NumPy Cheat Sheet

This cheat sheet offers a quick and practical reference for essential NumPy commands, focusing on array creation, manipulation, and analysis, using examples drawn from the NYC Taxis Dataset. It covers critical topics such as importing data from files, creating and reshaping arrays, and performing scalar and vector math.

You'll also find easy-to-follow instructions on inspecting array properties, combining and splitting arrays, Boolean filtering, and computing statistics like mean, variance, and standard deviation. Whether you're analyzing 1D or 2D arrays, this cheat sheet helps you leverage NumPy's capabilities for efficient data handling.

Designed to be clear and actionable, this reference ensures that you can quickly apply NumPy's powerful array operations in your data analysis workflow.

Table of Contents

Importing Data

Syntax for	How to use	Explained
IMPORT	`import numpy as np`	Imports NumPy using its standard alias, `np`
LOADTXT	`np.loadtxt('file.txt')`	Create an array from a `.txt` file
GENFROMTXT	`np.genfromtxt('file.csv', delimiter=',')`	Create an array from a `.csv` file
SAVETXT	`np.savetxt('file.txt', arr, delimiter=' ')`	Writes an array to a `.txt` file
	`np.savetxt('file.csv', arr, delimiter=',')`	Writes an array to a `.csv` file

Creating Arrays

Syntax for	How to use	Explained
ARRAY	`arr = np.array([1, 2, 3])`	Create a 1D array
	`arr = np.array([(1, 2, 3), (4, 5, 6)])`	Create a 2D array
ZEROS	`arr = np.zeros(3)`	1D array of length 3; all values set to 0
ONES	`arr = np.ones((3, 4))`	3x4 array with all values set to 1
EYE	`arr = np.eye(5)`	5x5 array of 0 with 1 on diagonal (identity matrix)

Syntax for	How to use	Explained
LINSPACE	`arr = np.linspace(0, 100, 6)`	Array of 6 evenly divided values from 0 to 100 (`[0, 20, 40, 60, 80, 100]`)
ARANGE	`arr = np.arange(0, 10, 3)`	Array of values from 0 to less than 10 with step 3 (`[0, 3, 6, 9]`)
FULL	`arr = np.full((2, 3), 8)`	2x3 array with all values set to 8
RAND	`arr = np.random.rand(4, 5)`	4x5 array of random floats between 0 and 1
	`arr = np.random.rand(6, 7) * 100`	6x7 array of random floats between 0-100
RANDINT	`arr = np.random.randint(5, size=(2, 3))`	2x3 array with random integers between 0 and 4

Inspecting Properties

Syntax for	How to use	Explained
ASTYPE	`arr.astype(dtype)`	Convert `arr` elements to type `dtype`
TOLIST	`arr.tolist()`	Convert `arr` to a Python list
INFO	`np.info(np.eye)`	View documentation for `np.eye`
SIZE	`arr.size`	Returns number of elements in `arr`
SHAPE	`arr.shape`	Returns dimensions of `arr` (rows, columns)
DTYPE	`arr.dtype`	Returns type of elements in `arr`

Copying, Sorting, & Reshaping

Syntax for	How to use	Explained
COPY	`np.copy(arr)`	Copies `arr` to new memory
VIEW	`arr.view(dtype)`	Creates view of `arr` elements with type `dtype`
SORT	`arr.sort()`	Sorts `arr`
SORT	`arr.sort(axis=0)`	Sorts specific axis of `arr`
FLATTEN	`two_d_arr.flatten()`	Flattens 2D array `two_d_arr` to 1D
T	`arr.T`	Transposes `arr` *(rows become columns and vice versa)*
RESHAPE	`arr.reshape(3, 4)`	Reshapes `arr` to 3 rows, 4 columns without changing data
RESIZE	`arr.resize((5, 6))`	Changes `arr` shape to 5x6 and fills new values with 0

Adding & Removing Elements

Syntax for	How to use	Explained
APPEND	`np.append(arr, values)`	Appends values to end of `arr`
INSERT	`np.insert(arr, 2, values)`	Inserts values into `arr` before index 2
DELETE	`np.delete(arr, 3, axis=0)`	Deletes row on index 3 of `arr`
	`np.delete(arr, 4, axis=1)`	Removes the 5th column from `arr`

Combining & Splitting

Syntax for	How to use	Explained
CONCATENATE	`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`
SPLIT	`np.split(arr, 3)`	Splits `arr` into 3 sub-arrays
HSPLIT	`np.hsplit(arr, 5)`	Splits `arr` horizontally on the index 5

Indexing & Slicing

Syntax for	How to use	Explained
INDEXING	`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
SLICING	`arr[0:3]`	Returns the elements at indices 0, 1, 2
	`arr[0:3, 4]`	Returns the elements on rows 0, 1, 2 in column index 4
	`arr[:2]`	Returns the elements at indices 0, 1
	`arr[:, 1]`	Returns column index 1, all rows
CONDITIONAL STATEMENTS	`arr < 5`	Returns an array of boolean values

Indexing & Slicing

Syntax for	How to use	Explained
CONDITIONAL STATEMENTS	`(arr1 < 3) & (arr2 > 5)`	To be `True`, both must be `True`
	`~arr`	Inverts a boolean array
	`arr[arr < 5]`	Returns array elements less than 5
	`(arr1 < 3) | (arr2 > 5)`	To be `True`, at least one must be `True`

Scalar Math

Syntax for	How to use	Explained
ADD	`np.add(arr, 1)`	Add 1 to each array element
SUBTRACT	`np.subtract(arr, 2)`	Subtract 2 from each array element
MULTIPLY	`np.multiply(arr, 3)`	Multiply each array element by 3
DIVIDE	`np.divide(arr, 4)`	Divide each array element by 4 (returns `np.nan` for division by zero)
POWER	`np.power(arr, 5)`	Raise each array element to the power of 5

