ANL251 Python Programming

Study Unit 5

Scientific Computing and Plotting with Python

Learning Outcomes and Learning Resources

1. Use NumPy arrays and vectorised operations
   - SU5 Chapter 1.1
   - https://www.numpy.org/devdocs/user/quickstart.html#array-creation
   - https://www.numpy.org/devdocs/user/quickstart.html#basic-operations
   - https://www.numpy.org/devdocs/user/basics.broadcasting.html
2. Implement subsetting NumPy arrays using index or Boolean mask
   - SU5 Chapters 1.2 and 1.3
   - https://www.numpy.org/devdocs/user/quickstart.html#indexing-slicing-and-iterating
   - https://www.numpy.org/devdocs/user/quickstart.html#indexing-with-boolean-arrays
3. Explain the basics of NumPy array attributes
   - SU5 Chapter 1.3
   - https://www.numpy.org/devdocs/user/quickstart.html#the-basics

Learning Outcomes and Learning Resources

4. Apply NumPy functions for statistics and random sampling
   - SU5 Chapter 1.4
   - https://www.numpy.org/devdocs/reference/routines.statistics.html#averages-and-variances
   - https://www.numpy.org/devdocs/reference/routines.random.html#distributions

5. Create basic plots and choose appropriate matplotlib.pyplot functions for customisation
   - SU5 Chapter 2
   - https://www.youtube.com/watch?v=aCULcv_IQYw
   - https://matplotlib.org/api/_as_gen/matplotlib.pyplot.html#module-matplotlib.pyplot

Seminars: discussion and activities to reinforce students' understanding

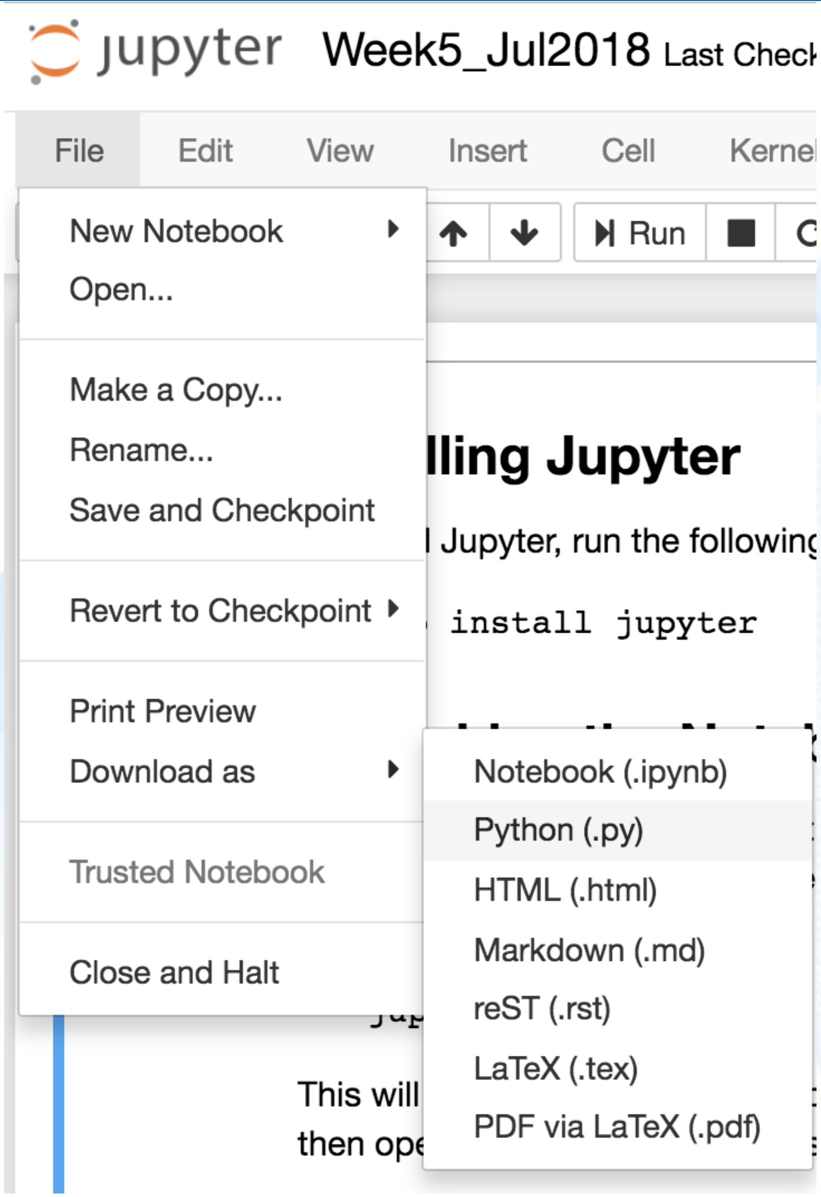
Jupyter Notebook

What are Jupyter Notebooks?

- Typically you'd be coding in the Python shell or an editor like Atom. Your visualizations would then be in separate windows.
- The notebook is a web application that allows you to combine explanatory text, math equations, code, and visualizations all in one easily sharable document.
- Notebooks have quickly become **an essential tool when working with data**.
- The coding in Weeks 5 and 6 will be done in Jupyter notebooks.

