Table Functions and Methods:

In the examples in the left column,`np` refers to the NumPy module, as usual. Everything else is a function, a method, an example of an argument to a function or method, or an example of an object we might call the method on. For example,`tbl` refers to a table,`array` refers to an array, and`num` refers to a number.`array.item(0)` is an example call for the method`item`, and in that example,`array` is the name previously given to some array.

Name	Chapter	Description	Input	Output
`Table()`	6	Create an empty table, usually to extend with data	None	An empty **Table**
`Table().read_table(filename)`	6	Create a table from a data file	**string**: the name of the file	**Table** with the contents of the data file
`tbl.with_columns(name, values)` `tbl.with_columns(n1, v1, n2, v2,...)`	6	A table with an additional or replaced column or columns. `name` is a string for the name of a column, `values` is an array	1. **string**: the name of the new column; 2. **array**: the values in that column	**Table**: a copy of the original Table with the new columns added
`tbl.column(column_name_or_index)`	6	The values of a column (an array)	**string** *or* **int**: the column name or index	**array**: the values in that column
`tbl.num_rows`	6	Compute the number of rows in a table	None	**int**: the number of rows in the table
`tbl.num_columns`	6	Compute the number of columns in a table	None	**int**: the number of

Name	Chapter	Description	Input	Output
				columns in the table
`tbl.labels`	6	Lists the column labels in a table	None	**array**: the names of each column (as strings) in the table
`tbl.select(col1, col2, ...)`	6	Create a copy of a table with only some of the columns. Each column is the column name or index.	**string** *or* **int**: column name(s) or index(es)	**Table** with the selected columns
`tbl.drop(col1, col2, ...)`	6	Create a copy of a table without some of the columns. Each column is the column name or index.	**string** *or* **int**: column name(s) or index(es)	**Table** without the selected columns
`tbl.relabel(old_label, new_label)`	6	Modifies the existing table *in place*, changing the column heading in the first argument to the second	1. **string**: the old column name 2. **string**: the new column name	**Table**: a copy of the original with the changed label
`tbl.sort(column_name_or_index)`	6.1	Create a copy of a table sorted by the values in a column. Defaults to ascending order unless `descending = True` is included.	1. **string** *or* **int**: column index or name 2. (Optional) `descending = True`	
`tbl.where(column, predicate)`	6.2	Create a copy of a table with only the rows that match some *predicate* See `Table.where` predicates below.	1. **string** *or* **int**: column name or index 2. `are.(...)` predicate	
`tbl.take(row_indices)`	6.2	A table with only the rows at the given indices. `row_indices` is either an array of indices or an integer corresponding to one index.	**array** of ints: the indices of the rows to be included in the Table OR **int**: the index of the row to be included	**Table**: a copy of the original with only the

Name	Chapter	Description	Input	Output
				rows at the given indices
`tbl.scatter(x_column, y_column)`	[7]	Draws a scatter plot consisting of one point for each row of the table. Note that `x_column` and `y_column` must be strings specifying column names.	1. **string**: name of the column on the x-axis 2. **string**: name of the column on the y-axis	None: draws a scatter plot
`tbl.plot(x_column, y_column)`	[7]	Draw a line graph consisting of one point for each row of the table.	1. **string**: name of the column on the x-axis 2. **string**: name of the column on the y-axis	None: draws a line graph
`tbl.barh(categories)` `tbl.barh(categories, values)`	[7.1]	Displays a bar chart with bars for each category in a column, with height proportional to the corresponding frequency. values argument unnecessary if table has only a column of categories and a column of values.	1. **string**: name of the column with categories 2. (Optional) **string**: the name of the column with values for corresponding categories	None: draws a bar chart
`tbl.hist(column, unit, bins)`	[7.2]	Generates a histogram of the numerical values in a column. `unit` and `bins` are optional arguments, used to label the axes and group the values into intervals (bins), respectively. Bins have the form `[a, b)`, where `a` is included in the bin and `b` is not.	1. **string**: name of the column with categories 2. (Optional) **string**: units of x-axis 3. (Optional) **array** of ints/floats denoting bin boundaries	None: draws a histogram
`tbl.apply(function, column)`	[8.1]	Returns an array of values resulting from applying a function to each item in a column.	1. **function**: function to apply to column 2. **string**: name of the column to apply function to	**array**: contains an element for each value in the original column after

Name	Chapter	Description	Input	Output
				applying the function to it
`tbl.group(column_or_columns, func)`	8.2	Group rows by unique values or combinations of values in a column(s). Multiple columns must be entered in array or list form. Other values aggregated by count (default) or optional argument `func`.	1. **string** or **array of strings**: column(s) on which to group 2. (Optional) **function**: function to aggregate values in cells (defaults to count)	**Table**: a new Table
`tbl.pivot(col1, col2, values, collect)` `tbl.pivot(col1, col2)`	8.3	A pivot table where each unique value in `col1` has its own column and each unique value in c`col2` has its own row. Count or aggregate values from a third column, collect with some function. Default `values` and `collect` return counts in cells.	1. **string**: name of column whose unique values will make up columns of pivot table 2. **string**: name of column whose unique values will make up rows of pivot table 3. (Optional) **string**: name of column that describes the values of cell 4. (Optional) **function**: how the values are collected, e.g. `sum` or `np.mean`	**Table**: a new Table
`tblA.join(colA, tblB, colB)` `tblA.join(colA, tblB)`	8.4	Generate a table with the columns of tblA and tblB, containing rows for all values of a column that appear in both tables. Default `colB` is `colA`. `colA` and `colB` must be strings specifying column names.	1. **string**: name of column in tblA with values to join on 2. **Table**: other Table 3. (Optional) **string**: if column names are different between Tables, the name of the shared column in tblB	**Table**: a new Table
`tbl.sample(n)` `tbl.sample(n, with_replacement)`	10	A new table where `n` rows are randomly sampled from the original table. Default is with replacement. For	1. **int**: sample size 2. (Optional) `with_replacement=True`	**Table**: a new Table with `n` rows

