NumPy

BILD 62

Before we dive into new content, let’s see if we can apply what we’ve learned to some *real* code documentation

`Class`

Template for objects

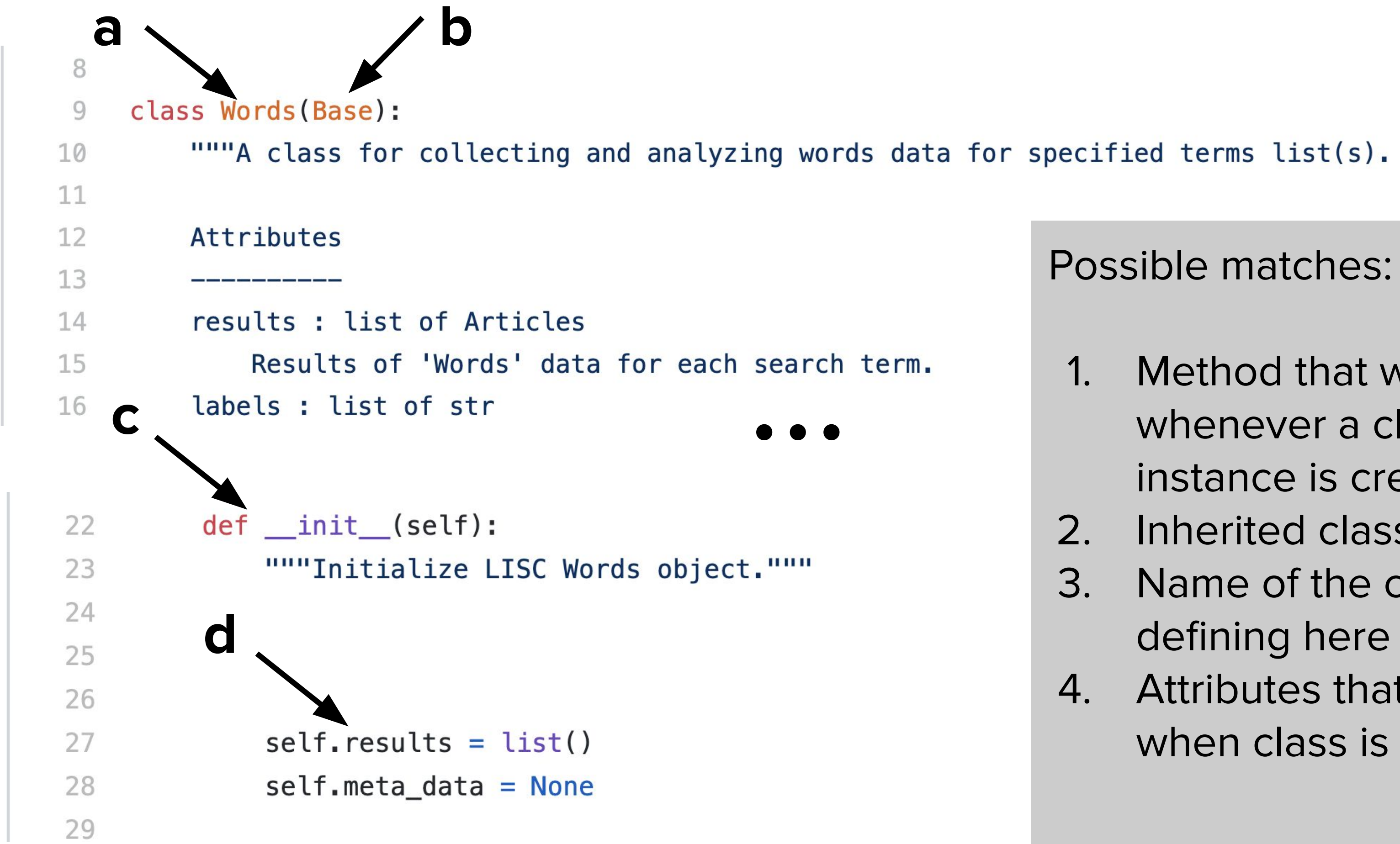
Defines properties for objects (**attributes**)

Defines behaviors for objects (**methods**)

`Object`

Instance of a class

Has attributes that are defined differently in each instance (using `__init__` method) or that are always inherited from the `Class`