Important Note:

You may explore datasets in a notebook but do remember to download and save your work as a **Python .py file for your ECA submission**.



Installing Jupyter

To install Jupyter, run the following command at the Terminal (Mac/Linux) or Command Prompt (Windows):

```
pip install jupyter
```

Launching Notebook Server

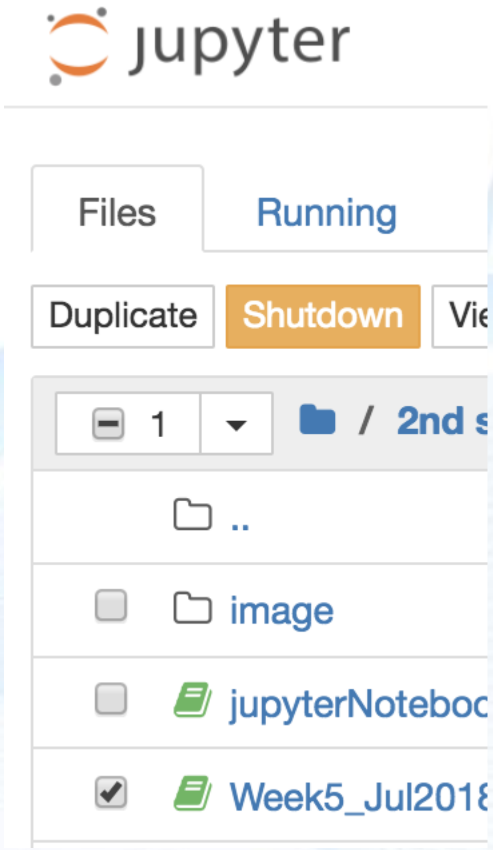
To start the notebook server, run the following command at the Terminal (Mac/Linux) or Command Prompt (Windows) in the directory where your notebooks are saved.

```
jupyter notebook
```

- This will start the server in the directory you ran the command in.
- This will print some information about the notebook server in your terminal, including the URL of the web application (by default, http://localhost:8888). It will then open your default web browser to this URL.

Shutting Down Notebook Server

- Make sure you've saved your work before you shut down it.
- You can shutdown **individual notebooks** by marking the checkbox next to the notebook on the server home and clicking "Shutdown."
- You can shutdown the **entire server** by pressing control + C twice in the terminal. This will immediately shutdown all the running notebooks.



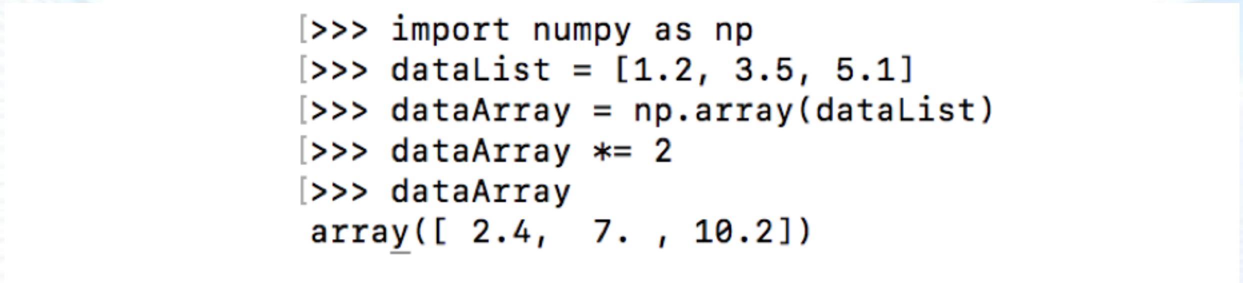
1. NumPy Array and Vectorised Operations

2. Subsetting NumPy Array

3. NumPy Array Attributes

Recap

NumPy arrays and vectorised operations (SU5 Chapter 1.1, https://www.numpy.org/devdocs/user/quickstart.html#array-creation, https://www.numpy.org/devdocs/user/quickstart.html#basic-operations)



```
>>> import numpy as np
>>> dataList = [1.2, 3.5, 5.1]
>>> dataArray = np.array(dataList)
>>> dataArray *= 2
>>> dataArray
array([ 2.4,  7. , 10.2])
```

Figure 5.1 Applying arithmetic operators on arrays elementwise

Note:

- Open the command line and execute *pip install numpy* if you haven't installed the NumPy library.
- NumPy provides functions and operations on entire arrays of data without having to write loops, whereas lists go through each element using loops.