Vector Math

Syntax for	How to use	Explained
ADD	`np.add(arr1, arr2)`	Elementwise add `arr1` to `arr2`
SUBTRACT	`np.subtract(arr1, arr2)`	Elementwise subtract `arr2` from `arr1`
MULTIPLY	`np.multiply(arr1, arr2)`	Elementwise multiply `arr1` by `arr2`
DIVIDE	`np.divide(arr1, arr2)`	Elementwise divide `arr1` by `arr2`
POWER	`np.power(arr1, arr2)`	Elementwise, raise `arr1` to the power of `arr2`
ARRAY_EQUAL	`np.array_equal(arr1, arr2)`	Returns `True` if the arrays have the same elements and shape
SQRT	`np.sqrt(arr)`	Square root of each element in the array
SIN	`np.sin(arr)`	Sine of each element in the array
LOG	`np.log(arr)`	Natural log of each element in the array
ABS	`np.abs(arr)`	Absolute value of each element in the array
CEIL	`np.ceil(arr)`	Rounds up each element to the nearest integer

Vector Math

Syntax for	How to use	Explained
FLOOR	`np.floor(arr)`	Rounds down each element to the nearest integer
ROUND	`np.round(arr)`	Rounds each element to the nearest integer

Statistics

Syntax for	How to use	Explained
MEAN	`np.mean(arr, axis=0)`	Returns mean of `arr` along specified axis
SUM	`arr.sum()`	Returns the sum of elements in `arr`
MIN	`arr.min()`	Returns minimum value of `arr`
MAX	`arr.max(axis=0)`	Returns maximum value of `arr` along specified axis
VAR	`np.var(arr)`	Returns the variance of `arr`
STD	`np.std(arr, axis=1)`	Returns the standard deviation of `arr` along specified axis
CORRCOEF	`arr.corrcoef()`	Returns correlation coefficient of `arr`

Working with Data

Syntax for	How to use	Explained

CREATING NDARRAYS	`import numpy as np`
	`array_1d = np.array([1, 2, 3, 4, 5])`
	`array_2d = np.array([[1, 2, 3], [4, 5, 6]])`
		Create a 1D or 2D `ndarray`

CONVERTING A LIST OF LISTS	`import csv`
	`f = open("nyc_taxis.csv", "r")`
	`taxi_list = list(csv.reader(f))`
	`taxi = np.array(taxi_list)`
		Convert a list of lists into a 2D `ndarray`

SELECTING ROWS	`second_row = taxi[1]`	Select the second row in `taxi`
	`all_but_first_row = taxi[1:]`	Select all rows from the second row onward in `taxi`
	`fifth_row_second_column = taxi[4, 1]`	Select the element from the fifth row and second column in `taxi`
SELECTING COLUMNS	`second_column = taxi[:, 1]`	Select all values from the second column in `taxi`

	`second_third_columns = taxi[:, 1:3]`
	`cols = [1, 3, 5]`
	`second_fourth_sixth_columns = taxi[:, cols]`
		Select the second and third columns, then the second, fourth, and sixth columns in `taxi`

	`twod_slice = taxi[1:4, :3]`	Select a slice of rows 2 to 4 and columns 1 to 3 in `taxi`

Working with Data

Syntax for	How to use	Explained
VECTOR OPERATIONS	`vector_a + vector_b`	Element-wise addition of two `ndarray` objects
	`vector_a - vector_b`	Element-wise subtraction of two `ndarray` objects
	`vector_a * vector_b`	Element-wise multiplication of two `ndarray` objects
	`vector_a / vector_b`	Element-wise division of two `ndarray` objects
STATISTICS FOR 1D NDARRAYS	`array_1d.min()`	Return the minimum value of `array_1d`
	`array_1d.max()`	Return the maximum value of `array_1d`
	`array_1d.mean()`	Calculate the average of values in `array_1d`
	`array_1d.sum()`	Calculate the sum of the values in `array_1d`
STATISTICS FOR 2D NDARRAYS	`array_2d.max()`	Return the maximum value for the entire `array_2d`
	`array_2d.max(axis=1)   # returns a 1D ndarray`	Return the maximum value in each row in `array_2d`
	`array_2d.max(axis=0)   # returns a 1D ndarray`	Return the maximum value in each column in `array_2d`

Syntax for	How to use	Explained

CREATING AN NDARRAY FROM CSV FILE	`import numpy as np`
	`taxi = np.genfromtxt('nyc_taxis.csv',`
	`       delimiter=',', skip_header=1)`
		Load data from the `nyc_taxis.csv` file into an `ndarray`, skipping the header row

WORKING WITH BOOLEAN ARRAYS	`np.array([2, 4, 6, 8]) < 5`	Create a Boolean array for elements less than 5

	`a = np.array([2, 4, 6, 8])`
	`filter = a < 5`
	`a[filter]  # returns [2, 4]`
		Use Boolean filtering to return elements less than 5 from an `ndarray`

	`tip_amount = taxi[:, 12]`
	`tip_bool = tip_amount > 50`
	`top_tips = taxi[tip_bool, 5:14]`
		Use Boolean filtering to return rows with `tip_amount > 50` and columns 6 to 14

ASSIGNING NDARRAY VALUES	`taxi[1066, 5] = 1`
	`taxi[:, 0] = 16`
	`taxi[550:552, 7] = taxi[:, 7].mean()`
		Assign values to specific elements, a column, and a slice in `taxi`

	`taxi[taxi[:, 5] == 2, 15] = 1`	Use Boolean indexing to assign a value of 1 in column index 15 to rows where the 6th column equals 2