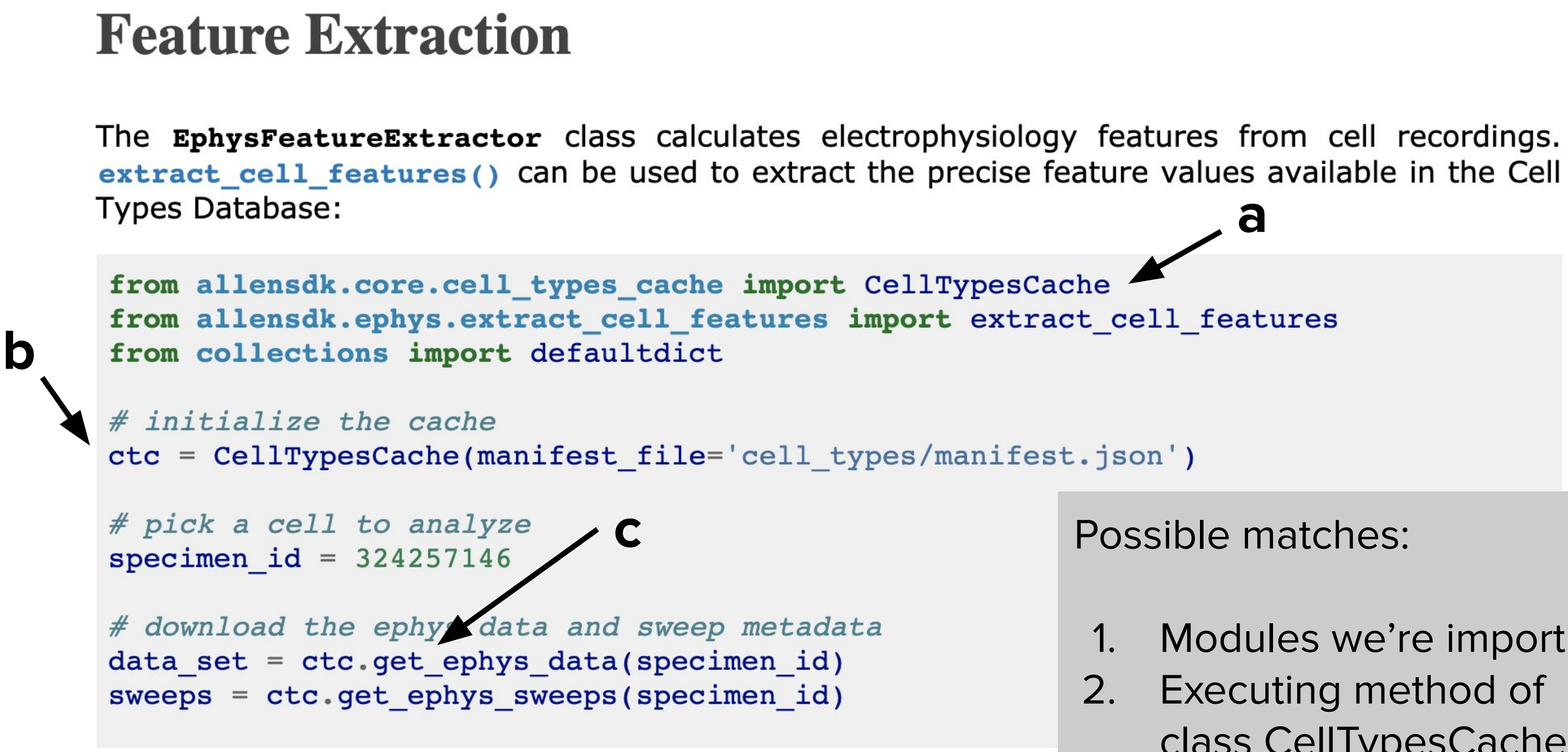
```python
class Words(Base):
    """A class for collecting and analyzing words data for specified terms list(s).

    Attributes
    ----------
    results : list of Articles
        Results of 'Words' data for each search term.
    labels : list of str
```

```python
    def __init__(self):
        """Initialize LISC Words object."""



        self.results = list()
        self.meta_data = None

```

Possible matches:

1. Method that will execute whenever a class instance is created
2. Inherited class
3. Name of the class we are defining here
4. Attributes that will update when class is initialized

From https://github.com/lisc-tools/lisc/blob/c44af07492165f9a35b653b6aa1da1f397044593/lisc/objects/words.py