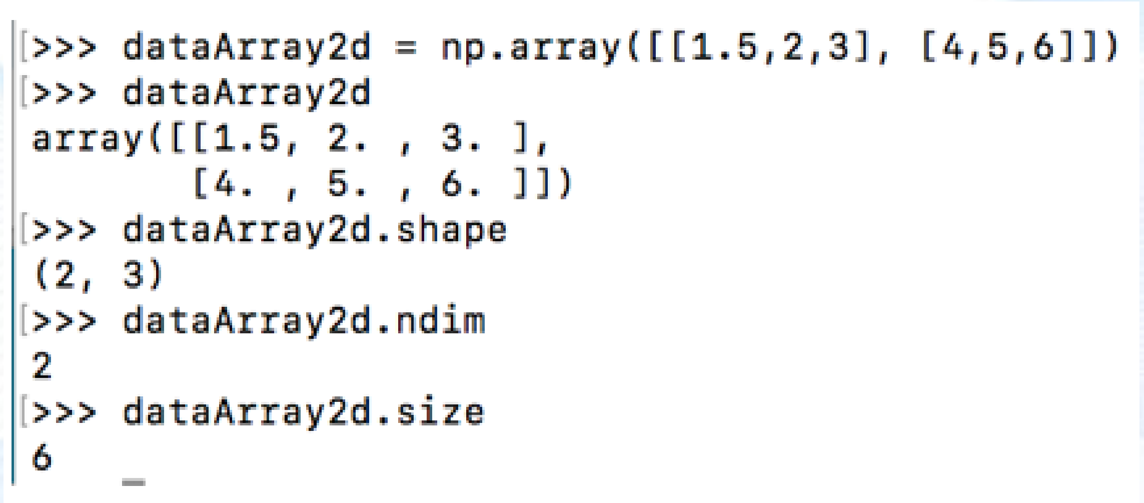
Discussion

Using Numpy's arrays can be 10 times faster than Python's lists. To enable the speed,

- Numpy arrays are fixed in size (but can change the values), unlike lists which can change in size.
- Numpy arrays must all be the same type whereas lists can hold any type. By restricting ndarrays in this way, it makes ndarrays both much more space efficient than lists, but also opens up a range of memory and computational optimizations.

Recap

NumPy array attributes (SU5 Chapter 1.3, https://www.numpy.org/devdocs/user/quickstart.html#the-basics)



```
>>> dataArray2d = np.array([[1.5,2,3], [4,5,6]])
>>> dataArray2d
array([[1.5, 2. , 3. ],
       [4. , 5. , 6. ]])
>>> dataArray2d.shape
(2, 3)
>>> dataArray2d.ndim
2
>>> dataArray2d.size
6
```

Figure 5.3 2-D Numpy Array and its attributes

Quiz

Which of the following is correct to create a 2D array?

```
np.array([[11,12,13],[21,22,23]])

np.array([11,12,13],[21,22,23])

np.array([11,12,13,21,22,23])

np.array[[11,12,13,21,22,23]]
```

Quiz

Do the two NumPy arrays have the same shape?

```
first_array = np.array([3, 33, 333])

second_array = np.array([[3, 33, 333]])
```

Discussion

Other ways to create NumPy arrays?(https://www.numpy.org/devdocs/user/quickstart.html#array-creation)

Write code to create each of the following NumPy arrays.

```
array([[ 0.,  0.,  0.,  0.],
       [ 0.,  0.,  0.,  0.],
       [ 0.,  0.,  0.,  0.]])
```

```
array([10, 15, 20, 25])
```

```
array([ 0.  ,  0.25,  0.5 ,  0.75,  1.  ,  1.25,  1.5 ,  1.75,  2.  ])
```

Recap

Basic operations on NumPy arrays (https://www.numpy.org/devdocs/user/quickstart.html#basic-operations) and broadcasting (https://www.numpy.org/devdocs/user/basics.broadcasting.html)

General Broadcasting Rules

When operating on two arrays, NumPy compares their shapes element-wise. It starts with the trailing dimensions, and works its way forward. Two dimensions are compatible when

- they are equal, or
- one of them is 1

Discussion

```
a = np.array( [20,30,40,50] )
b = np.arange( 4 )
```

What does each of the following evaluate to?

```
a-b
b**2
a<35
```

Discussion

```
a = np.array([[0,0],[0,0]])
b1 = np.array([1,1])
b2 = 1
```

Do print(a+b1) and print(a+b2) result in the same matrix?

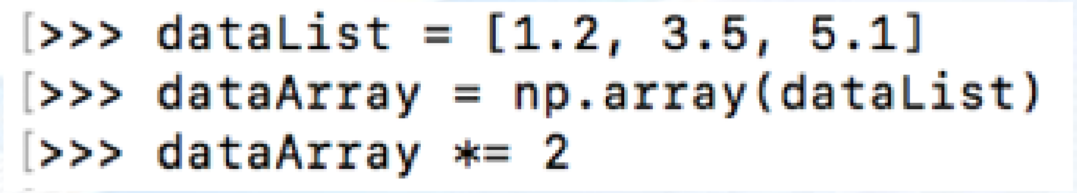
Discussion

What’s printed after executing the code below?

