Getting Started with pandas

Part 2

Introduction to pandas Data Structures

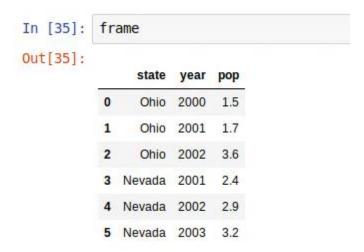
Part 2

DataFrame

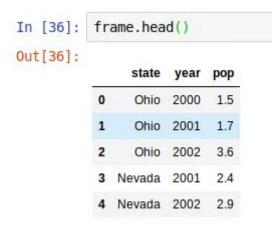
- A DataFrame represents a rectangular table of data and contains an ordered collection of columns, each of which can be a different value type (numeric, string, boolean, etc.).
- The DataFrame has both a row and column index; it can be thought of as a dict of Series all sharing the same index.
- Under the hood, the data is stored as one or more two-dimensional blocks rather than a list, dict, or some other collection of one-dimensional arrays.

 There are many ways to construct a DataFrame, though one of the most common is from a dict of equal-length lists or NumPy arrays:

 The resulting DataFrame will have its index assigned automatically as with Series, and the columns are placed in sorted order. (*Changed in version* 0.23.0: If data is a dict, argument order is maintained for Python 3.6 and later.)



• For large DataFrames, the head method selects only the first five rows:



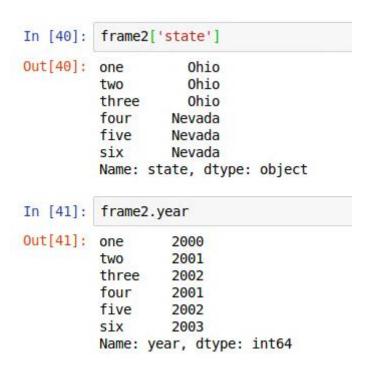
• If you specify a sequence of columns, the DataFrame's columns will be arranged in that order:

n [37]:	pd.DataFrame(data,				columns=['year', 'state', 'pop'])
ut[37]:		year	state	рор	
	0	2000	Ohio	1.5	
	1	2001	Ohio	1.7	
	2	2002	Ohio	3.6	
	3	2001	Nevada	2.4	
	4	2002	Nevada	2.9	
	5	2003	Nevada	3.2	

 If you pass a column that isn't contained in the dict, it will appear with missing values in the result:

```
In [38]: frame2 = pd.DataFrame(data, columns=['year', 'state', 'pop', 'debt'],
                                index=['one', 'two', 'three', 'four',
                                       'five', 'six'])
         frame2
Out[38]:
                       state pop debt
                       Ohio 1.5 NaN
           one 2000
            two 2001
                       Ohio 1.7 NaN
          three 2002
                       Ohio 3.6 NaN
           four 2001 Nevada 2.4 NaN
           five 2002 Nevada 2.9 NaN
            six 2003 Nevada 3.2 NaN
In [39]: frame2.columns
Out[39]: Index(['year', 'state', 'pop', 'debt'], dtype='object')
```

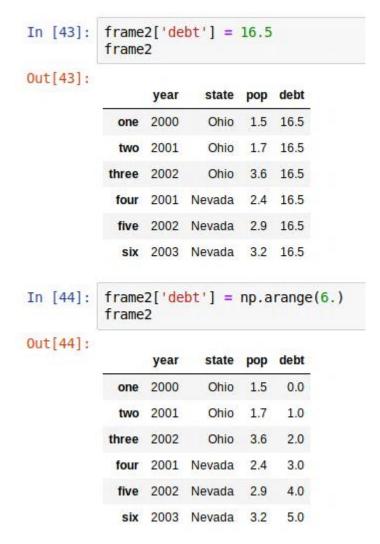
 A column in a DataFrame can be retrieved as a Series either by dict-like notation or by attribute:



 Note that the returned Series have the same index as the DataFrame, and their name attribute has been appropriately set. • Rows can also be retrieved by position or name with the special 10c attribute:

```
In [42]: frame2.loc['three']
Out[42]: year    2002
    state    Ohio
    pop     3.6
    debt    NaN
    Name: three, dtype: object
```

- Columns can be modified by assignment.
- For example, the empty 'debt' column could be assigned a scalar value or an array of values:



- When you are assigning lists or arrays to a column, the value's length must match the length of the DataFrame.
- If you assign a Series, its labels will be realigned exactly to the DataFrame's index, inserting missing values in any holes:



- Assigning a column that doesn't exist will create a new column.
- The del keyword will delete columns as with a dict.

• As an example of del, I first add a new column of boolean values where the state column equals 'Ohio':

In [46]:	<pre>frame2['eastern'] = frame2.state == 'Ohio' frame2</pre>								
Out[46]:		year	state	pop	debt	eastern			
	one	2000	Ohio	1.5	NaN	True	3 2		
	two	2001	Ohio	1.7	-1.2	True			
	three	2002	Ohio	3.6	NaN	True			
	four	2001	Nevada	2.4	-1.5	False			
	five	2002	Nevada	2.9	-1.7	False			
	six	2003	Nevada	3.2	NaN	False			

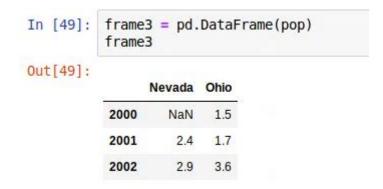
• The del method can then be used to remove this column:

```
In [47]: del frame2['eastern']
    frame2.columns

Out[47]: Index(['year', 'state', 'pop', 'debt'], dtype='object')
```

Another common form of data is a nested dict of dicts:

 If the nested dict is passed to the DataFrame, pandas will interpret the outer dict keys as the columns and the inner keys as the row indices:

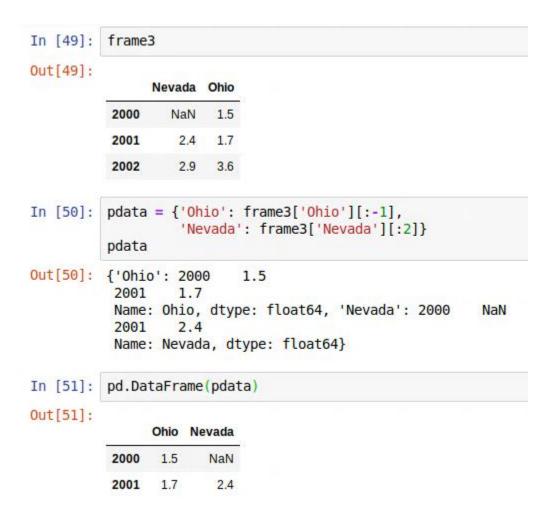


 You can transpose the DataFrame (swap rows and columns) with similar syntax to a NumPy array:



- The keys in the inner dicts are combined and sorted to form the index in the result.
- This isn't true if an explicit index is specified:

• Dicts of Series are treated in much the same way:



• If a DataFrame's index and columns have their name attributes set, these will also be displayed:

In [52]:	<pre>frame3.index.name = 'year'; frame3.columns.name = 'state' frame3</pre>								
Out[52]:	state year	Nevada	Ohio						
	2000	NaN	1.5						
	2001	2.4	1.7						
	2002	2.9	3.6						

• As with Series, the values attribute returns the data contained in the DataFrame as a two-dimensional ndarray:

• If the DataFrame's columns are different dtypes, the dtype of the values array will be chosen to accommodate all of the columns: