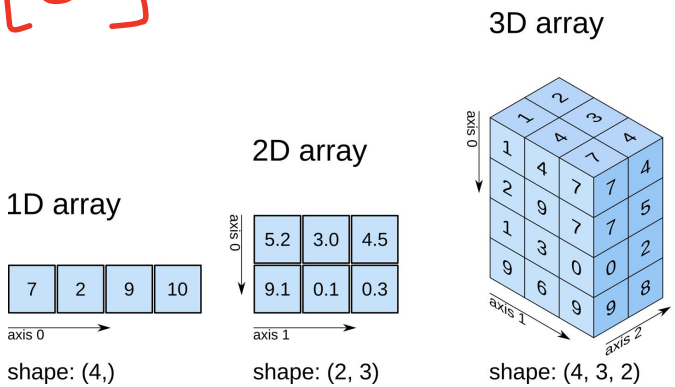
columns

```
>>> np.random.rand(2,3,4)
array([[[0.7873232 , 0.70076764, 0.20354636, 0.54698903],
        [0.62416786, 0.62458573, 0.68440225, 0.30577102],
        [0.84894257, 0.79773177, 0.02707296, 0.91428988]],

       [[0.15647893, 0.4441765 , 0.37088784, 0.14510847],
        [0.36523074, 0.77289487, 0.76270447, 0.69857119],
        [0.50276354, 0.96613687, 0.82106285, 0.17481613]]])
```

a[0,:,:]

a[0]

a[1,:,:] or a[1]



Accessing Arrays

```
np.set_printoptions(precision=3)
```

```
a = array([[[0.746, 0.665, 0.228, 0.552],
            [0.922, 0.323, 0.891, 0.8  ],
            [0.387, 0.176, 0.687, 0.267]],

           [[0.3  , 0.768, 0.928, 0.46 ],
            [0.088, 0.762, 0.471, 0.995],
            [0.385, 0.396, 0.971, 0.677]]])
```

New array with ★ : a[0, :, 2] → 2th column; ↳ all rows

```
a[0][1][2]
a[0][1][:]
a[0][:][1]
a[0,1,:]
a[:,2,:]
a[1]
a[1,:,2]
a[0][2][1:2]
a[0][2][1:]
a[0,2,1:]
a[0,1:,:]
a[0,:1,:]
```

a = `np.random.rand(2,3,4)`

New array with ★ : a[1, 1:, 1]

Introducing `matplotlib`, a plotting library for Python

What is `matplotlib`?

https://matplotlib.org/
https://github.com/matplotlib

- Fully-featured plotting library
  - Gives publication-quality figures, similar to MATLAB
  - Very customizable
- Free & Open Source
- Lives on github
- A package that you must import
  - Does not come pre-installed with Python installation

Conventions for `matplotlib`

For now, stick to this usage:

commonly-used submodule of matplotlib

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

Must use `plt.<function>`

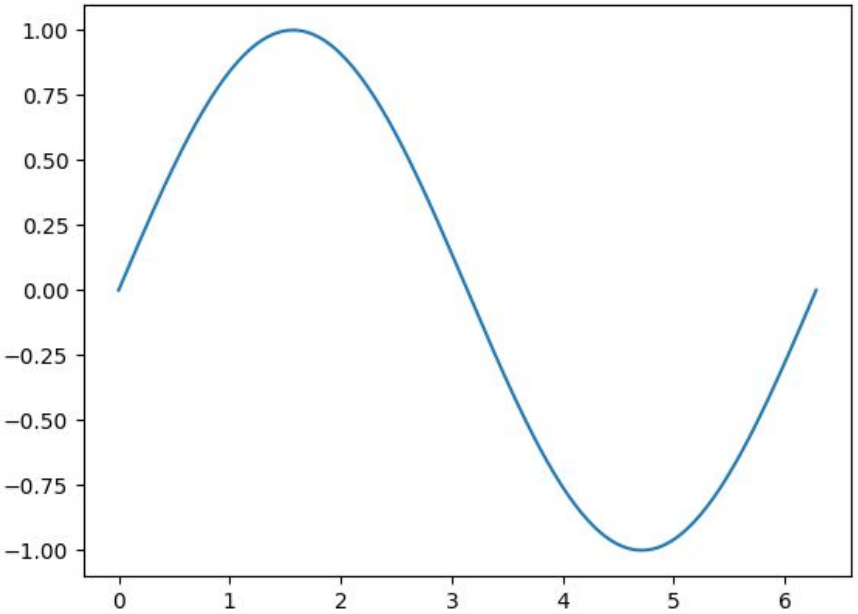
first → pip install matplotlib

Get code from Resources page!

