Numpy Cheat Sheet

BrainStation®

Name	Description	Example
Creating Arrays		
`np.array()`	Creates an array with a list of values	`x = np.array([1, 2, 3])` #1d `a = np.array([[1, 2, 3], [4, 5, 6]])` #2d
`np.ones((i, j))`	Creates an array with all 1s. (i, j) is the shape of the array	`np.ones((3, 4))`
`np.eye(n)`	Creates an identity matrix of size n	`np.eye(5)` #creates 5x5 matrix
`np.full(shape)`	Creates an array or matrix of specified dimensions with a single value	`np.full((2, 3), 10)` #creates 2x3 matrix with all 10

`np.arange(n)`
`np.arange(i, j, k)`
	Creates an array of ascending integers from 0 to n or from i to j with step size k	`np.arange(5)`
		#returns array([0, 1, 2, 3, 4])
		
		`np.arange(0, 10, 2)`
		#returns array([0, 2, 4, 6, 8])

`np.linspace(i, j, k)`	Creates an array of k evenly divided values from i to j	`np.linspace(0, 0.5, 3)` #returns array([0., 0.25, 0.5])
`np.random.rand(i)` `np.random.rand(i, j)`	Creates an array or matrix of samples from the uniform distribution with values between 0 and 1	`np.random.rand(100)` #1D `np.random.rand(4, 5)` #2D
`np.random.rand(i)` `np.random.randn(i, j)`	Creates an array or matrix of samples from the standard normal distribution	`np.random.randn(100)` #1D `np.random.randn(4, 5)` #2D
`np.random.choice(a, size)`	Generates a random sample of specified size from a specified 1-D array or up to a given integer value	`np.random.choice(5, 3)` #returns array([0, 3, 4]) #5 is same as np.arange(5)
Properties & Type Conversion		
`array.shape`	Returns the shape of the array	`a.shape` #returns (2, 3)
`a.dtype`	Returns the type of the elements in an array	`a.dtype` #returns dtype(int32)
`a.astype(dtype)`	Converts the array 'a' to specified 'dtype'	`a.astype('int64')`
`np.argmax()` `np.argmin()`	Returns the index of the max or min value respectively	`np.argmax(a)` #returns 5 for 6 `np.argmin(a)` #returns 0 for 1
`a.tolist()`	Converts an array to a Python list	`a.tolist()` #[[1, 2, 3], [4, 5, 6]]

Numpy Cheat Sheet

BrainStation®

Statistics		
`np.mean(array, axis)`	Returns the mean along a specific axis	`np.mean(a, axis=0)` #row average
`np.percentile(a, q, axis)` `np.quantile(a, q, axis)`	Compute the q-th percentile / quantile value of the data along the specified axis	`np.percentile(a, 99)` #returns 5.95 `np.quantile(a, 0.5)` #returns 3.5
`np.std(array, axis)`	Returns standard deviation	`np.std(a, axis=1)`

`a.max(axis)`
`np.amax(a, axis)`
 	
`a.min(axis)`
`np.amin(a, axis)`
	Returns the max or min value in an array or along a specified axis	`a.max()` #returns 6
		`np.amax(a, 1)` #returns array([3, 6])
		
		`a.min()` #returns 1
		`np.amin(a, 1)` #return array([1, 4])

`np.histogram(array, bins)`	Compute the occurrences of the values from a flattened array that fall within each given bin. If bin is an integer, you will have equal size bins	`np.histogram([1, 2, 1], bins=[0, 1, 2, 3])` #returns array([0,2,1]) since 1 occurs twice between bin [1,2] and 2 occurs once between [2,3]
Mathematics		
`np.add(a, b)` `a+b`	Adds 'b' to each value in 'a' if 'b' is a scalar. Performs element-wise additions if 'a' and 'b' are arrays	`np.add(a, 1)` `np.add(array1, array2)`
`np.subtract(a, b)` `a-b`	Subtracts 'b' from each value in 'a' if 'b' is a scalar. Performs element-wise subtraction if 'a' from 'b' are arrays	`np.subtract(a, 1)` `np.subtract(array1, array2)`
`np.multiply(a, b)` `a*b`	Multiply each value in 'a' by 'b' if 'b' is a scalar. Perform element-wise multiplication if 'a' and 'b' are arrays	`np.multiply(a, 2)` `np.multiply(array1, array2)`
`np.divide(a, b)` `a/b`	Divides each value in 'a' by 'b' if 'b' is a scalar. Performs element-wise division is 'a' and 'b' are arrays	`np.divide(a, 2)` `np.divide(array1, array2)`
`np.array_equal(a, b)`	Returns True if array 'a' and 'b' are equal	`np.array_equal(a, array([1, 2, 3]))` #returns False

`np.unique(a)`	Returns the unique values in an array. 'return_counts=True' will return the counts of each unique value as well	`np.unique(array([5, 5, 6]))`
		#returns array([5, 6])
		
		`np.unique(array([5, 5, 6]), return_counts=True)`
		#returns array([5, 6], [2, 1])

`np.sqrt(a)`	Square root of each element in an array	`np.sqrt(a)`
`np.log(a)`	Natural log of each element in an array	`np.log(a)`
`np.exp(a)`	Returns exponential of all the elements in the input array	`np.exp([1, 3, 5])` #returns [2.718, 20.085, 148.413]

