Python for Data Science

* NumPy - "Numerical Python". Package for scientific computing. Useful methods of arrays.
* Pandas - "Python Data Analysis Library". Useful for cleaning, transforming, analyzing data.
* Matplotlib - makes 2d plots such as histograms, bar-graphs, scatterplots etc.
* SciKit-Learn - Machine Learning procedures (both supervised and unsupervised).

NumPy

* Create

np.zeros((3,3), dtype) # create 3\*3 matrix with zeros

np.ones(5) # create vector of length 5 with ones

np.eye(4) # create identity matrix of size 4 (square matrix)

np.full((2,3,4),5) # create 3-dims array 2 slis, 3 rows and 4 cols with 5

np.array([[1,2],[3,4],[4,5]]) # create 3\*2 matrix, by a list of lists

np.array([1,2,3]) # create one dimension array (default as column vector)

by create random variables

np.random.random((3,3)) # create 3\*3 matrix with uniform(0,1) entries

np.random.normal(0,1,(3,3)) # create 3\*3 matrix with standard normal entries

np.random.randint(10, size = (3,4,5)) # fill an array with random ints in range(10)

by extract entries from old array

new\_array [ : : p+1] = x # in\_class\_p4

new\_array [1:n-1, 1:m-1] = 0 # hw5\_p1

by concatenate two arrays

np.concatenate([A[np.newaxis, :, :], B[np.newaxis, :, :]], axis = 0) # in\_class\_p4

* attributes of ndarrays

ndarray.ndim / ndarray.shape / ndarray.size / ndarray.dtype

* indexing

y = np.array(range(1,25)).reshape(2,3,4); y[1,2,3]

* slicing 注意~

# slice is a view of the original array, modifying the slice will modify the array

z = np.array(range(24)).reshape(3,4,2)

slice = z[1, :2, ] # slicing a subarray

ind\_slice = slice.copy() # make an independent copy

ind\_slice[0,0] = 90 # modifying the slice independently

* reshaping 注意~

# reshape of an array is a view of the original array, not an independent copy

np.concatenate([A,B]) # concatenate two arrays (default along rows)

np.concatenate([A,B], axis = 1) # along columns

np.vstack([A,B]) # concatenate along rows

np.hstack([A,B]) # concatenate along columns

left, right = np.split(x,[1], axis = 1) # split off at the first column

top, bottom = np.vsplit(x,[1]) # split off at the first row

left, right = np.hsplit(x,[1]) # split off at the first column

* Math

# Operations on nparrays are implemented as vectorized functions.

# Vectorized programs can run multiple operations from a single instruction, which can be carried out much faster than loop, doing operations on each element one-by-one.

np.log(x) / np.sqrt(x)

* Data Aggregation

np.sum | np.mean | np.std | np.var

np.min | np.max | np.median | np.percentile

np.argmin | np.argmax # find index of min and max values

* Broadcasting

# Tip: for different dimensions arrays, we could “inflate” NumPy arrays by **np.newaxis** to map them to a higher dimension in order to combine arrays with each other.

X = np.array([1, 2, 3])

Y = np.array([0.1, 0.2])

X[ : , np.newaxis] + Y[np.newaxis, : ]

* Boolean Masks

np.sum(x<3, axis = 1) # count satisfying entries per row

* Fancy Indexing

x[[0,2,4]] # extract the 1st, 3rd, 5th elements

x[1,[3,0]] # extract the 4th and 1st elements in the second row

subset = np.array([[0,1],[8,9]]); x[subset]

SciPy