Data Science Cheat Sheet

NumPy

KEY

We'll use shorthand in this cheat sheet

`arr` - A numpy Array object

IMPORTS

Import these to start

`import numpy as np`

IMPORTING/EXPORTING

`np.loadtxt('file.txt')` - From a text file

`np.genfromtxt('file.csv',delimiter=',')` - From a CSV file

`np.savetxt('file.txt',arr,delimiter=' ')` - Writes to a text file

`np.savetxt('file.csv',arr,delimiter=',')` - Writes to a CSV file

CREATING ARRAYS

`np.array([1,2,3])` - One dimensional array

`np.array([(1,2,3),(4,5,6)])` - Two dimensional array

`np.zeros(3)` - 1D array of length **3** all values **0**

`np.ones((3,4))` - **3**x**4** array with all values **1**

`np.eye(5)` - **5**x**5** array of **0** with **1** on diagonal (Identity matrix)

`np.linspace(0,100,6)` - Array of **6** evenly divided values from **0** to **100**

`np.arange(0,10,3)` - Array of values from **0** to less than **10** with step **3** (eg **[0,3,6,9]**)

`np.full((2,3),8)` - **2**x**3** array with all values **8**

`np.random.rand(4,5)` - **4**x**5** array of random floats between **0**-**1**

`np.random.rand(6,7)*100` - **6**x**7** array of random floats between **0**-**100**

`np.random.randint(5,size=(2,3))` - **2**x**3** array with random ints between **0**-**4**

INSPECTING PROPERTIES

`arr.size` - Returns number of elements in **arr**

`arr.shape` - Returns dimensions of **arr** (rows, columns)

`arr.dtype` - Returns type of elements in **arr**

`arr.astype(dtype)` - Convert **arr** elements to type **dtype**

`arr.tolist()` - Convert **arr** to a Python list

`np.info(np.eye)` - View documentation for **np.eye**

COPYING/SORTING/RESHAPING

`np.copy(arr)` - Copies **arr** to new memory

`arr.view(dtype)` - Creates view of **arr** elements with type **dtype**

`arr.sort()` - Sorts **arr**

`arr.sort(axis=0)` - Sorts specific axis of **arr**

`two_d_arr.flatten()` - Flattens 2D array **two_d_arr** to 1D

`arr.T` - Transposes **arr** (rows become columns and vice versa)

`arr.reshape(3,4)` - Reshapes **arr** to **3** rows, **4** columns without changing data

`arr.resize((5,6))` - Changes **arr** shape to **5**x**6** and fills new values with **0**

ADDING/REMOVING ELEMENTS

`np.append(arr,values)` - Appends **values** to end of **arr**

`np.insert(arr,2,values)` - Inserts **values** into **arr** before index **2**

`np.delete(arr,3,axis=0)` - Deletes row on index **3** of **arr**

`np.delete(arr,4,axis=1)` - Deletes column on index **4** of **arr**

COMBINING/SPLITTING

`np.concatenate((arr1,arr2),axis=0)` - Adds **arr2** as rows to the end of **arr1**

`np.concatenate((arr1,arr2),axis=1)` - Adds **arr2** as columns to end of **arr1**

`np.split(arr,3)` - Splits **arr** into **3** sub-arrays

`np.hsplit(arr,5)` - Splits **arr** horizontally on the **5**th index

INDEXING/SLICING/SUBSETTING

`arr[5]` - Returns the element at index **5**

`arr[2,5]` - Returns the 2D array element on index **[2][5]**

`arr[1]=4` - Assigns array element on index **1** the value **4**

`arr[1,3]=10` - Assigns array element on index **[1][3]** the value **10**

`arr[0:3]` - Returns the elements at indices **0,1,2** (On a 2D array: returns rows **0,1,2**)

`arr[0:3,4]` - Returns the elements on rows **0,1,2** at column **4**

`arr[:2]` - Returns the elements at indices **0,1** (On a 2D array: returns rows **0,1**)

`arr[:,1]` - Returns the elements at index **1** on all rows

`arr<5` - Returns an array with boolean values

`(arr1<3) & (arr2>5)` - Returns an array with boolean values

`~arr` - Inverts a boolean array

`arr[arr<5]` - Returns array elements smaller than **5**

SCALAR MATH

`np.add(arr,1)` - Add **1** to each array element

`np.subtract(arr,2)` - Subtract **2** from each array element

`np.multiply(arr,3)` - Multiply each array element by **3**

`np.divide(arr,4)` - Divide each array element by **4** (returns **np.nan** for division by zero)

`np.power(arr,5)` - Raise each array element to the **5**th power

VECTOR MATH

`np.add(arr1,arr2)` - Elementwise add **arr2** to **arr1**

`np.subtract(arr1,arr2)` - Elementwise subtract **arr2** from **arr1**

`np.multiply(arr1,arr2)` - Elementwise multiply **arr1** by **arr2**

`np.divide(arr1,arr2)` - Elementwise divide **arr1** by **arr2**

`np.power(arr1,arr2)` - Elementwise raise **arr1** raised to the power of **arr2**

`np.array_equal(arr1,arr2)` - Returns **True** if the arrays have the same elements and shape

`np.sqrt(arr)` - Square root of each element in the array

`np.sin(arr)` - Sine of each element in the array

`np.log(arr)` - Natural log of each element in the array

`np.abs(arr)` - Absolute value of each element in the array

`np.ceil(arr)` - Rounds up to the nearest int

`np.floor(arr)` - Rounds down to the nearest int

`np.round(arr)` - Rounds to the nearest int

STATISTICS

`np.mean(arr,axis=0)` - Returns mean along specific axis

`arr.sum()` - Returns sum of **arr**

`arr.min()` - Returns minimum value of **arr**

`arr.max(axis=0)` - Returns maximum value of specific axis

`np.var(arr)` - Returns the variance of array

`np.std(arr,axis=1)` - Returns the standard deviation of specific axis

`arr.corrcoef()` - Returns correlation coefficient of array