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')`

Pandas Cheat Sheet

BrainStation®

Name	Description	Example
	Pandas Data Structures	
`Pandas Series`	A Pandas Series is a one-dimensional array with axis labels, similar to a Numpy array but with an additional index. This index is constant and unchanging through the operations we apply to the Series.	`pd.Series([100000, 2300], index=['Grace', 'John'])`
`Pandas DataFrame`	A Pandas DataFrame is a collection of Pandas Series (columns) sharing a common index.	`pd.DataFrame( data= [[1, 2], [3, 4]], columns=['First', 'Second'] )`
	Import/Export Data	
`pd.read_csv()`	Read a CSV file into a Pandas dataframe from a local filesystem or URL.	`pd.read_csv('path/file_name.csv')`
`pd.read_excel()`	Read an Excel file into a Pandas dataframe (xls, xlsx, xlsm, xlsb, odf, ods and odt file extensions).	`pd.read_excel('path/file_name.xlsx')`

`pd.read_sql()`	Supports SQL query or database, where the parameter `conn` contains the connection sources and `sql` specifies the SQL query.	`conn = engine.connect()`
		`query = "SELECT * FROM trips"`
		
		`pd.read_sql(query, conn)`

`df.to_csv()`	Writes a Pandas dataframe to a CSV file.	`df.to_csv(r'path/file_name.csv', index=False)`
`df.to_excel()`	Writes a Pandas dataframe to an Excel file.	`df.to_excel(r'path/file_name.xlsx', sheet_name='Your_sheet_name', index=False)`

Pandas Cheat Sheet

BrainStation®

Index and Slice Pandas Objects		
By Indices		
`df.iloc[]`	Selects rows and columns by indices, in the order that they appear in the dataframe. Slicing is end-inclusive.	`df.iloc[:5, -3:]` `#returns first 5 rows and last 3 columns`
By Column Labels/Column Types		
`df.loc[]`	Selects based on column labels and row index values.	`df.loc[:, ['firstname', 'lastname']]` `# returns all rows in columns firstname and lastname`
	Passes an array or series of *True/False* values to the .loc indexer to select the rows where the values are *True*.	`df.loc[df['firstname'] == 'Jenna', :]` `#returns rows where the first_name column is 'Jenna'`
`df[[...]]`	Passes a list of column names into the data frame.	`df[['firstname', 'lastname']]` `# returns all rows in columns firstname and lastname`

`df.select_dtypes()`	Returns a subset of the dataframe's columns based on the column dtypes.	`df.select_dtypes(include='bool')`
		`# returns all boolean columns`
		
		`df.select_dtypes(exclude='int64')`
		`# returns all columns except integer columns`

Describe Data		
Example:	`pd.DataFrame(data=[[1, 'a'], [None, 'b']], columns=['First', 'Second'])`	
`df.shape`	Returns the number of rows and columns in the dataframe.	`df.shape  #returns (2, 2)`

`df.columns`	Returns the column labels of the dataframe.	`df.columns  #returns`
		`Index(['First', 'Second'],`
		`dtype='object')`

`df.index`	Returns the index of the dataframe.	`df.index  #returns` `RangeIndex(start=0, stop=2, step=1)`
`df.values`	Returns the dataframe values as a numpy array.	`df.values  #returns` `array([[1, 'a'], [None, 'b']])`

Pandas Cheat Sheet

BrainStation®

`df.count()` `df.nunique()`	Returns the count of non-null cells/number of unique observations for each column or row.	`df.count()` `#returns a series where index is ['First', 'Second'] and value is [1, 2]`
`df.value_counts()`	Returns a series containing counts of unique rows (for dataframe) or values (for series).	`df['First'].value_counts()` `# returns a series with index [1.] and value [1]`
`df.info()`	Returns a summary of the dataframe, including the column labels, count of non-null values and data types.	
`df.describe()`	Returns descriptive statistics of the numerical columns, excluding the null value.	
`df.plot()`	Returns a line graph containing data from every column in the dataframe; plot type can be specified by the `kind` argument and accepts the usual `matplotlib` arguments.	
Data Types		
`df.dtypes`	Returns the data type of each column in a dataframe.	`df.dtypes` `#returns a series where index is ['First', 'Second'] and value is ['int64', 'object']`
`df.astype()`	Casts a Pandas object to a specified dtype.	`df.astype(object).dtypes` `#returns a series where index is ['First', 'Second'] and value is ['object', 'object']`
Missing Values and Boolean Indicators		

`df.isin()`	Returns a boolean series or dataframe showing whether each element matches an item in a list of values, a dictionary or series.	`df.isin(['a', 1])`
		`# returns a boolean dataframe with values [[True, True],`
		`[False, False]]`

		`df.isin({'Second': ['b', 'c']})`
		`# returns a boolean dataframe with value [[False, False],`
		`[False, True]]`

`df.isna()`	Returns a boolean series or dataframe indicating if the values are NA. NA values, such as None or NaN, are mapped to True, and everything else is mapped to False.	`df.isna()`
		`# returns a boolean dataframe with values [[False, False],`
		`[True, False]]`

Pandas Cheat Sheet

BrainStation®

df.notna()	Inverse of `.isna()`. Not NA values are mapped to True.	**df.notna()** # returns a boolean dataframe with values [[True, True], [False, True]]
df.dropna()	Drop rows or columns which contain missing values. A threshold for the number of missing values (row-wise or column-wise) can also be specified.	**df.dropna(axis=0)** #drop rows (default) **df.dropna(axis=1)** # drop columns **df.dropna(thresh=10)** # drop rows with less than 10 non-NA values
df.fillna()	Fill missing (NaN) values, either with a single specified value or by a specific method	**df.fillna(0)** #fill missing with zeros **df.fillna(method='ffill')** #forward-fill