Name	Chapter	Description	Input	Output
		sampling without replacement, use argument `with_replacement=False`. For a non-uniform sample, provide a third argument `weights=distribution` where `distribution` is an array or list containing the probability of each row.		

String Methods:

Name	Chapter	Description
`str.split(separator)`	N/A	Splits the string (`str`) into a list based on the `separator` that is passed in
`str.join(array)`	N/A	Combines each element of `array` into one string, with `str` being in-between each element
`str.replace(old_string, new_string)`	4.2.1	Replaces each occurrence of `old_string` in `str` with the value of `new_string`

Array Functions and Methods:

Name	Chapter	Description
`max(array)`	3.3	Returns the maximum value of an array
`min(array)`	3.3	Returns the minimum value of an array
`sum(array)`	3.3	Returns the sum of the values in an array
`abs(num), np.abs(array)`	3.3	Take the absolute value of number or each number in an array.

Name	Chapter	Description
`round(num), np.round(array)`	3.3	Round number or array of numbers to the nearest integer.
`len(array)`	3.3	Returns the length (number of elements) of an array
`make_array(val1, val2, ...)`	5	Makes a numpy array with the values passed in
`np.average(array) np.mean(array)`	5.1	Returns the mean value of an array
`np.diff(array)`	5.1	Returns a new array of size `len(arr)-1` with elements equal to the difference between adjacent elements; val_2 - val_1, val_3 - val_2, etc.
`np.sqrt(array)`	5.1	Returns an array with the square root of each element
`np.arange(start, stop, step)` `np.arange(start, stop)` `np.arange(stop)`	5.2	An array of numbers starting with `start`, going up in increments of `step`, and going up to but excluding `stop`. When `start` and/or `step` are left out, default values are used in their place. Default step is 1; default start is 0.
`array.item(index)`	5.3	Returns the i-th item in an array (remember Python indices start at 0!)
`np.random.choice(array, n)` `np.random.choice(array)`	9	Picks one (by default) or some number 'n' of items from an array at random. By default, with replacement.
`np.count_nonzero(array)`	9	Returns the number of non-zero (or `True`) elements in an array.
`np.append(array, item)`	9.2	Returns a copy of the input array with `item` (must be the same type as the other entries in the array) appended to the end.
`percentile(percentile, array)`	12.1	Returns the corresponding percentile of an array.

Table.where Predicates:

Any of these predicates can be negated by adding `not_` in front of them, e.g. `are.not_equal_to(Z)` or `are.not_containing(S)`.

Predicate	Description
`are.equal_to(Z)`	Equal to `Z`
`are.above(x)`	Greater than `x`
`are.above_or_equal_to(x)`	Greater than or equal to `x`

Predicate	Description
`are.below(x)`	Less than `x`
`are.below_or_equal_to(x)`	Less than or equal to `x`
`are.between(x,y)`	Greater than or equal to `x` and less than `y`
`are.between_or_equal_to(x,y)`	Greater than or equal to `x`, and less than or equal to `y`
`are.contained_in(A)`	Is a substring of `A` (if `A` is a string) or an element of `A` (if `A` is a list/array)
`are.containing(S)`	Contains the string `S`
`are.strictly_between(x,y)`	Greater than `x` and less than `y`

Miscellaneous Functions:

These are functions in the `datascience` library that are used in the course that don't fall into any of the categories above.

Name	Chapter	Description	Input	Output
`sample_proportions(sample_size, model_proportions)`	11.1	`Sample_size` should be an integer, `model_proportions` an array of probabilities that sum up to 1. The function samples `sample_size` objects from the distribution specified by `model_proportions`. It returns an array with the same size as `model_proportions`. Each item in the array corresponds to the proportion of times it was sampled out of the `sample_size` times.	1. **int**: sample size 2. **array**: an array of proportions that should sum to 1	**array**: each item corresponds to the proportion of times that corresponding item was sampled from **model_proportions** in **sample_size** draws, should sum to 1
`minimize(function)`	15.4	Returns an array of values such that if each value in the array was passed into `function` as arguments,	**function**: name of a function that	**array**: An array in which each element corresponds to an argument that minimizes the

Name	Chapter	Description	Input	Output
		it would minimize the output value of `function`.	will be minimized.	output of the function. Values in the array are listed based on the order they are passed into the function; the first element in the array is also going to be the first value passed into the function.

Created by Nishant Kheterpal and Jessica Hu (http://data8.org/fa20/python-reference.html)