Lecture 4: NumPy

Course: Biomedical Data Science

Parisa Rashidi
Fall 2018

Agenda

- More Python
  - NumPy, Pandas
- Later
  - Preprocessing structured data
  - Preprocessing unstructured data (time series, text, ..)

Important Dates

Date	Tuesday Class	Thursday Class
Week 1 (08/20 - 08/24)		Intro
Week 2 (08/27- 08/31)		Paper Reading
Week 3 (09/03 - 09/07)		Paper Reading
Week 4 (09/10 - 09/14)		Paper Reading
Week 5 (09/17 - 09/21)	HW1	Guest Lecture
Week 6 (09/24 - 09/28)	HW1 Due	Paper Reading
Week 7 (10/01 - 10/05)	HW2	Paper Reading
Week 8 (10/08 - 10/12)	HW2 Due	Paper Reading
Week 9 (10/15 – 10/19)	HW3	BMES Conference
Week 10 (10/22 – 10/26)	Grad Survey Proposal	Paper Reading
Week 11 (10/29 – 11/02)	HW3 Due	Paper Reading
Week 12 (11/05 – 11/09)	HW4	Paper Reading
Week 13 (11/12 – 11/16)	HW4 Due	Grad Survey Presentation
Week 14 (11/19 – 11/23)		Holiday
Week 15 (11/26 – 11/30)		Exam
Week 16 (12/03 – 12/07)		Reading Day

Graduate Survey

- Review and present a survey paper
- Seven groups (2 members each)
  - Please sign up as a group or let me know so I can assign randomly
  - If you do not sign up in two weeks, I will randomly assign you to a group
- Topic proposal deadline: 10/23
- Grad only

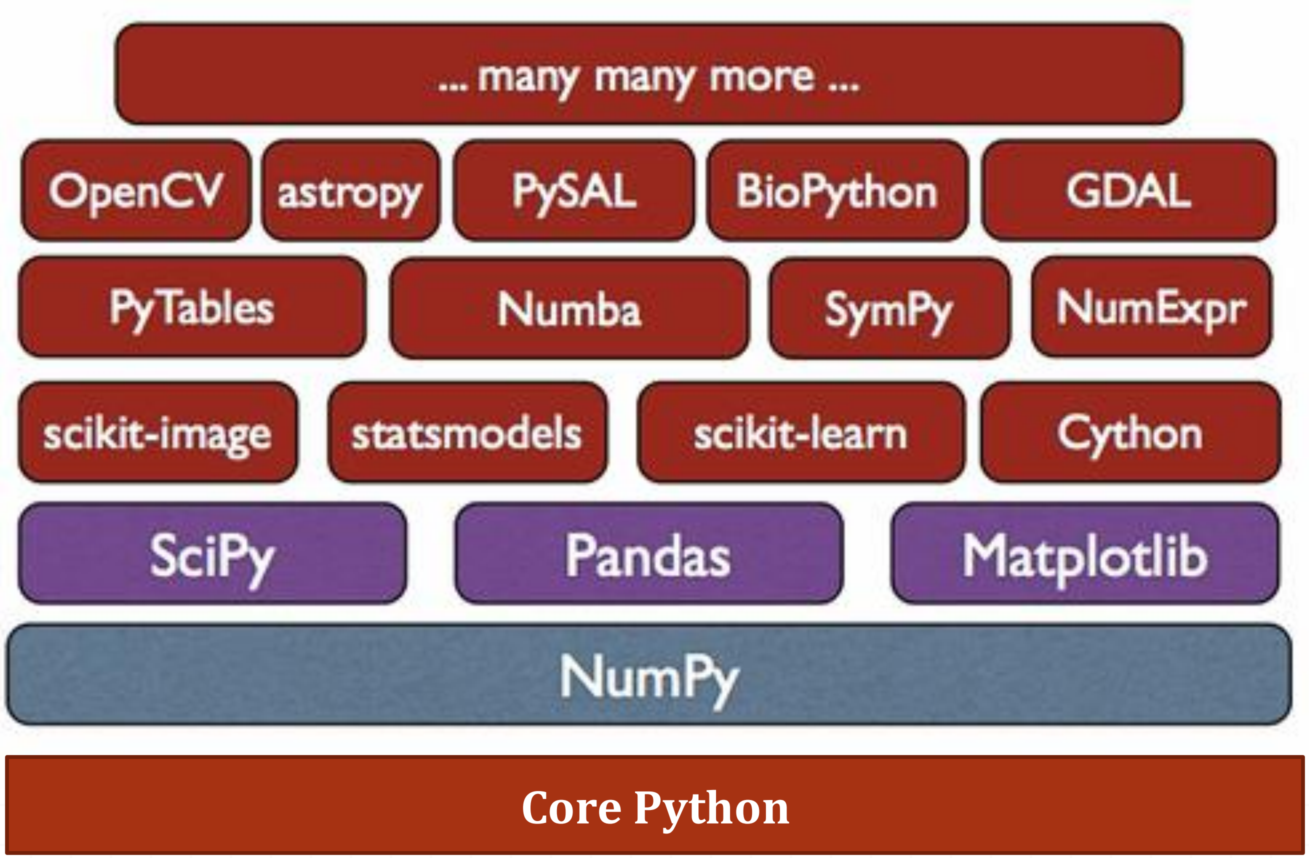
Disclaimer

The following slides are based on:

PHY 546: Python for Scientific Computing

Instructor: Michael Zingale

Python Library Stack



... many many more ...