Feature Extraction

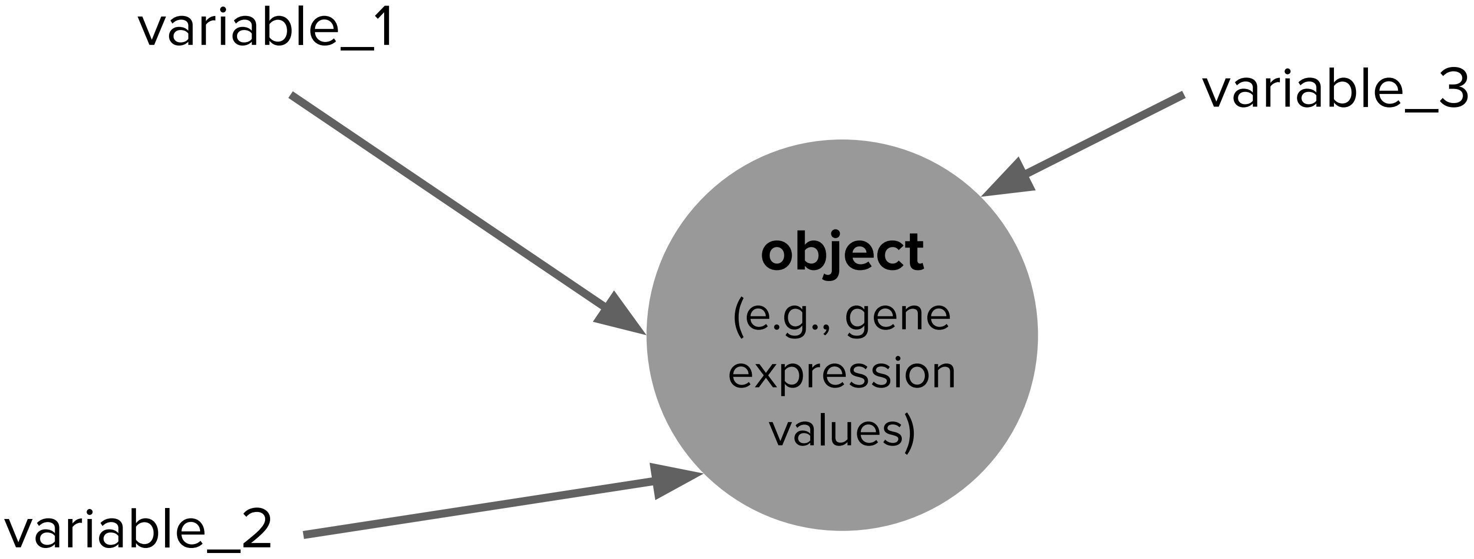
The **EphysFeatureExtractor** class calculates electrophysiology features from cell recordings. `extract_cell_features()` can be used to extract the precise feature values available in the Cell Types Database:

```
from allensdk.core.cell_types_cache import CellTypesCache
from allensdk.ephys.extract_cell_features import extract_cell_features
from collections import defaultdict

# initialize the cache
ctc = CellTypesCache(manifest_file='cell_types/manifest.json')

# pick a cell to analyze
specimen_id = 324257146

# download the ephys data and sweep metadata
data_set = ctc.get_ephys_data(specimen_id)
sweeps = ctc.get_ephys_sweeps(specimen_id)
```

Possible matches:

