Intro to Data Structures

We’ll start with a quick, non-comprehensive overview of the fundamental data structures in pandas to get you started. The fundamental behavior about data types, indexing, and axis labeling / alignment apply across all of the objects. To get started, import NumPy and load pandas into your namespace:

**In [1]: import** **numpy** **as** **np**  
  
**In [2]: import** **pandas** **as** **pd**

Here is a basic tenet to keep in mind: **data alignment is intrinsic**. The link between labels and data will not be broken unless done so explicitly by you.

We’ll give a brief intro to the data structures, then consider all of the broad categories of functionality and methods in separate sections.

Series

[**Series**](https://pandas.pydata.org/pandas-docs/stable/generated/pandas.Series.html#pandas.Series) is a one-dimensional labeled array capable of holding any data type (integers, strings, floating point numbers, Python objects, etc.). The axis labels are collectively referred to as the **index**. The basic method to create a Series is to call:

**>>>** s = pd.Series(data, index=index)

Here, data can be many different things:

* a Python dict
* an ndarray
* a scalar value (like 5)

The passed **index** is a list of axis labels. Thus, this separates into a few cases depending on what **data is**: [We will only focus on dictionary.]

**From dict**

Series can be instantiated from dicts:

**In [7]:** d = {'b' : 1, 'a' : 0, 'c' : 2}  
  
**In [8]:** pd.Series(d)  
**Out[8]:**   
b    1  
a    0  
c    2  
dtype: int64

**Note**

When the data is a dict, and an index is not passed, the Series index will be ordered by the dict’s insertion order, if you’re using Python version >= 3.6 and Pandas version >= 0.23.

If you’re using Python < 3.6 or Pandas < 0.23, and an index is not passed, the Series index will be the lexically ordered list of dict keys.

In the example above, if you were on a Python version lower than 3.6 or a Pandas version lower than 0.23, the Series would be ordered by the lexical order of the dict keys (i.e. ['a', 'b', 'c'] rather than ['b', 'a', 'c']).

If an index is passed, the values in data corresponding to the labels in the index will be pulled out.

**In [9]:** d = {'a' : 0., 'b' : 1., 'c' : 2.}  
  
**In [10]:** pd.Series(d)  
**Out[10]:**   
a    0.0  
b    1.0  
c    2.0  
dtype: float64  
  
**In [11]:** pd.Series(d, index=['b', 'c', 'd', 'a'])  
Out[11]:   
b    1.0  
c    2.0  
d    NaN  
a    0.0  
dtype: float64

**Note**

NaN (not a number) is the standard missing data marker used in pandas.

Viewing Series Data

Display the index and the underlying NumPy data:

**In [16]:** series.index  
**Out[16]:**   
DatetimeIndex(['2013-01-01', '2013-01-02', '2013-01-03', '2013-01-04',  
               '2013-01-05', '2013-01-06'],  
              dtype='datetime64[ns]', freq='D')  
  
**In [18]:** series.values  
Out[18]:   
[ 0.4691, -0.2829, -1.5091, -1.1356, -1.5091, -1.1356]