NumPy Cheat Sheet

NumPy is the fundamental package for scientific computing with Python.

This cheat sheet acts as a intro to Python for data science. ***Contact me here for typos or suggestions, and - of course - fork and tune it to your taste!***

Index

Basics

One of the most commonly used functions of NumPy are *NumPy arrays*: The essential difference between *lists* and *NumPy arrays* is functionality and speed. *lists* give you basic operation, but *NumPy* adds FFTs, convolutions, fast searching, basic statistics, linear algebra, histograms, etc.

The most important difference for data science is the ability to do **element-wise calculations** with *NumPy arrays*.

`axis 0` always refers to row

`axis 1` always refers to column

Operator	Description	Documentation
`np.array([1,2,3])`	1d array	link
`np.array([(1,2,3),(4,5,6)])`	2d array	see above
`np.arange(start,stop,step)`	range array	link

Placeholders

Operators	Description	Documentation
`np.linspace(0,2,9)`	Add evenly spaced values btw interval to array of length	link
`np.zeros((1,2))`	Create and array filled with zeros	link
`np.ones((1,2))`	Creates an array filled with ones	link
`np.random.random((5,5))`	Creates random array	link
`np.empty((2,2))`	Creates an empty array	link

Examples

```
# 1 dimensional
x = np.array([1,2,3])
# 2 dimensional
y = np.array([(1,2,3),(4,5,6)])

x = np.arange(3)
>>> array([0, 1, 2])

y = np.arange(3.0)
>>> array([ 0.,  1.,  2.])

x = np.arange(3,7)
>>> array([3, 4, 5, 6])

y = np.arange(3,7,2)
>>> array([3, 5])
```

Array

Array Properties

Syntax	Description	Documentation
`array.shape`	Dimensions (Rows,Columns)	link
`len(array)`	Length of Array	link
`array.ndim`	Number of Array Dimensions	link
`array.size`	Number of Array Elements	link
`array.dtype`	Data Type	link
`array.astype(type)`	Converts to Data Type	link
`type(array)`	Type of Array	link

Copying/Sorting

Operators	Descriptions	Documentation
`np.copy(array)`	Creates copy of array	link
`other_array = ...`	Creates deep copy of array	see above

`other = array.copy()`	Creates deep copy of array	see above
`array.sort()`	Sorts an array	link
`array.sort(axis=0)`	Sorts axis of array	see above

Examples

```
# Sort sorts in ascending order
y = np.array([10, 9, 8, 7, 6, 5, 4, 3, 2, 1])
y.sort()
print(y)
>>> [ 1  2  3  4  5  6  7  8  9  10]
```

Array Manipulation Routines

Adding or Removing Elements

Operator	Description	Documentation
`np.append(a,b)`	Append items to array	link
`np.insert(array, 1, 2, axis)`	Insert items into array at axis 0 or 1	link
`array.resize((2,4))`	Resize array to shape(2,4)	link
`np.delete(array,1,axis)`	Deletes items from array	link

Combining Arrays

Operator	Description	Documentation
`np.concatenate((a,b),axis=0)`	Concatenates 2 arrays, adds to end	link
`np.vstack((a,b))`	Stack array row-wise	link
`np.hstack((a,b))`	Stack array column wise	link

Splitting Arrays

Operator	Description	Documentation
`numpy.split()`		link
`np.array_split(array, 3)`	Split an array in sub-arrays of (nearly) identical size	link
`numpy.hsplit(array, 3)`	Split the array horizontally at 3rd index	link

More

Operator	Description	Documentation
`other = ndarray.flatten()`	Flattens a 2d array to 1d	link
`array = np.transpose(other)` `array.T`	Transpose array	link

Mathematics

Operations

Operator	Description	Documentation
`np.add(x,y)`	Addition	link
`np.substract(x,y)`	Subtraction	link
`np.divide(x,y)`	Division	link
`np.multiply(x,y)`	Multiplication	link
`np.sqrt(x)`	Square Root	link
`np.sin(x)`	Element-wise sine	link
`np.cos(x)`	Element-wise cosine	link
`np.log(x)`	Element-wise natural log	link
`np.dot(x,y)`	Dot product	link

Remember: NumPy array operations work element-wise.

Example

```
# If a 1d array is added to a 2d array (or the other way), NumPy
# chooses the array with smaller dimension and adds it to the one
# with bigger dimension
a = np.array([1, 2, 3])
b = np.array([(1, 2, 3), (4, 5, 6)])
print(np.add(a, b))
>>> [[2 4 6]
     [5 7 9]]
```

Comparison

Operator	Description	Documentation
`==`	Equal	link
`!=`	Not equal	link
`<`	Smaller than	link
`>`	Greater than	link
`<=`	Smaller than or equal	link
`>=`	Greater than or equal	link
`np.array_equal(x,y)`	Array-wise comparison	link

Example

```
# Using comparison operators will create boolean NumPy arrays
z = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10])
c = z < 6
print(c)
>>> [ True  True  True  True  True False False False False False]
```

Basic Statistics

Operator	Description	Documentation
`array.mean()` `np.mean(array)`	Mean	link

`np.mean(array)`		
`np.median(array)`	Median	link
`array.corrcoef()`	Correlation Coefficient	link
`array.std(array)`	Standard Deviation	link

More

Operator	Description	Documentation
`array.sum()`	Array-wise sum	link
`array.min()`	Array-wise minimum value	link
`array.max(axis=0)`	Maximum value of specified axis	
`array.cumsum(axis=0)`	Cumulative sum of specified axis	link

Slicing and Subsetting

Operator	Description	Documentation
`array[i]`	1d array at index i	link
`array[i,j]`	2d array at index[i][j]	see above
`array[i<4]`	Boolean Indexing, see Tricks	see above
`array[0:3]`	Select items of index 0, 1 and 2	see above
`array[0:2,1]`	Select items of rows 0 and 1 at column 1	see above
`array[:1]`	Select items of row 0 (equals array[0:1, :])	see above
`array[1:2, :]`	Select items of row 1	see above
[comment]: <> (	`array[1,...]`	equals array[1,:,:]	see above	)

`array[ : :-1]`	Reverses `array`	see above

Examples <a name"exp">

```
b = np.array([(1, 2, 3), (4, 5, 6)])

# The index *before* the comma refers to *rows*,
# the index *after* the comma refers to *columns*
print(b[0:1, 2])
>>> [3]

print(b[:len(b), 2])
>>> [3 6]

print(b[0, :])
>>> [1 2 3]

print(b[0, 2:])
>>> [3]

print(b[:, 0])
>>> [1 4]

c = np.array([(1, 2, 3), (4, 5, 6)])
d = c[1:2, 0:2]
print(d)
>>> [[4 5]]
```

Tricks

This is a growing list of examples. Know a good trick? Let me know here or fork it and create a pull request.

boolean indexing (available as separate `.py` file here

```
# Index trick when working with two np-arrays
a = np.array([1,2,3,6,1,4,1])
b = np.array([5,6,7,8,3,1,2])

# Only saves a at index where b == 1
other_a = a[b == 1]
#Saves every spot in a except at index where b != 1
other_other_a = a[b != 1]
```

```
x = np.array([4,6,8,1,2,6,9])
y = x > 5
print(x[y])
>>> [6 8 6 9]

# Even shorter
x = np.array([1, 2, 3, 4, 4, 35, 212, 5, 5, 6])
print(x[x < 5])
>>> [1 2 3 4 4]
```

Credits

Datacamp,
Quandl & Official docs