1. Modules we're importing
2. Executing method of class CellTypesCache
3. Instance of class CellTypesCache

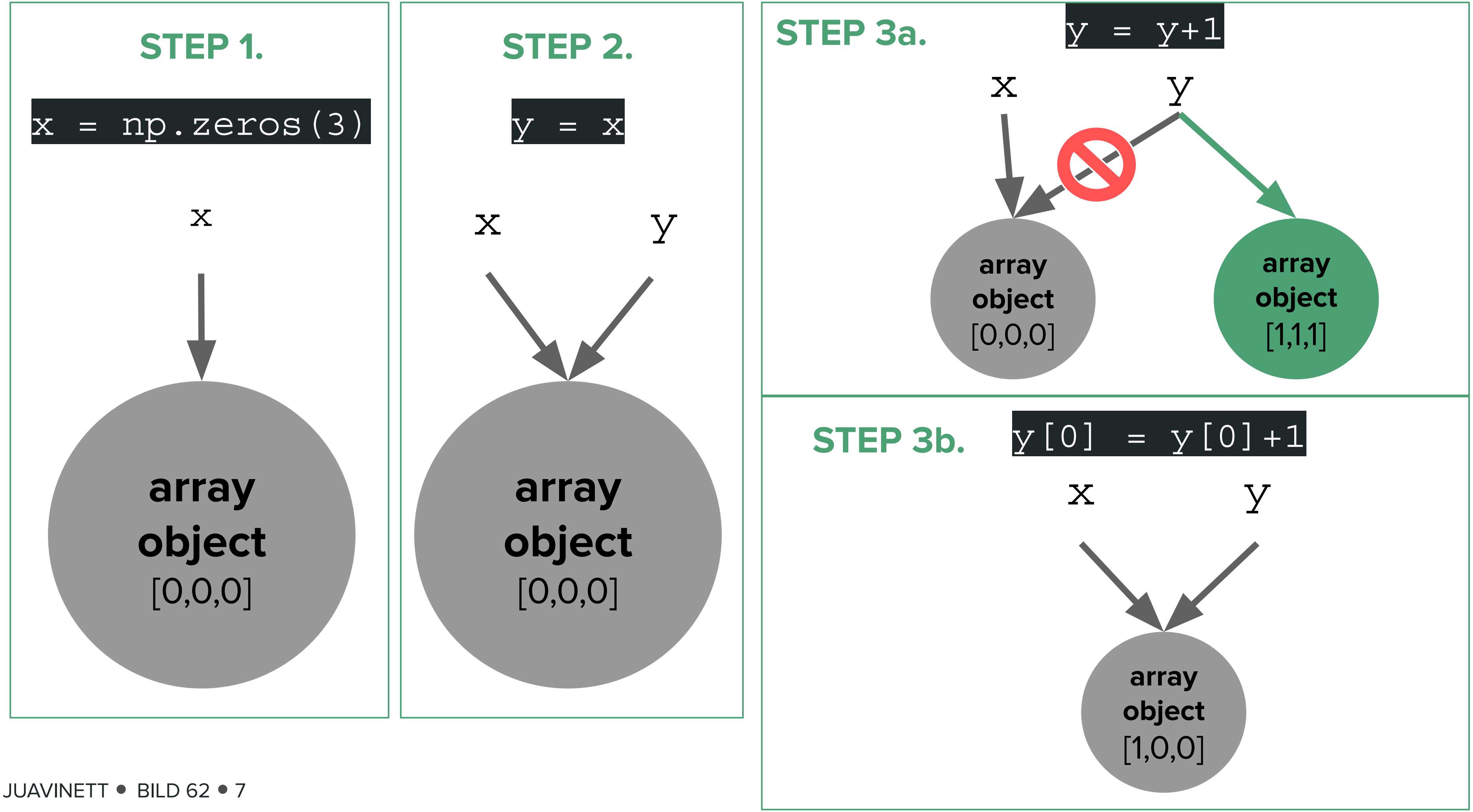
From https://alleninstitute.github.io/AllenSDK/cell_types.html



Object-oriented programming

This is how Python containers typically work, but saving all of our data in lists isn't great for performance or memory.
NumPy is a tool for computing with big arrays, and is much more efficient.*

* for details, see this breakdown



STEP 1.

```
x = np.zeros(3)
```

STEP 2.

```
y = x
```

STEP 3a.

```
y = y+1
```

STEP 3b.

```
y[0] = y[0]+1
```

Objectives for today

- Install and import packages for Python
- Create NumPy arrays
- Execute methods & access attributes of arrays
- Demonstrate how images can be stored in arrays

Python supports **modular programming** in multiple ways.

Functions and **classes** are examples of tools for low-level modular programming.

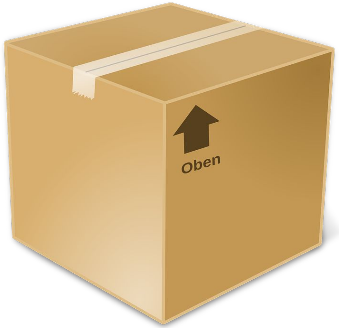
Python **modules** are a higher-level modular programming construct, where we can collect related variables, functions and classes in a module.

Modules are often bundled up into **packages**.

Packages in Python

Python's standard library works for some purposes, but there are many very useful packages for additional purposes:

- **numpy** (http://numpy.scipy.org): numerical Python
- **scipy** (http://www.scipy.org): scientific Python; built on numpy
- **matplotlib** (http://www.matplotlib.org) graphics library



Installing packages & importing modules

To install packages, use

```
$ pip install PACKAGE
```

We typically won't need to do this in the DataHub, because many packages have been installed into our container. However, you *may* need to do this for local notebook operation.

