Python Cheatsheet

Steve Young

ABSTRACT: Everything I know about Python.

Contents

TODO: make sidebar marker telling what python library command depends on (and needs to be loaded beforehand)

1 Python

1.1 Lists and Dicts

1.1.1 Lists

! With list name `foo`:

Init list of length <n> with <constant>: `foo = [<constant>] * <n>`

Add <val> to end of list: `foo.append(<val>)`

Remove list item at pos <n> (or end if <n> unspecified): `foo.pop(<n>)`

Insert <val> into list before position <n>: `foo.insert(<n>, <val>)`

Sum of list: `sum(foo)`

Get indexed elements as tuples (index, el) from iterable: `enumerate(iterable)`

Same as above, with index starting at <startval>: `enumerate(iterable, startval)`

Append <list2> to end of list: `foo.extend(<list2>)`

1.1.2 Dicts

! Dictionaries are key-value stores, *i.e.* hashtables. With dictionary name `foo`:

Add <key>-<value> pair: `foo[<key>] = <value>`

Iterate though <key>-<value> pairs: `for (key, value) in foo.items()`

Iterate though keys: `for key in foo.keys()`

Iterate though values: `for value in foo.values()`

Test if <key> in dict: `<key> in foo`

1.2 Sorting

! With sortable-thing name `foo`:

Sort (modify in place): `foo.sort()`

Sort (make a copy): `foo.sorted()`

Specify field to sort by (here by 2nd element of tuple): `foo.sorted(key=lambda x: x[1])`

Sort in reverse order: `foo.sorted(reverse=True)`

1.3 Data Processing

Read data from json file: `with open('data.json','r') as f: data = json.load(f)`

Write data to json file: `with open('data.json','w') as f: data = json.dump(f)`

Regex processing of text, `re` **package:** look at https://docs.python.org/3/library/re.html

1.4 Misc

Return integer representing Unicode <char>: `ord(<char>)`

1.5 Regex (re)

List all files in dir that match some regular expression:

```
regex = re.compile(<regex string>)
filepaths = [f for f in os.listdir(<dir>) if re.match(regex, f)]
```

2 Numpy

! Using `import numpy as np`:

2.1 Data Processing

Import data from csv file: `np.genfromtxt('filename', delimiter=',')`

2.2 Number generation

Constant matrix of <val>: `np.full(shape, <val>)`

Matrix of ones/zeros: `np.ones(shape)`, `np.zeros(shape)`

Id matrix: `np.eye(dim)`

Uniform dist on (low,high): `np.random.uniform(low, high, numsamps or shape)`

Uniform dist on (0,1) with given dims: `np.random.rand(d1, d2, ..)`

Normal dist: `np.random.normal(mean, stddev, numsamps)`

Normal dist on with given dims: `np.random.randn(d1, d2, ..)`

Multivariate normal: `np.random.multivariate_normal(..args)`

Random permutation of elements in ndarray: `np.random.permutation(NDArray)`

Permute elements of (range or ndarray) *in place*: `np.random.shuffle(int or NDArray)`

Integers over specified range: `np.arange(beg, end)`

Evenly spaced numbers over range w/ interval `stepsize`**:** `np.arange(beg, end, stepsize)`

`numvals` **Evenly spaced numbers over range:** `np.linspace(beg, end, numvals)`

2.3 NDArray handling

! NDArrays are naturally *row vectors*, and of shape $(m,)$.

Reshape array: `np.reshape(NDArray, tuple of shape)`

2.4 NDArray ops

max/min element of array: `np.max(NDArray)`, `np.min(NDArray)`

Index of max/min element of array: `np.argmax(NDArray)`, `np.argmin(NDArray)`

Fill diagonal of sq matrix: `np.fill_diagonal(NDArray, val)`

Make diag matrix w/ <vec> as diagonal: `np.diag(<vec>)`

Round elements to nearest int: `np.rint(NDArray)`

Return bin counts in histogram: `np.histogram(NDArray, binboundaries)`

<n>th difference of array: [1] `np.diff(NDArray, <n>)`

[1] Think transforming array of tick prices into array of tick prices *changes*

2.5 Linear Algebra

Inverse matrix: `np.linalg.inv(square NDArray)`

Transpose matrix: `np.linalg.transpose(NDArray)`

Eigenvalues and right eigenvectors: `np.linalg.eig(square NDArray)`

3 Matplotlib

! Using `import matplotlib as mpl`, `import matplotlib.pyplot as plt`:

3.1 MATLAB interface

Show image (if not in inline mode): `plt.show()`

Plot image: `plt.imshow(NDArray)`

Set axis bounds: `plt.axis([xmin, xmax, ymin, ymax])`

Set x,y axis label: `plt.xlabel(name)`, `plt.ylabel(name)`

Set plot title: `plt.title(name)`

Show plot legend: `plt.legend()`

Visualize matrix vals as heat map: `plt.matshow(NDArray)`

Pan/zoomable plots in PyCharm: insert `mpl.use('Qt5agg')` before `import matplotlib.pyplot as plt`

3.2 Object-oriented interface

4 Pandas

5 Scikit-learn

Cross Validation: `sklearn.model_selection.cross_val_score`