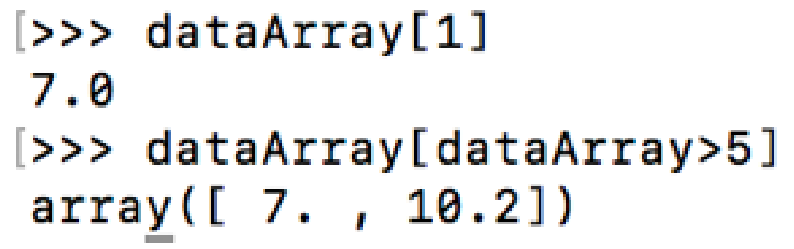
```
arrA = np.array([[1,2,3,4],[5,6,7,8],[9,10,11,12]])
arrB = [0,1,0,2]
print(arrA + arrB)
```

Recap

Subsetting NumPy arrays (SU5 Chapters 1.2 and 1.3, https://www.numpy.org/devdocs/user/quickstart.html#indexing-slicing-and-iterating, https://www.numpy.org/devdocs/user/quickstart.html#indexing-with-boolean-arrays)



```
>>> dataList = [1.2, 3.5, 5.1]
>>> dataArray = np.array(dataList)
>>> dataArray *= 2
```



```
>>> dataArray[1]
7.0
>>> dataArray[dataArray>5]
array([ 7. , 10.2])
```

Figure 5.2 Subsetting NumPy array using index or Boolean masking

Quiz

After the code below has been executed, what value does a[i] refer to?

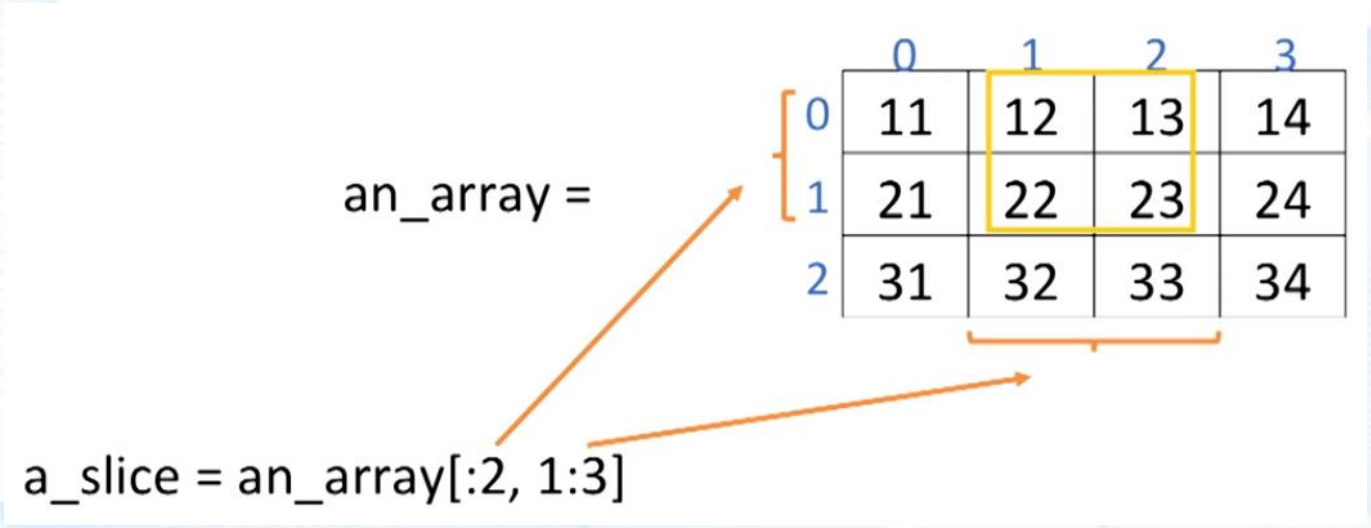
```
a = np.arange(12)**2
i = np.array( [ 1,1,3,8,5 ] )
print(a[i])
```

Then execute the code below. What will happen?

```
a[i] = ‘OK’
```

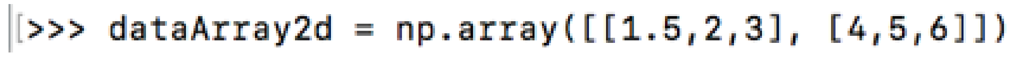
Recap

Subsetting NumPy 2D arrays

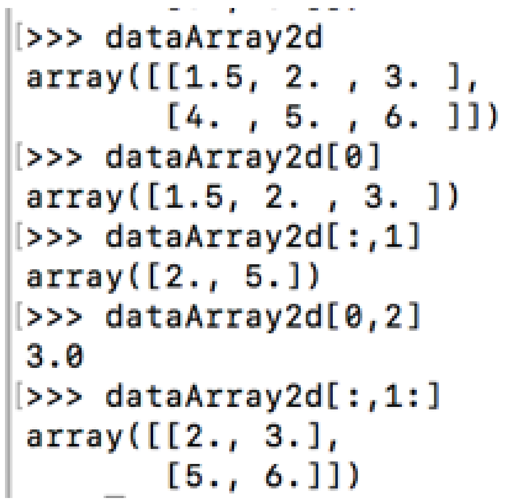


$$\begin{array}{c|cccc} & 0 & 1 & 2 & 3 \\ \hline 0 & 11 & 12 & 13 & 14 \\ 1 & 21 & 22 & 23 & 24 \\ 2 & 31 & 32 & 33 & 34 \end{array}$$

Recap



```
>>> dataArray2d = np.array([[1.5,2,3], [4,5,6]])
```



```
>>> dataArray2d
array([[1.5, 2. , 3. ],
       [4. , 5. , 6. ]])
>>> dataArray2d[0]
array([1.5, 2. , 3. ])
>>> dataArray2d[:,1]
array([2., 5.])
>>> dataArray2d[0,2]
3.0
>>> dataArray2d[:,1:]
array([[2., 3.],
       [5., 6.]])
```

Figure 5.4 Subsetting 2-D Numpy array

Quiz

Given a 3-by-3 NumPy array called “arr”, how many rows will be returned by calling arr[:2,]?

