Computational Environments and Toolchains

Topic 02 — Scientific Computing using Python

Lecture 02 — Numpy

Kieran Murphy and David Power

Department of Computing and Mathematics,
SETU (Waterford).
(kieran.murphy@setu.ie, david.power@setu.ie)

Autumn Semester, 2022

Outline

- matplotlib (2D and 3D plotting library)
- numpy (high performance matrix library)
- scipy (scientific computation library)

Outline

What is NumPy?

NumPy

is an extension to the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large library of high-level mathematical functions to operate on these arrays.

- At the core of the NumPy package, is the ndarray object which encapsulates n-dimensional arrays of homogeneous data.
- Many operations performed using ndarray objects execute in compiled code for performance.
- The standard mathematical and scientific packages in Python use NumPy arrays.

Array Creation

array_creation .py

```
5  import numpy as np
6
7  # from a list
8  arr = np.array([[1, 2, 3], [4, 5, 6]])
9
10 # from sequences
11 np.arange(0, 10, 0.1)
12 np.linspace(0, 2*np.pi, 100)
13
14 # zeros & ones
15 np.zeros((5, 5))
16 np.ones((5, 5))
17
18 # random
19 np.random.random(size=(3, 4))
20 np.random.normal(loc=10., scale=3., size=(3, 4, 5))
```

Array IO

array_io .py

```
5  import numpy as np
6
7  # create an array, write to file, read from file
8  arr = np.array([[1, 2, 3], [4, 5, 6]])
9
10 # save to a text file
11 # creates a space delimited file by default
12 np.savetxt(fname='array_out.txt', X=arr)
13
14 # load text file
15 loaded_arr = np.loadtxt(fname='array_out.txt')
16
17 # verify that save->load worked
18 np.all(arr == loaded_arr)  # True
```

Array Attributes

Arrays are objects and so have attributes and methods.

array_attributes .py

```
5  import numpy as np
6
7  arr = np.arange(10).reshape((2, 5))
8
9  arr.ndim          # 2 number of dimensions
10 arr.shape         # (2, 5) shape of the array
11 arr.size          # 10 number of elements
12 arr.T             # transpose
13 arr.dtype         # data type of elements in the array
```

And many others. Explore in documentation or with TAB complete in ipython.

Array Operations & ufuncs

array_operations .py

```
7  arr1 = np.arange(10).reshape((2, 5))
8  arr2 = np.random.random((2, 5))
9
10 # element-wise for basic and boolean operations
11 #    +, -, *, /, **, np.log, <, >=, ==
12 #    arrays are upcast, resulting in float or boolean arrays
13 arr1 + arr2          # elementwise sum
14 arr1 * arr2          # elementwise multiplication
15
16 # operations in place
17 arr1 += arr2
18
19 # matrix product
20 np.dot(arr1, arr2)
21
22 # similarly numpy ufunc's operate element-wise
23 np.sin(arr1)
24 np.sqrt(arr1)
```

Default behaviour is element-wise

Array Slicing

More powerful than slicing in lists (see boolean indexing).

array_slice .py

```
7  arr = np.arange(20).reshape((4, 5))
8
9  # slicing (like lists for each dimension)
10 arr[0:4, 3:5]  # all rows and last two columns
11 arr[:4,  3:5]  # equivalent - can leave off start
12 arr[:, 3:]     # equivalent - can leave off end
13 arr[slice(None), slice(3, None)]  # equivalent - can use slice()
14
15 # integer indices
16 arr[[1, 2], :]              # rows one and two, all columns
17 arr[np.array([1, 2]), :]    # equivalent
18
19 # boolean indices
20 arr[[False, True, True, False], :]              # equivalent
21 arr[np.array([False, True, True, False]), :]    # equivalent
```

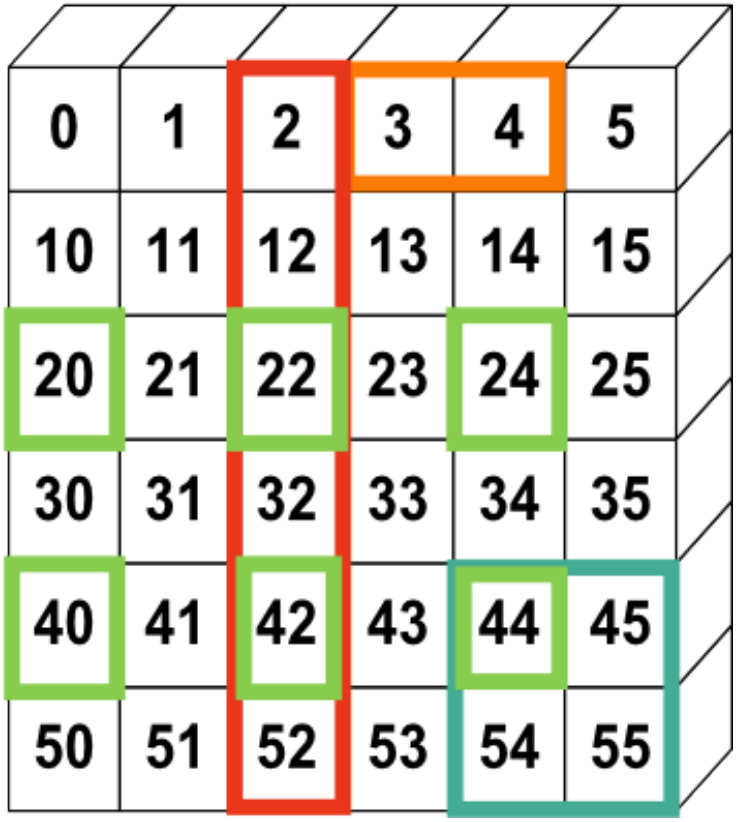
Array Slicing

```
>>> a[0,3:5]
array([3,4])

>>> a[4:,4:]
array([[44, 45],
       [54, 55]])

>>> a[:,2]
array([2,12,22,32,42,52])

>>> a[2::2,::2]
array([[20,22,24]
       [40,42,44]])
```