OpenCV astropy PySAL BioPython GDAL

PyTables Numba SymPy NumExpr

scikit-image statsmodels scikit-learn Cython

SciPy Pandas Matplotlib

NumPy

Core Python

http://quasiben.github.io/PyDataEMC/#/9

Intro to NumPy: Arrays

- NumPy provides a multidimensional array
  - All elements must be the same data type
  - Many different datatypes supported

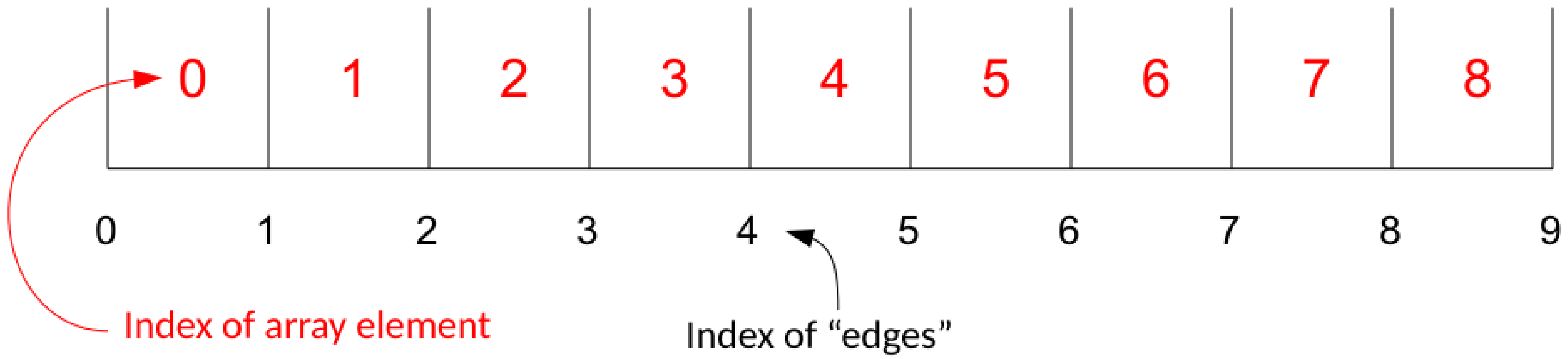
- Arithmetic operations work on arrays
- Provides MANY functions that operate on whole arrays
  - These operations are written in a compiled language, and run fast
  - Generally speaking, you want to avoid loops to get the best performance.
    - Can sometimes make code unreadable
- Lots of ways to create arrays

Intro to NumPy: Array Operations

- Arithmetic operator (+, –, /, *) work elementwise
  - `A * B` is not a matrix product, but instead multiples the corresponding elements in each array together
  - `dot(A,B)` does a dot product
- Universal functions (`sin`, `cos`, `exp`, ...) work elementwise
- New @ operator
  - Accepted for python 3.5, the "@" is a new operator in python available for overloading. NumPy will implement it as matrix multiplication
    - http://legacy.python.org/dev/peps/pep-0465/
    - `A @ B` will be equivalent to `np.dot(A,B)`
- Array creation and operations examples...

Intro to NumPy: Array Indexing/Slicing

- Biggest source of confusion: selecting a range is best thought of as referring to the "edges" of the array locations
  - Differs from Fortran, IDL

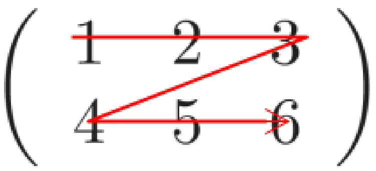


  - For the array above:
    - `A[2] = 2`
    - `A[2:3] = [2]`
    - `A[2:4] = [2 3]`
  - Note also: zero-based indexing

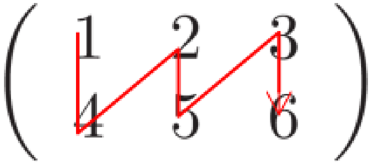
Note: this same behavior applies to Python lists and strings when slicing

Arrays

- Building block of many numerical methods
- Row vs. Column major: A(m,n)
  - First index is called the row
  - Second index is called the column
  - Multi-dimensional arrays are flattened into a one-dimensional sequence for storage
  - Row-major (C, python): rows are stored one after the other
  - Column-major (Fortran, matlab): columns are stored one after the other
- Ordering matters for:
  - Passing arrays between languages
  - Deciding which index to loop over first



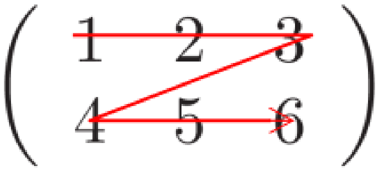
Row major



Column major

Intro to NumPy: Array Indexing/Slicing

- Remember, multi-dimensional arrays are stored in row-major fashion
  - Rows are stored one after the other, within a row, the column data is closest to one another



  - You see this when you print an array:
    - `a = numpy.arange(15).reshape(3,5)`
    - `print a`

```
[[ 0  1  2  3  4]
 [ 5  6  7  8  9]
 [10 11 12 13 14]]
```

3 rows,
5 columns

Note that the braces [ ]
show that the columns
are together

- Some slicing examples…

Intro to NumPy: Array Views/Copies

- When “copying”, need to understand if two arrays, A and B, point to:
  - the same array (including shape and data/memory space)
  - the same data/memory space (but perhaps different shapes)
  - a separate copy of the data (i.e. stored separately in memory)
- `B = A` (assignment)
  - No copy is made. A and B point to the same data in memory and share the same shape, etc.
- `B = A[:]` (view or shallow copy)
  - The shape info for A and B are stored independently, but both point to the same memory location for the data
- `B = A.copy()` (deep copy)
  - The data in B is stored completely separately in memory from A
- Copying examples…

Intro to NumPy: Boolean Indexing

- Many fancy ways to index arrays
- `A[A > 4] = 0`
  - Boolean indexing