You can then import modules from the package with

```
>>> from PACKAGE import MODULE
```

to see all of the modules available, use

```
>>> print(dir(MODULE))
```

Module	Built-In	Description
csv	Yes	Aids in the reading, writing, and analysis of CSV files.
zipfile	Yes	Aids in the creation and extraction of compressed ZIP archive files.
matplotlib	No	Graphics library for plotting
plotly	No	A graphics library used for creating interactive plots for the web.
seaborn	No	A graphics library built on top of matplotlib with high-quality plots
pandas	No	A data processing library that specializes in data frames, which are analogous to spreadsheets.
scikit-learn	No	Contains basic tools for machine learning (i.e., helping to learn from data and make predictions).
numpy	No	Offers highly efficient data processing.
pygame	No	A game programming library that helps to build interactive, graphical games in Python.
django	No	Web development library that aids in designing websites and web applications.

Common Python modules — ones we’ll work with are highlighted

We’re learning how to deal with more and more complex data

```
data_point = 8.02
```

→ **single variable** (int, float, string)

```
data = [8.38,3.34,6.35]
```

→ **data structure** (list, tuple, dictionary)

```
big_data = [data_1,data_2,...]
```

→ **array** or **dataframe**

NumPy is the fundamental package for scientific computing with Python

- A numpy **array** is a grid of values which are all the same type (they're **homogenous**)
- Useful attributes:
  - `ndim` = # of dimensions
  - `shape` = a tuple of integers giving the size of the array along each dimension
  - `dtype` = type of data

Numpy Arrays

my_array = 1D array

3	2	4	1

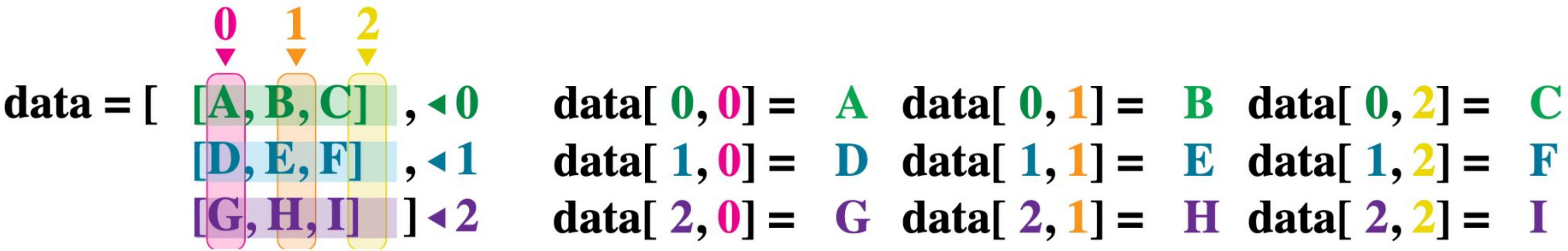
```
my_array[0] = 3

my_array.ndim = 1

my_array.shape = (4,)

my_array.size = 4
```

2D array

3	2	4	1
1	2	5	3

how to index 2D NumPy arrays

```
my_array[1,3] = 3

my_array.ndim = 2

my_array.shape = (2,4)

my_array.size = 8
```