From Scipy Lecture Notes
(http://www.scipy-lectures.org/index.html)

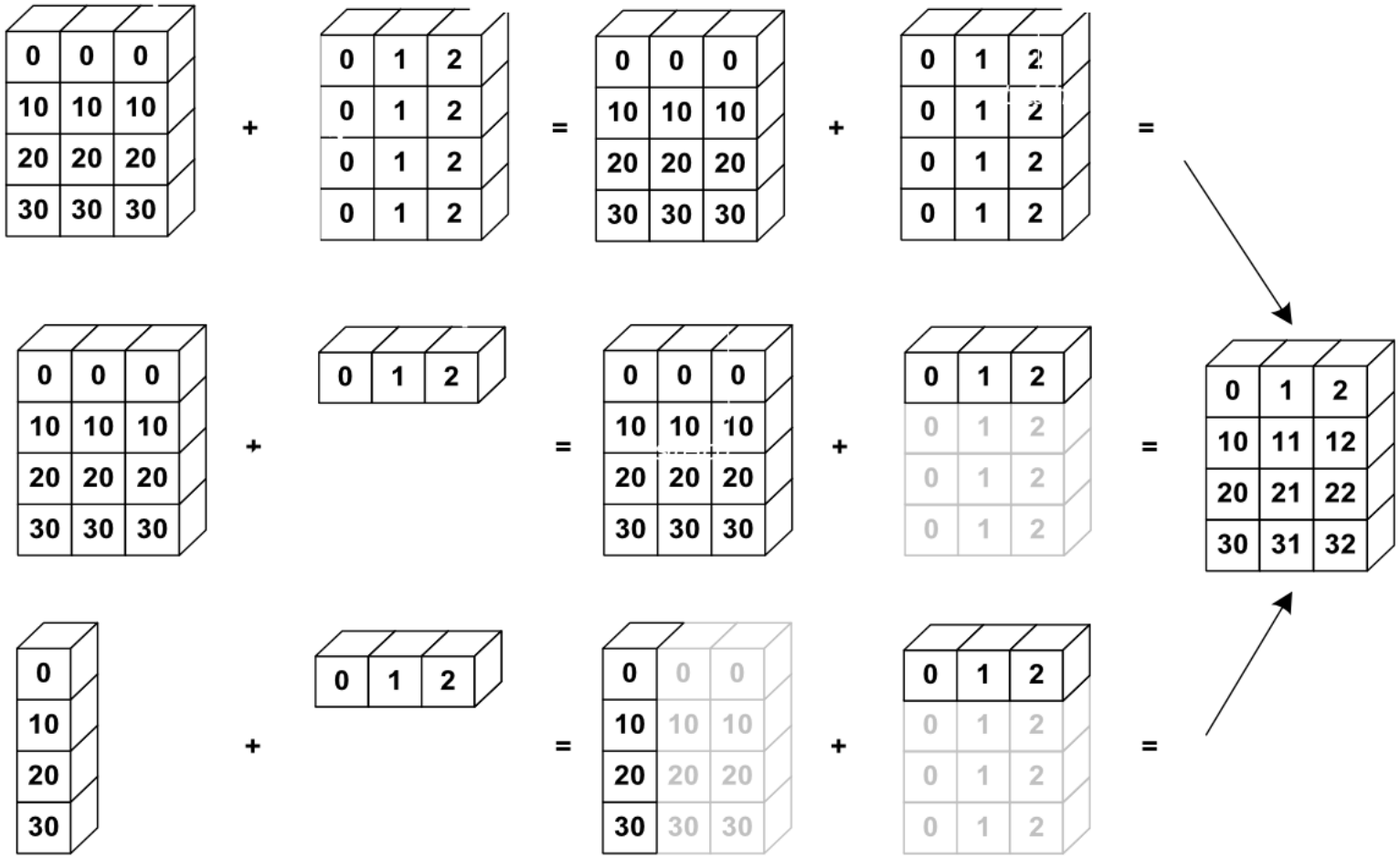
Array Broadcasting & Vectorisation

Broadcasting allows us to operate on arrays of different shapes by 'copying' smaller arrays when possible. This allows us to write more efficient and readable code (with fewer for loops).

array_broadcasting .py

```
5  import numpy as np
6
7  # multiplication by a scalar
8  arr = np.random.random((4, 5))
9  result = arr * 5                # multiply each element by 5
10
11 # scales the first column by 0.
12 # scales the second column by 1.
13 # etc.
14 result = arr * np.arange(5)
```

Array Broadcasting



From Scipy Lecture Notes
(`http://www.scipy-lectures.org/index.html`)

Summary

- Know how to create arrays : `array`, `arange`, `ones`, `zeros`.
- Know the shape of the array with `array.shape`, then use slicing to obtain different views of the array: `array[::2]`, etc. Adjust the shape of the array using `reshape` or flatten it with `ravel`.
- Obtain a subset of the elements of an array and/or modify their values with masks (boolean indexing)
- Know miscellaneous operations on arrays, such as finding the mean or max (`array.max()`, `array.mean()`).

  No need to retain everything, but have the reflex to search in the documentation

  (online docs, `help()`, `lookfor()`)!!
- For advanced use:
  - Master the indexing with arrays of integers, as well as broadcasting.
  - Know more Numpy functions to handle various array operations.