2

3

IndexError: index out of bounds

1

Quiz

```
an_array = np.array([[11,12,13,14], [21,22,23,24], [31,32,33,34]])
```

What does each of the following evaluate to?

```
an_array.shape
an_array[1, :]
an_array[1:2, :]
an_array[:, 1]
an_array[:, 1:2]
```

Quiz

an_array = np.array([[11,12,13,14], [21,22,23,24], [31,32,33,34]])

Write code to print

elements greater than 15

elements greater than 15 and less than 30

elements which are even

Discussion

```
an_array = np.array([[11,12,13,14], [21,22,23,24], [31,32,33,34]])
```

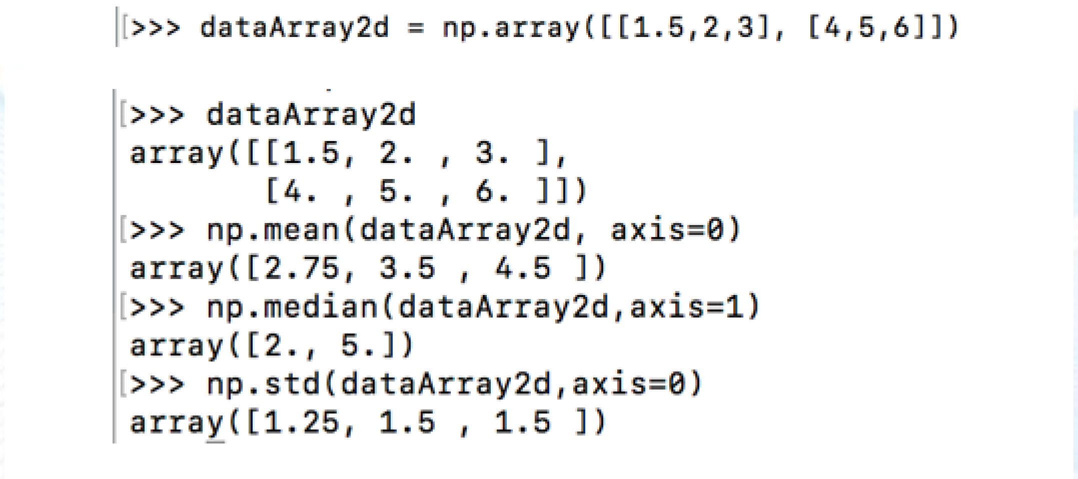
Write code to change the elements in an_array to

```
[[100011 12 13] [ 21 100022 23] [ 31 32 100033] [100041 42 43]]
```

4. NumPy Functions for Statistics and Random Sampling

Recap

NumPy functions (SU5 Chapter 1.4, https://www.numpy.org/devdocs/reference/routines.statistics.html#averages-and-variances)



```
>>> dataArray2d = np.array([[1.5,2,3], [4,5,6]])

>>> dataArray2d
array([[1.5, 2. , 3. ],
       [4. , 5. , 6. ]])
>>> np.mean(dataArray2d, axis=0)
array([2.75, 3.5 , 4.5 ])
>>> np.median(dataArray2d,axis=1)
array([2., 5.])
>>> np.std(dataArray2d,axis=0)
array([1.25, 1.5 , 1.5 ])
```

Figure 5.5 Calculating summary statistics with 2-D Numpy array

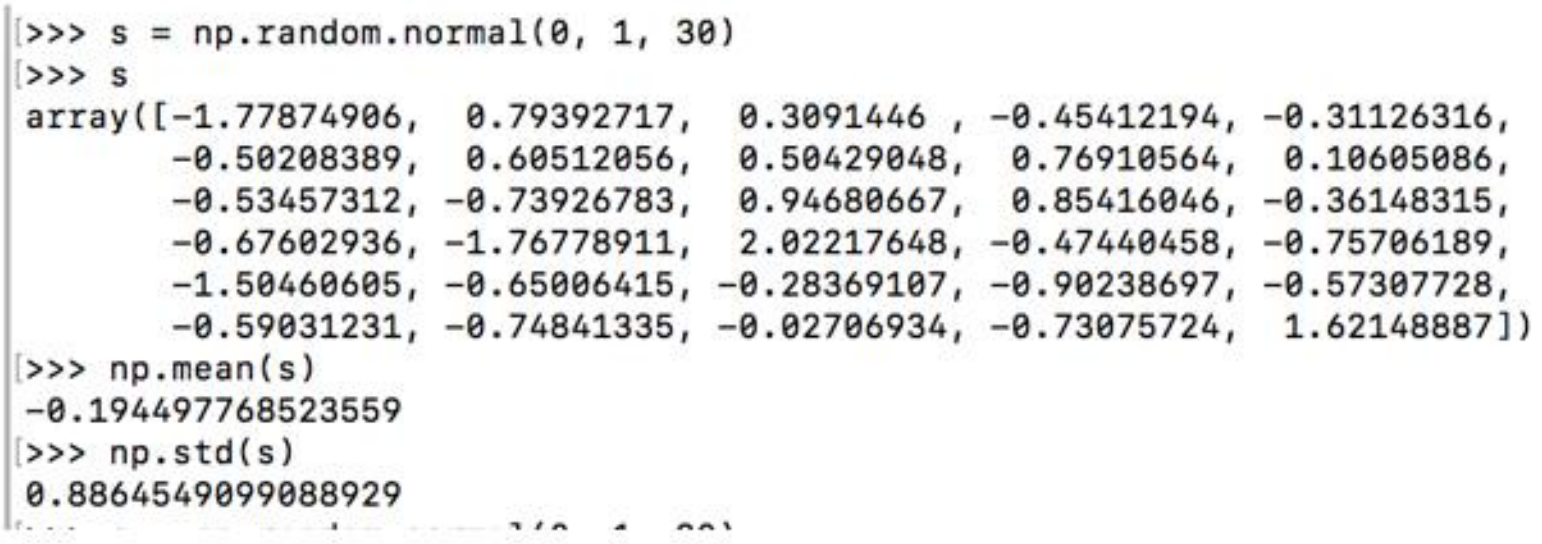
Note:

– A value of 0 for the axis parameter means the calculation is done along the first dimension/axis, namely along the rows.

– A value of 1 for the axis parameter means the calculation is done along the second dimension/axis, namely along the columns.

Recap

NumPy functions (SU5 Chapter 1.4, https://www.numpy.org/devdocs/reference/routines.random.html#distributions)



```
>>> s = np.random.normal(0, 1, 30)
>>> s
array([-1.77874906,  0.79392717,  0.3091446 , -0.45412194, -0.31126316,
       -0.50208389,  0.60512056,  0.50429048,  0.76910564,  0.10605086,
       -0.53457312, -0.73926783,  0.94680667,  0.85416046, -0.36148315,
       -0.67602936, -1.76778911,  2.02217648, -0.47440458, -0.75706189,
       -1.50460605, -0.65006415, -0.28369107, -0.90238697, -0.57307728,
       -0.59031231, -0.74841335, -0.02706934, -0.73075724,  1.62148887])
>>> np.mean(s)
-0.194497768523559
>>> np.std(s)
0.8864549099088929
```

Figure 5.6 Generating 30 random samples from standard normal distribution

Recap

- When we need to do a rather standard task, **functions** are one way we may look for.
- But some basic tasks instead are implemented as **objects**’ **methods**. (SU4)

Discussion

Working with NumPy arrays' methods (https://www.numpy.org/devdocs/reference/arrays.ndarray.html#array-methods).

Many operations are implemented as methods of the NumPy array objects.

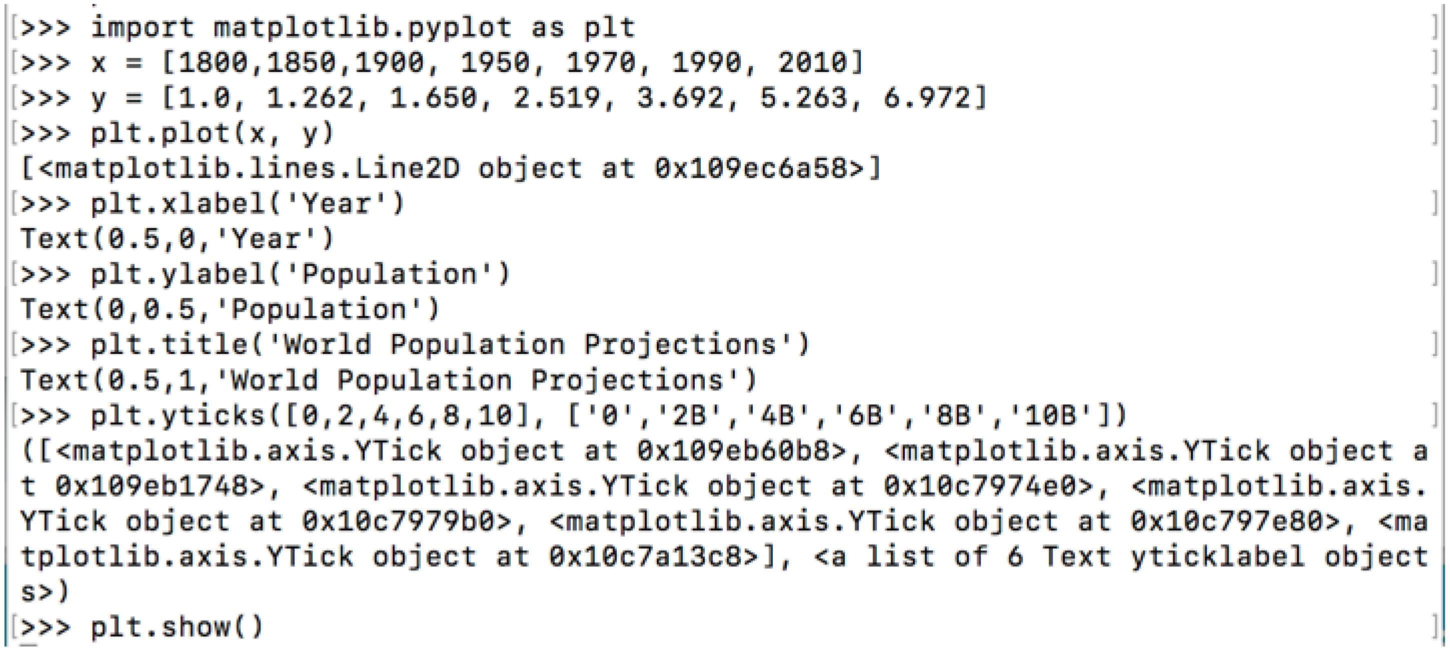
```
b = np.arange(12).reshape(3,4)
b.sum(axis=0)
b.min(axis=1)
b.cumsum(axis=1)
```