Modify Pandas Objects

Example Series: `x = pd.Series([1, 2, 3]), y = pd.Series([4, 5, 6])`

Example Dataframes:
```
df1 = pd.DataFrame([[1, 2], [3, 4]], columns=['col_a', 'col_b'])
df2 = pd.DataFrame([[5, 6], [7, 8]], columns=['col_a', 'col_b'])
```

df.append()	Concatenates two or more series.	**x.append(y)** # returns a series with values [1, 2, 3, 4, 5, 6]
	Appends the rows of the dataframe in the brackets to the end of the other, returning a new dataframe.	**df1.append(df2)** # returns a dataframe of shape 2x4
df.insert()	Inserts a new column into the dataframe at a specified position.	**df1.insert(loc=2,** **column='col_3',** **value=[7, 8, 9])** # returns a dataframe with an additional column named 'col_3'
df.drop()	Removes rows or columns by specifying indices and corresponding axis.	**df1.drop(columns=['col_a'])** # returns df1 without 'col_a' **df1.drop([0])** # returns df1 without the row at index 0
df.replace()	Replaces values given in first argument with second argument.	**x.replace(1, 3)** # returns [3, 1, 3]
df.where()	Replaces values where the condition is False.	**y.where(y < 5, 5)** # returns [5, 5, 6]

Pandas Cheat Sheet

BrainStation®

df.map()	Substitutes each value in a series with another value using a dictionary or a function.	**x.map({1: 3, 2: 3})** # returns a series with values [3, 3, 3] **x.map(lambda i: i+1)** # returns a series with values [2, 3, 4]
df.apply()	Applies a function along either the rows or columns of the dataframe.	**df1.apply(np.sum, axis=1)** # returns the sum of values by rows, that is [3, 7]
df.rename()	Renames the columns in a dataframe without changing the values.	**df1.rename(columns={'col_a': 'A',** **'col_b': 'B'})** # returns dataframe with columns renamed to “A” and “B”
pd.concat()	Concatenates pandas objects along a particular axis with optional set logic along the other axes.	**pd.concat([df1, df2])** # returns a dataframe joining df1 and df2 vertically (shape 4x2)
df.merge()	Merges dataframes or named series objects with a database-style join.	**df1.merge(df2, how='outer', on='col_a')** # returns a merged dataframe of df1 and df2 by outer join on col_a
Datetime and String Operations		
Example:	**a = pd.Series(['2020-10-21 09:30', '2010-10-22 12:30'])** **b = pd.Series(["ABC", "CDE"])**	
pd.to_datetime()	Converts datetime strings into Pandas datetime objects.	**a = pd.Series(['2020-10-21 09:30',** **'2010-10-22 12:30'])** **pd.to_datetime(a)** # returns a series containing Python datetime objects
df.dt.date **.time** **.year** **.month** **.dayofweek**	Converts datetime-like objects and extracts components.	**a = pd.to_datetime(a)** **a.dt.year** # returns [2020, 2010] **a.dt.dayofweek** # returns [6, 3]
df.str.contains **.slice** **.split** **.replace** **.extract**	Applies string methods to a dataframe or series of string dtype.	**b.str.slice(start=1)** # returns a series with ["BC","DE"] **b.str.contains("A")** # returns a series with [True, False]

Pandas Cheat Sheet

BrainStation®

Group By & Aggregation Functions		
Example:	`df = pd.DataFrame({'Student': ['Jim', 'Jim','Kelly', 'Pam'], 'Score': [78, 90, 98, 84]})` `grouped = df.groupby('Student')`	
`df.group_by()`	Groups dataframe by column(s) and returns a Pandas DataFrameGroupBy object.	

`.sum()`
`.mean()`
`.count()`
`.min()/.max()`
`.var()/.std()`
	Aggregate functions applicable to a grouped dataframe.	`grouped.mean()`
		`# returns a dataframe with index ['Jim', 'Kelly', 'Pam'] and value`
		`[84, 98, 84]`

`df.apply()`	Apply a function group-wise and combine the results together.	`grouped.apply(np.mean)` `# returns the same as grouped.mean()`
`df.transform()`	Apply a function group-wise and return an object with the same indexes as the original filled with the transformed values.	`grouped.transform(np.mean) # returns a series with values` `[84, 84, 98, 84]`
Statistical Functions		
Example:	`x = pd.Series([1.9, 7.1, 9.1]), y = pd.Series([0.8, 9, 12.1, 11.9])`	
`df.mean()`	Returns the means of the values for the given axis (`axis=0` ~ rows, `axis=1` ~ columns).	`x.mean() # returns 6.03`
`df.median()`	Returns the medians of the values for the given axis.	`x.median() # returns 7.1`
`df.var()`	Returns unbiased variances for the required axis.	`x.var()  # returns 13.81`
`df.std()`	Returns sample standard deviation for the given axis.	`x.std() # returns 3.72`
`df.cov()`	Returns the pairwise covariance of columns.	`x.cov(y) # return 21.7`
`df.corr()`	Returns the pairwise correlation of columns.	`x.corr(y) # returns 0.99`

`df.sample()`	Returns a random sample of items from an axis.	`y.sample(n=2, random_state=1)`
		`# returns a sample of 2`
		
		`y.sample(frac=0.5, random_state=1)`
		`# returns sample of half original size`