CS 133 Cheat Sheet of Methods

Data Science Python Reference

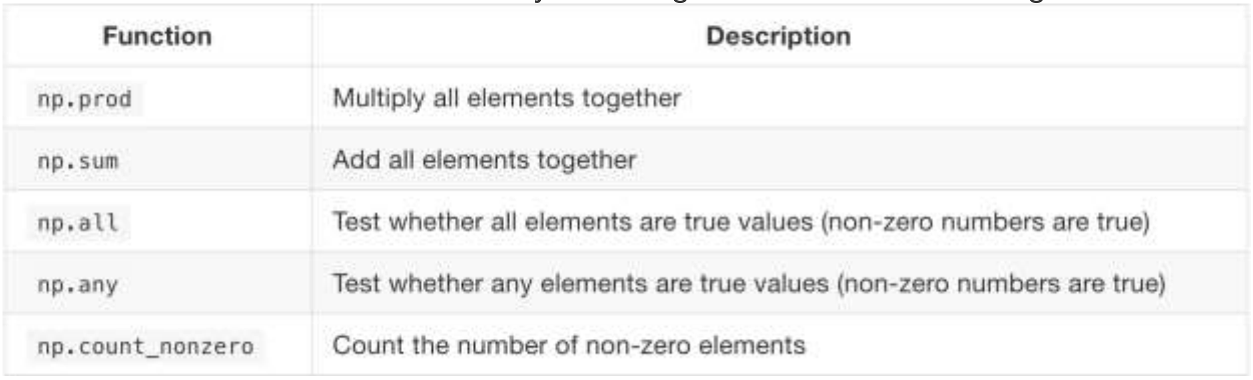
Python's Built in Functions

Math Module Functions

String Methods	What it does	Example
String.upper()	Casts the letters in the string all capital letters	"loud".upper() LOUD
String.lower()	Casts the letters in the string to lower case letters	"HaPPy".lower() Happy
String.replace('str', 'str')	Replaces the first string passed in as a parameter with the second string passed in	'hitchhiker'.replace('hi', 'ma') Matchmaker

NumPy Reference

Each of these functions takes an array as an argument and returns a single value.



Function	Description
`np.prod`	Multiply all elements together
`np.sum`	Add all elements together
`np.all`	Test whether all elements are true values (non-zero numbers are true)
`np.any`	Test whether any elements are true values (non-zero numbers are true)
`np.count_nonzero`	Count the number of non-zero elements

Each of these functions takes an array as an argument and returns an array of values.

Function	Description
`np.diff`	Difference between adjacent elements
`np.round`	Round each number to the nearest integer (whole number)
`np.cumprod`	A cumulative product: for each element, multiply all elements so far
`np.cumsum`	A cumulative sum: for each element, add all elements so far
`np.exp`	Exponentiate each element
`np.log`	Take the natural logarithm of each element
`np.sqrt`	Take the square root of each element
`np.sort`	Sort the elements

Each of these functions takes an array of strings and returns an array.

Function	Description
`np.char.lower`	Lowercase each element
`np.char.upper`	Uppercase each element
`np.char.strip`	Remove spaces at the beginning or end of each element
`np.char.isalpha`	Whether each element is only letters (no numbers or symbols)
`np.char.isnumeric`	Whether each element is only numeric (no letters)

Each of these functions takes both an array of strings and a *search string*; each returns an array.

Function	Description
`np.char.count`	Count the number of times a search string appears among the elements of an array
`np.char.find`	The position within each element that a search string is found first
`np.char.rfind`	The position within each element that a search string is found last
`np.char.startswith`	Whether each element starts with the search string