5. Plotting with matplotlib

Recap

matplotlib (SU5 Chapter 2, https://www.youtube.com/watch?v=aCULcv_IQYw, https://matplotlib.org/api/_as_gen/matplotlib.pyplot.html#module-matplotlib.pyplot)

Line plot



```
>>> import matplotlib.pyplot as plt
>>> x = [1800,1850,1900, 1950, 1970, 1990, 2010]
>>> y = [1.0, 1.262, 1.650, 2.519, 3.692, 5.263, 6.972]
>>> plt.plot(x, y)
[<matplotlib.lines.Line2D object at 0x109ec6a58>]
>>> plt.xlabel('Year')
Text(0.5,0,'Year')
>>> plt.ylabel('Population')
Text(0,0.5,'Population')
>>> plt.title('World Population Projections')
Text(0.5,1,'World Population Projections')
>>> plt.yticks([0,2,4,6,8,10], ['0','2B','4B','6B','8B','10B'])
([<matplotlib.axis.YTick object at 0x109eb60b8>, <matplotlib.axis.YTick object a
t 0x109eb1748>, <matplotlib.axis.YTick object at 0x10c7974e0>, <matplotlib.axis.
YTick object at 0x10c7979b0>, <matplotlib.axis.YTick object at 0x10c797e80>, <ma
tplotlib.axis.YTick object at 0x10c7a13c8>], <a list of 6 Text yticklabel object
s>)
>>> plt.show()
```

Figure 5.7 Building a line plot with customised labels, title and axis ticks

Recap

- Each matplotlib.pyplot function makes some change to a figure: e.g., creating a figure, creating a plotting area in a figure, plotting some lines in a plotting area, decorating the plot with labels, etc.
- Various states are preserved across function calls, so that it keeps track of things like the current figure and plotting area, and then the plotting functions are directed to the current part of a figure.

- https://matplotlib.org/api/_as_gen/matplotlib.pyplot.plot.html#matplotlib.pyplot.plot
- https://matplotlib.org/api/_as_gen/matplotlib.pyplot.xticks.html#matplotlib.pyplot.xticks
- https://matplotlib.org/api/_as_gen/matplotlib.pyplot.yticks.html#matplotlib.pyplot.yticks