Indexing numpy arrays

Image from Programming with Python: Analyzing Patient Data

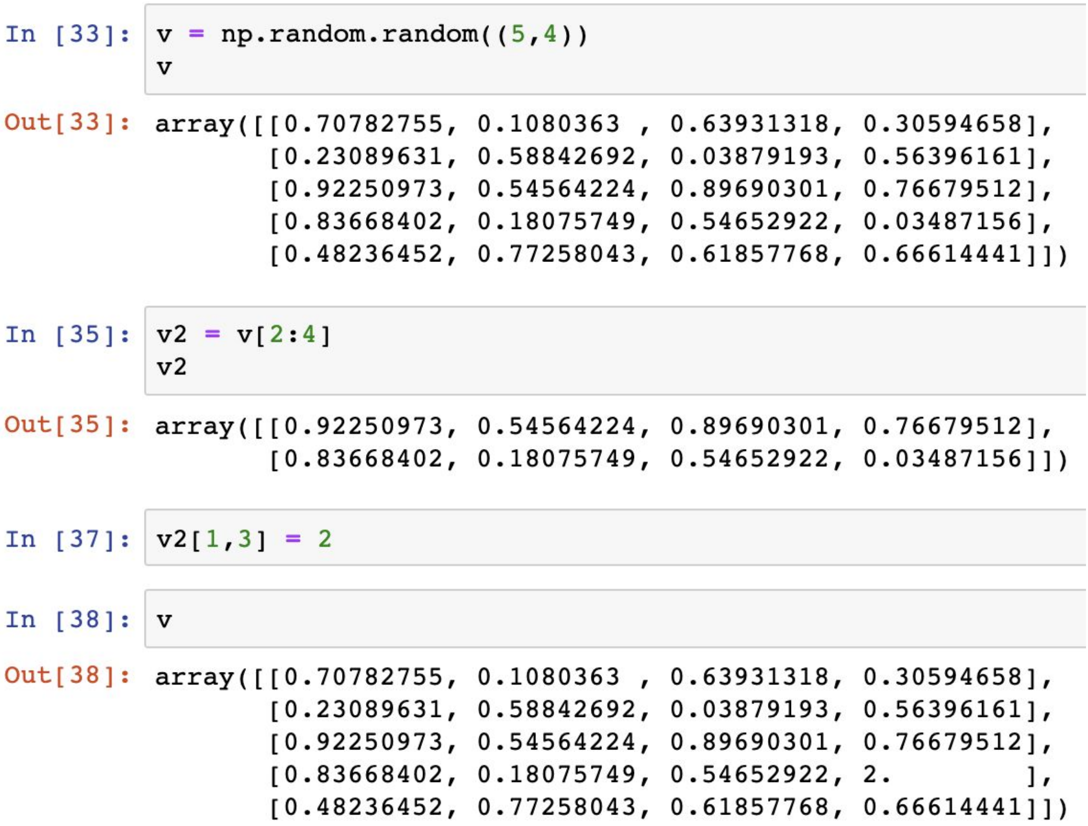
Slicing & indexing NumPy arrays works *almost* the same as with Python lists

However, **be aware that if you slice an array, it changes the original array**.

If you need to copy, you need to explicitly do:

`v3 = v[2:4].copy()`

In this case, we would not change original array (v).



```
In [33]: v = np.random.random((5,4))
         v

Out[33]: array([[0.70782755, 0.1080363 , 0.63931318, 0.30594658],
                [0.23089631, 0.58842692, 0.03879193, 0.56396161],
                [0.92250973, 0.54564224, 0.89690301, 0.76679512],
                [0.83668402, 0.18075749, 0.54652922, 0.03487156],
                [0.48236452, 0.77258043, 0.61857768, 0.66614441]])

In [35]: v2 = v[2:4]
         v2

Out[35]: array([[0.92250973, 0.54564224, 0.89690301, 0.76679512],
                [0.83668402, 0.18075749, 0.54652922, 0.03487156]])

In [37]: v2[1,3] = 2

In [38]: v

Out[38]: array([[0.70782755, 0.1080363 , 0.63931318, 0.30594658],
                [0.23089631, 0.58842692, 0.03879193, 0.56396161],
                [0.92250973, 0.54564224, 0.89690301, 0.76679512],
                [0.83668402, 0.18075749, 0.54652922, 2.        ],
                [0.48236452, 0.77258043, 0.61857768, 0.66614441]])
```

You can also use **lists** & **booleans** to index NumPy arrays

```
my_array[[1,2,3]]
```

```
my_array[my_array > 1]
```

We can also use this to selectively operate on values in the array that meet our criteria:

```
my_array[my_array > 1] = my_array[my_array > 1] * 2
```

Useful NumPy functions

`np.zeros()`

`np.empty()`

`np.linspace()`

`np.arange()`

`np.reshape()`

`np.random.random()`

`np.vstack()`

`np.hstack()`

`np.save()`

`np.load()`

See here for a useful Numpy overview.

Key NumPy takeaways

- Import a library into a program using `import libraryname`
- Use the NumPy library to work with arrays in Python.
- The expression array.shape gives the shape of an array.
- Use `array[x, y]` to select a single element from a 2D array.
- Array indices start at 0, not 1.
- Use `low:high` to specify a slice that includes the indices from low to high-1.
- Use `np.mean(array)`, `np.max(array)`, and `np.min(array)` to calculate simple statistics.
- Use `np.mean(array, axis=0)` or `np.mean(array, axis=1)` to calculate statistics across the specified axis.