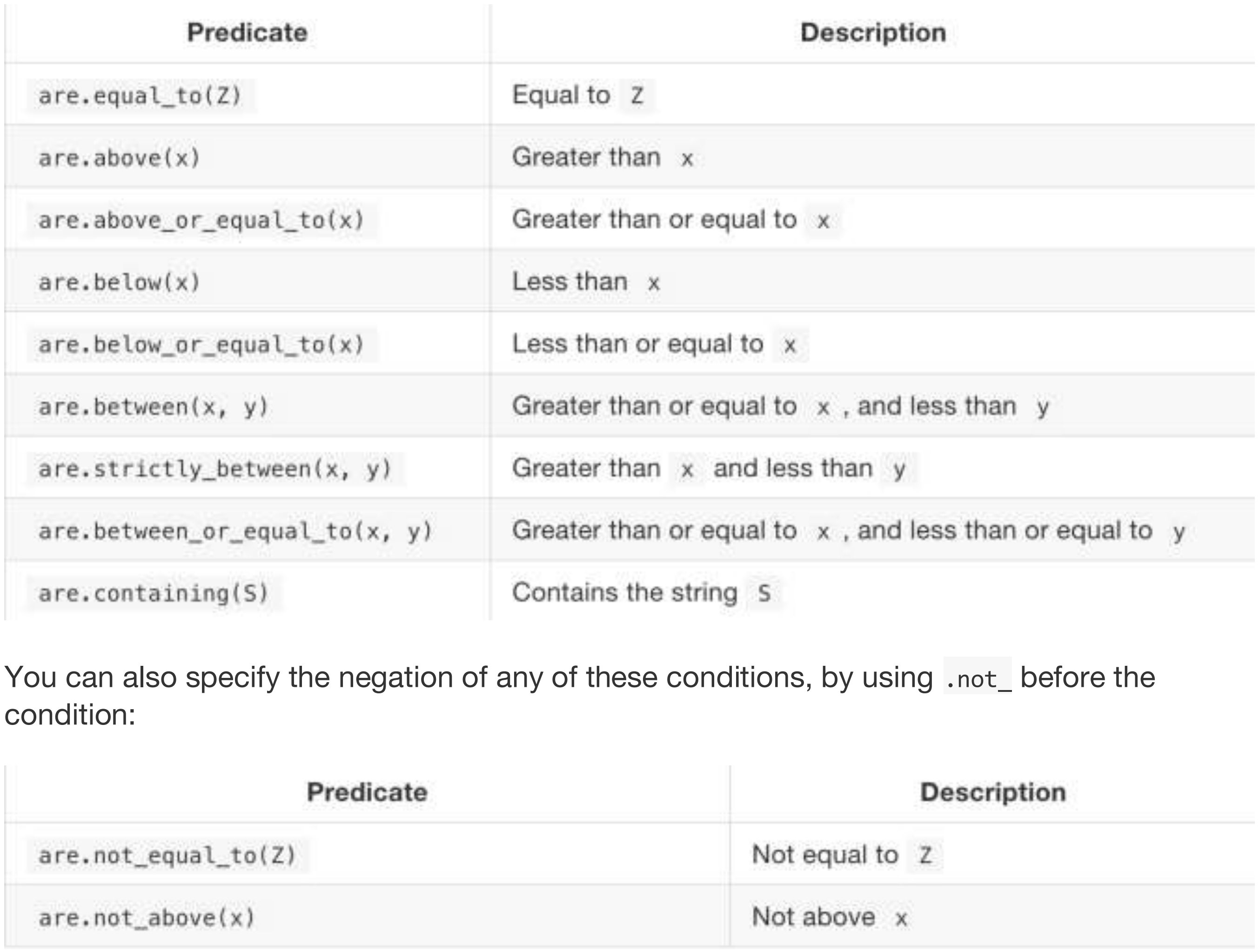
Making a table:

Method/Function Call	What it does
Table().with_columns(*column_name,* make_array())	Creates a table with the columns that are passed in as parameters

table_name.with_columns(*column_name,* make_array())	Adds a column to the already created table with the name *table_name*
Table.read_table(‘*csv_file_name*’)	Reads in a csv file and creates a table with that data
table_name.num_columns	Returns the number of columns in the table with the given *table_name*
table_name.num_rows	Returns the number of rows in the table with the given *table_name*
table_name.labels	Returns a list of the column names
table_name.relabeled(‘str’, ‘str’)	Relabels the column with the first string name with the new string name that is the second parameter
table_name.column(‘*column_name’/index*)	Returns the array that has the values in the designated column
.item(index)	Returns the item at the specified index

Method/Function Call	What it does
table_name.select(‘*column’/index*, ‘*column/index’*)	Selects whatever column you pass in as a parameter and creates a new table
table_name.drop(‘column_name’/index)	You can pass in as many parameters as need be but drops the columns on the table with the wanted request. Returns a new table
table_name.sort(‘*column_name’*) *table_name*.sort(‘*column_name’, descending = True*)	Sorts the specified column numerically or alphabetically
table_name.sample(*quantity*)	Randomly selects an element out of table with replacement however many times you specify (quantity).

Note that `x` and `y` are numbers, `STRING` is a string, and `Z` is either a number or a string; you have to specify these depending on the feature you want.



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`
`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.strictly_between(x, y)`	Greater than `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.containing(S)`	Contains the string `S`

You can also specify the negation of any of these conditions, by using `.not_` before the condition:

Predicate	Description
`are.not_equal_to(Z)`	Not equal to `Z`
`are.not_above(x)`	Not above `x`

Creating visual representations:

Method/Function call	What it does
table_name.scatter*(“name of column 1”, “name of column 2”)*	Creates a scatter plot with column 1 on x-axis and column 2 on y-axis
table_name.plot(“name of column 1”, name of column 2”)	Creates a line plot with column 1 on x-axis and column 2 on the y-axis
table_name.barh*(“category”, “numerical data”)*	Creates a horizontal bar graph
table_name.bar*(“category”, “numerical data”)*	Creates a vertical bar graph
table_name.hist*(‘name of bins’, unit=”unit name”)*	Creates a histogram