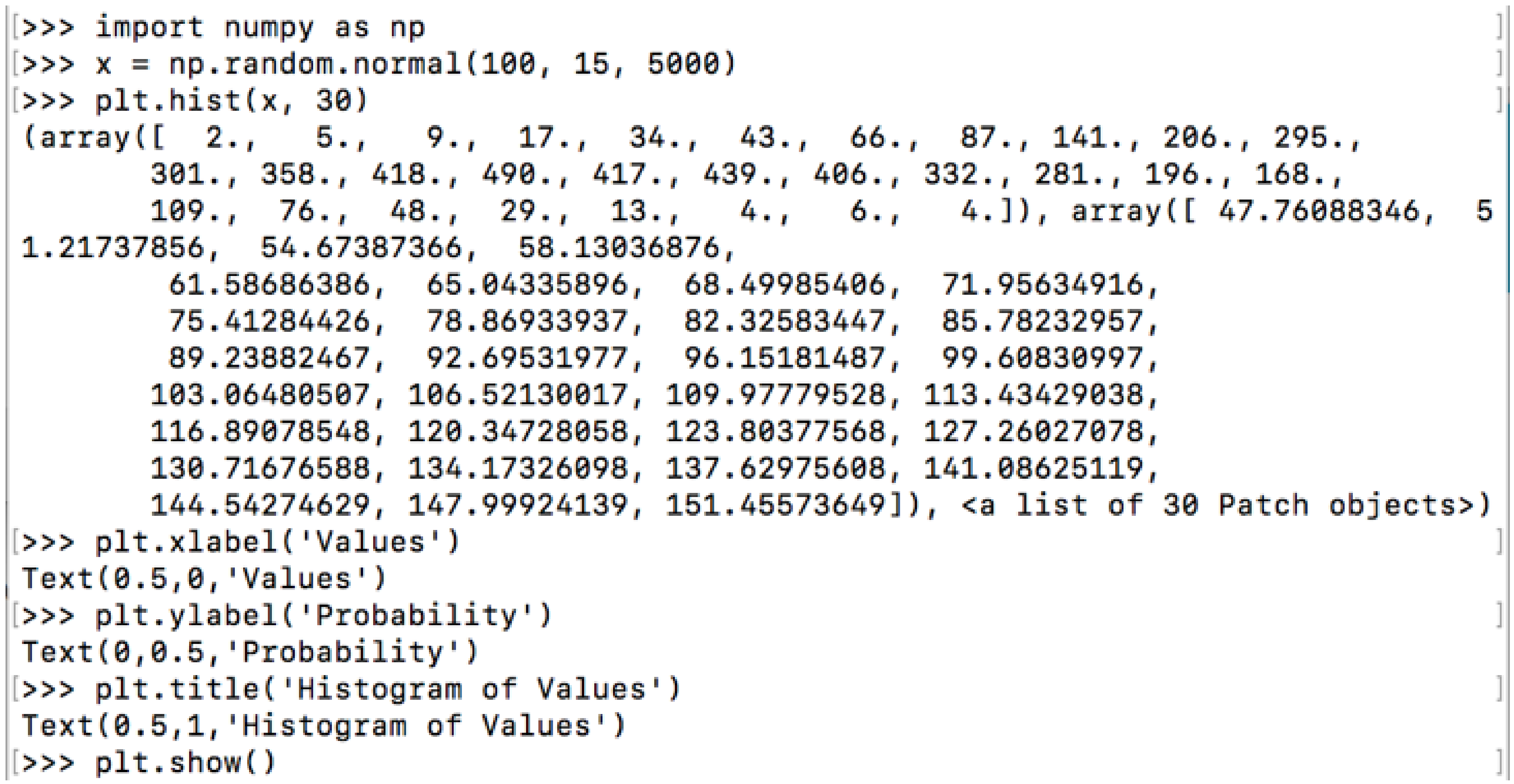
Discussion

Refer to the code in Figure 5.7

- How to set the min and max of x and y axes? (https://matplotlib.org/api/_as_gen/matplotlib.pyplot.axis.html#matplotlib.pyplot.axis)
- How to customize the color, marker and line style? (https://matplotlib.org/api/_as_gen/matplotlib.pyplot.plot.html#matplotlib.pyplot.plot)

Recap

Histogram



```
>>> import numpy as np
>>> x = np.random.normal(100, 15, 5000)
>>> plt.hist(x, 30)
(array([  2.,   5.,   9.,  17.,  34.,  43.,  66.,  87., 141., 206., 295.,
       301., 358., 418., 490., 417., 439., 406., 332., 281., 196., 168.,
       109.,  76.,  48.,  29.,  13.,   4.,   6.,   4.]), array([ 47.76088346,  5
1.21737856,  54.67387366,  58.13036876,
        61.58686386,  65.04335896,  68.49985406,  71.95634916,
        75.41284426,  78.86933937,  82.32583447,  85.78232957,
        89.23882467,  92.69531977,  96.15181487,  99.60830997,
       103.06480507, 106.52130017, 109.97779528, 113.43429038,
       116.89078548, 120.34728058, 123.80377568, 127.26027078,
       130.71676588, 134.17326098, 137.62975608, 141.08625119,
       144.54274629, 147.99924139, 151.45573649]), <a list of 30 Patch objects>)
>>> plt.xlabel('Values')
Text(0.5,0,'Values')
>>> plt.ylabel('Probability')
Text(0,0.5,'Probability')
>>> plt.title('Histogram of Values')
Text(0.5,1,'Histogram of Values')
>>> plt.show()
```

Figure 5.9 Building a histogram of random samples from a normal distribution

Discussion

Box plot

Understand the code below

```
normal_sample = np.random.normal(loc=0.0,scale=1.0,size=10000)
random_sample = np.random.random(size = 10000)
plt.figure()
plt.boxplot([normal_sample,random_sample])
```

Discuss the purposes of parameter *whis*, *showfliers*. (https://matplotlib.org/api/_as_gen/matplotlib.pyplot.boxplot.html#matplotlib.pyplot.boxplot)

Discussion

Scatter plot (https://matplotlib.org/api/_as_gen/matplotlib.pyplot.scatter.html#matplotlib.pyplot.scatter)

Understand the code below

```
x = np.arange(1,9)
y = x
plt.figure()
plt.scatter(x,y,s=100,c=‘red’)
```

Read the example at https://matplotlib.org/gallery/shapes_and_collections/scatter.html#sphx-glr-gallery-shapes-and-collections-scatter-py and customize the color, size and transparency.

Discussion

Bar chart
(https://matplotlib.org/api/_as_gen/matplotlib.pyplot.bar.html#matplotlib.pyplot.bar)

Understand the code below

```
plt.figure()
linear_data = np.arange(1,9)
quadratic_data = linear_data ** 2
xvals = np.arange(len(linear_data))
plt.bar(xvals, linear_data, width = 0.3)
```

To plot the two arrays of data: linear_data and quadratic_data

- Create a side-by-side bar chat
- Read the example at https://matplotlib.org/gallery/lines_bars_and_markers/bar_stacked.html#sphx-glr-gallery-lines-bars-and-markers-bar-stacked-py, and
  - create a stacked bar chart.
  - add legend

Thank you