Making your first plot: a sine curve

- Initialize an array of x-coordinates
- Calculate y-coordinates of desired curve

Need to provide (x,y) where x, y should be numpy arrays



Note 'pi' comes from numpy!

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


xvals = np.linspace(0,2*np.pi,200)
yvals = np.sin(xvals)

plt.plot(xvals,yvals)
plt.show()
```

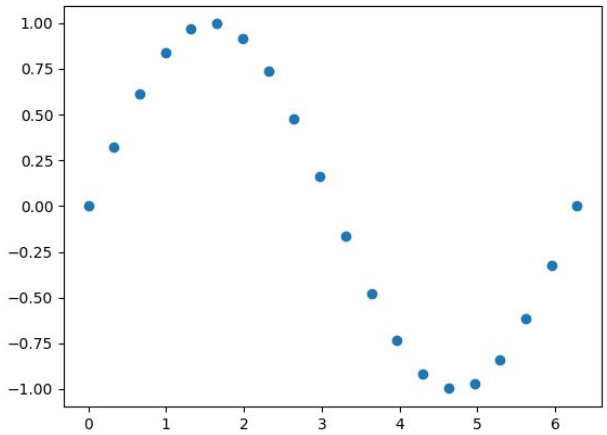
200 linearly spaced values between 0 and 2π

For immediate viewing.

Get code from Resources page!

Lines vs dots

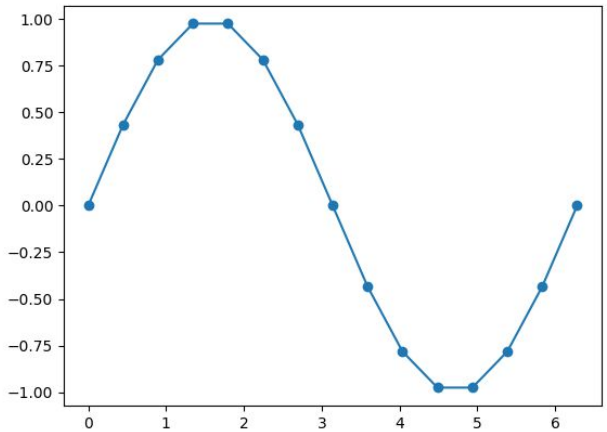
'-' or '--' are line types
'o' or '*' are marker types.




```
# Plot a sine curve
x = np.linspace(0,2*np.pi,20)
y = np.sin(x)

plt.plot(x,y,'o')
plt.show()
```

→ string that describes what plot should look like.

'- -' : dashed




```
# Plot a sine curve
x = np.linspace(0,2*np.pi,15)
y = np.sin(x)

plt.plot(x,y,'o-')
plt.show()
```

Saving figures

Matplotlib can save figures as PNG or PDF, and several other file formats.

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


xvals = np.linspace(0,2*np.pi,200)
yvals = np.sin(xvals)

plt.plot(xvals,yvals)
plt.savefig("test1.pdf")
plt.savefig("test1.png")
plt.savefig("test2.png",dpi=50)
plt.savefig("test3.png",dpi=300)
```

omit plt.show()

dots per inch

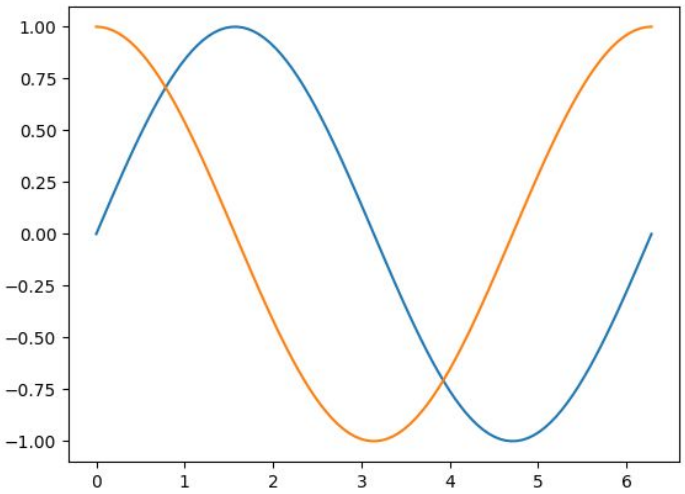
https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.savefig.html