Objectives for today

- Install and import packages for Python
- Create NumPy arrays
- Execute methods & access attributes of arrays
- **Demonstrate how images can be stored in arrays**

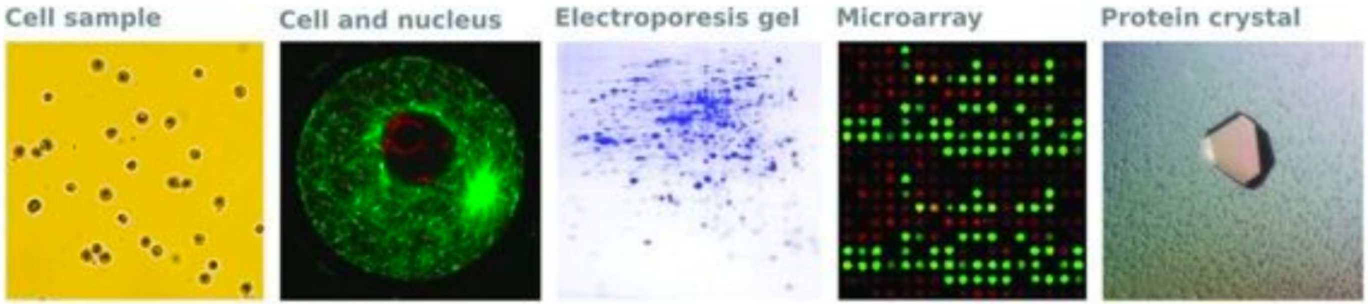


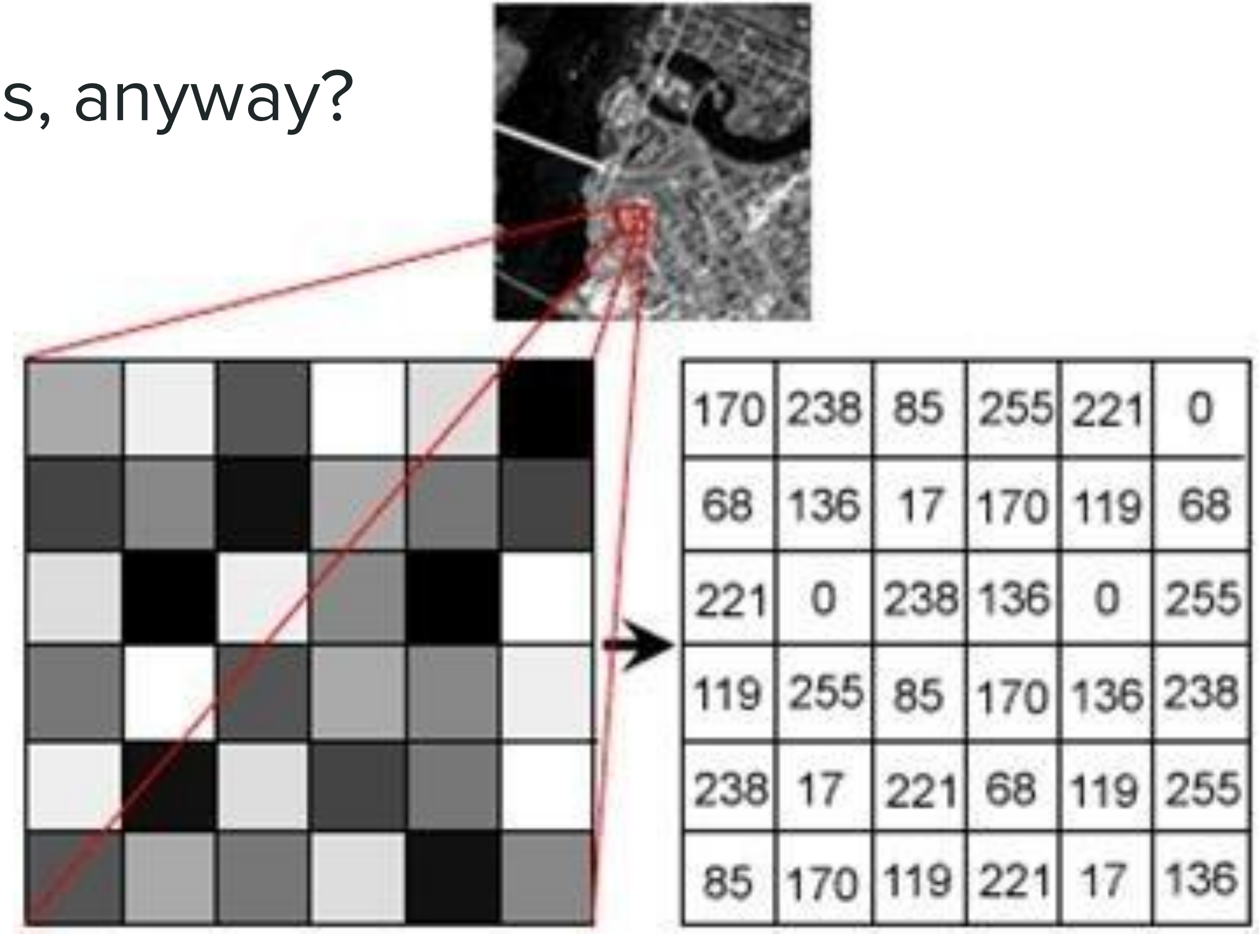
Figure 18.1 (Plate 5). Examples of a variety of different kinds of images used in biology. Shown from left to right are: a microscope image of a mammalian cell culture (courtesy Dr. Anja Winter, University of Leicester); a red-green fluorescence microscope image of an oocyte and its nucleus (courtesy Dr. Melina Schuh, MRC Laboratory of Molecular Biology); a two-dimensional electrophoresis gel of a plant proteome (courtesy Prof. Paul Dupree, University of Cambridge); an image of a DNA microarray (courtesy Karen Howarth, University of Cambridge); a protein crystal that has been grown for structure determination by X-ray crystallography (courtesy Dr. Aleksandra Watson, University of Cambridge).