Numpy Cheat Sheet

BrainStation®

`np.abs(a)`	Returns absolute value	`np.abs(a)`
`np.ceil(a)`	Rounds up each element to nearest integer	`np.ceil(a)`
`np.floor(a)`	Rounds down each element to nearest int	`np.floor(a)`
`np.round(a)` `np.round(a, decimals)`	Rounds each element to nearest integer or decimal place if specified as second argument	`np.round(a)  #nearest integer` `np.round(a, 2)  #two decimals`
`np.matmul(a,b)`	Performs matrix multiplication for a,b	`np.matmul(matrix1, matrix2)`
Boolean Logic		
`np.logical_and(arr1, arr2)`	Returns whether conditions of arr1 AND arr2 equate to True	`np.logical_and(True, False)` `#returns False`
`np.logical_or(arr1, arr2)`	Returns whether conditions of arr1 OR arr2 equate to True	`np.logical_and(True, False)` `#returns True`
`np.all(a, axis)`	Returns whether all of elements in an array are True or along a specified axis	`np.all([[True, False], [True, False]])` `#returns True`
`np.any(a, axis)`	Returns whether any of the elements in an array are True or along a specified axis	`np.any([[True, False], [True, False]])` `#returns True`

`np.invert(a)`
`~`
	Inverts a boolean array	`np.invert(array([True, False, True]))`
		`#returns array([False, True, False])`
		
		`~array([True, False, True])`
		`#returns array([False, True, False])`

Transforming/Sorting		
`a.sort()`	Sorts array in ascending order	`a.sort()`

`a.flatten()`
`a.ravel()`
`np.ravel(a)`
	Flattens 2D array into 1D. Flatten returns a copy and ravel returns a view of original array	`a.flatten(a)`
		`#returns array([1, 2, 3, 4, 5, 6])`
		
		`np.ravel(a)`
		`#returns array([1, 2, 3, 4, 5, 6])`

`a.T`	Transposes array (switches rows and columns)	`a.T`
		`#array([[1, 4],`
		`#  [2, 5],`
		`#  [3, 6]])`

`np.reshape(a, (i, j))`
`a.reshape(i, j)`
	Reshapes array to i rows, j columns. Has to have the same size as the original. e.g. 2x3 matrix can only be reshaped into 3x2, 1x6, or 6x1.	`a.reshape(3, 2)`
		`#array([[1, 2],`
		`#  [3, 4],`
		`#  [5, 6]])`

`np.flip(a, axis)`	Reverses the order of elements along an axis	`np.flip(a, axis=1)`

`np.squeeze(a)`	Removes all dimensions of size 1 from an array	`b.shape # (1, 3, 3)`
		`c = np.squeeze(b)`
		`c.shape # (3, 3)`

Numpy Cheat Sheet

BrainStation®

`np.expand_dims()`	Opposite of squeeze, adds a dimension of 1 before (axis=0) or after (axis=1) those existing	`c.shape # (3, 3)`
		`b = np.expand_dims(c, axis=0)`
		`b.shape # (1, 3 ,3)`

Indexing / Slicing / Subsetting		
`a[row, column]` `a[index]`	Returns the element(s) at specified index or indices	`a[1, 2]  #returns 6`
`a[i] = n`	Assigns array element at index i with value 'n'	`a[1] = 4`
`a[i:j]`	Returns values from index i to j	`a[0:2]`
`a[i, :]`	Returns all the columns at row 'i'	`a[1, :] #returns array([4, 5, 6]`
`a[:, j]`	Returns all the rows at column 'j'	`a[:, 1] #returns array([2, 5])`

`a[a == n]`
`a[a < n]`
`a[a > n]`
	Returns values that return True for the boolean condition specified	`a[a < 5]    #returns values <5`
		`a[a == 5]   #returns values == 5`

`np.where(condition, a, b)`	Replaces values in array 'a' where 'condition' is True with the corresponding value in array 'b'	`np.where(a>3, a, a*10)` `#np.array([[1, 2, 3], [40, 50, 60]])`
Combining / Adding		

`np.concatenate([arr1, arr2], axis)`	Combines list of arrays row-wise or column-wise (axis=0, axis=1)	`np.concatenate([a1, a2], axis=0)`
		`#concatenates a1 and a2 row-wise`
		
		`np.concatenate([a1, a2], axis=1)`
		`#concatenates a1 and a2 column-wise`

`np.hstack([array1, array2])`
`np.vstack([array1, array2])`
	Stacks the arrays either horizontally 'hstack' or vertical 'vstack'. When stacking ensure the dimension of the rows or columns are equal.	`a1 = array([1, 2])`
		`a2 = array([3, 4])`
		
		`np.hstack(a1, a2) #array([1, 2, 3, 4])`
		`np.vstack(a1, a2) #array([[1, 2],`
		`[3, 4]])`

`np.append(array, values)`	Appends values to the end of array 'a'. Note: this will flatten a 2D array into 1D	`np.append(a, 3)` `#returns array([1, 2, 3, 4, 5, 6, 3])`
`np.insert(array, index, values)`	Inserts values before a specified index	`np.insert(a, 2, 0)` `#returns array([1, 2, 0, 3, 4, 5, 6, 3])`

`np.delete(array, index, axis)`	Deletes row or column at index	`np.delete(a, 2, axis=1)`
		`#returns array([[1, 2],`
		`[4, 6]])`

Data Import / Export		
`np.loadtxt()`	Load a text file from a relative path from the notebook's current working directory	`np.loadtxt('file.txt')`
`np.genfromtxt()`	Loads CSV file	`np.genfromtxt('file.csv', delimiter=',')`
`np.savetxt()`	Saves to a TXT or CSV file	`np.savetxt('newfile.csv')`