Plot commands for other types of graphs

```
import matplotlib.pyplot
as plt


plt.plot(x, y)

plt.scatter(x, y)

plt.bar(x, y)

plt.hist(data)

plt.contour(X, Y, Z)

plt.pie(sizes)
```

The object-oriented structure of matplotlib

- Plotting commands produce an image
- They also return an object
  - Object has properties and methods
  - Different plotting commands may produce objects of different classes




```
>>> x = np.linspace(0,2*np.pi,100)
>>> c=plt.plot(x,np.sin(x),y,np.cos(y))

>>> type(c)
<class 'list'>

>>> len(c)
2

>>> type(c[0])
<class 'matplotlib.lines.Line2D'>
```

c = ... → gives an object 'c'.

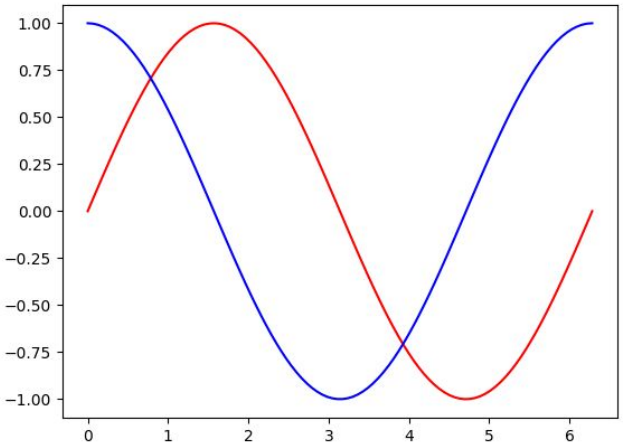
without assignment, just makes a plot.

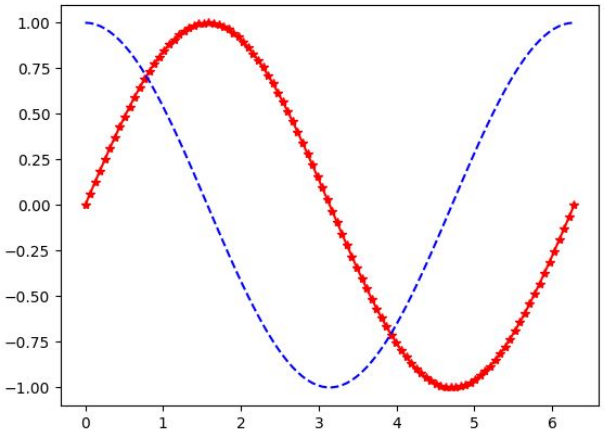
class provided by matplotlib

↳ useful methods.

https://matplotlib.org/stable/api/_as_gen/matplotlib.lines.Line2D.html

The object-oriented structure of matplotlib






```
>>> x = np.linspace(0,2*np.pi,100)

>>> c= plt.plot(x,np.sin(x),y,np.cos(y))

>>> type(c)
<class 'list'>

>>> len(c)
2

>>> type(c[0])
<class 'matplotlib.lines.Line2D'>


>>> c[0].set_marker('*')
>>> c[1].set_linestyle('--')
```

→ one of the available markers

Task using NumPy and Matplotlib

Plot `sin(nx)` for all even **n** between 1 and 10 inclusive, with `0 <= x <= 2*pi`

You must do this by first creating a numpy array of size N x (number of plots) that contains the y-coordinates of your data. Your data will all share x-coordinates.

The data should then be saved to a comma-delimited file using numpy.savetxt using two different conventions:

1. Each plotted series is a column in your data file
2. Each plotted series is a row in your data file.