We use lots of images in biology

Figure from Python Programming for Biology

What are images, anyway?

Gray scale images mean each pixel has just one value

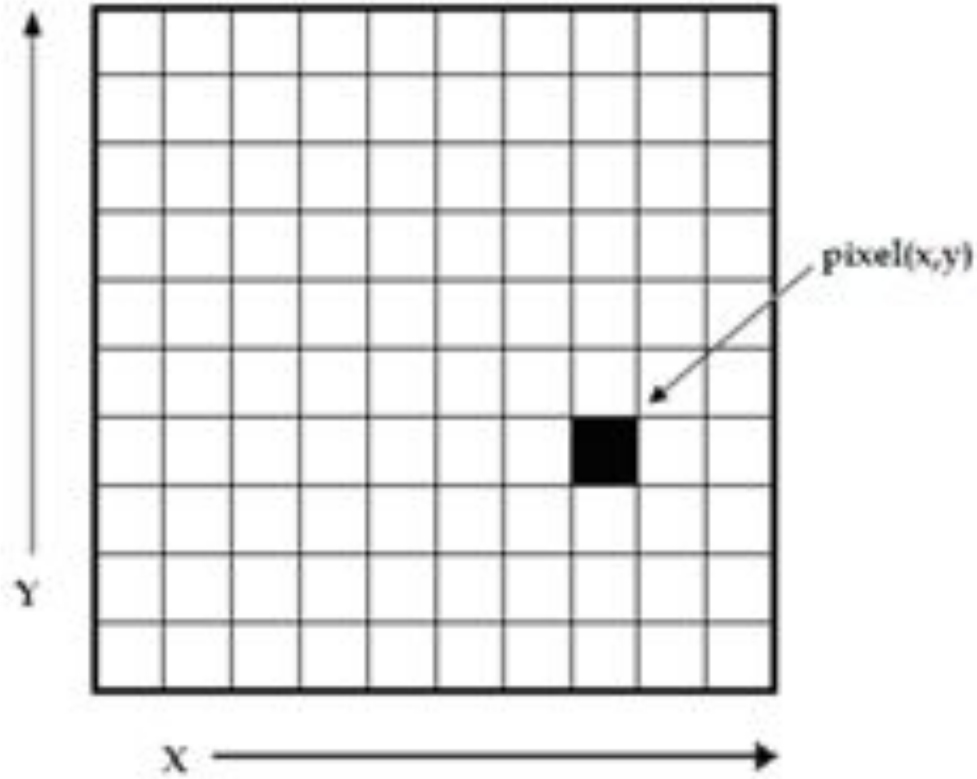


170	238	85	255	221	0
68	136	17	170	119	68
221	0	238	136	0	255
119	255	85	170	136	238
238	17	221	68	119	255
85	170	119	221	17	136

What are images, anyway?

Images can be represented as 2D NumPy arrays

By convention [0,0] is the top left corner



Resources

Numerical & Scientific Computing with Python: Introduction into NumPy

Lecture-2-Numpy.ipynb

Programming with Python: Analyzing Patient Data