



MERGING DATAFRAMES WITH PANDAS

Appending & concatenating Series



append()

- `.append(): Series & DataFrame method`
- Invocation:
 - `s1.append(s2)`
- Stacks rows of `s2` below `s1`
- Method for Series & DataFrames



concat()

- concat(): pandas module *function*
- Invocation:
 - pd.concat([s1, s2, s3])
- Can stack row-wise or column-wise



concat() & .append()

- Equivalence of concat() & .append():
 - `result1 = pd.concat([s1, s2, s3])`
 - `result2 = s1.append(s2).append(s3)`
- `result1 == result2` elementwise



Series of US states

```
In [1]: import pandas as pd
```

```
In [2]: northeast = pd.Series(['CT', 'ME', 'MA', 'NH', 'RI', 'VT',
...: 'NJ', 'NY', 'PA'])
```

```
In [3]: south = pd.Series(['DE', 'FL', 'GA', 'MD', 'NC', 'SC', 'VA',
...: 'DC', 'WV', 'AL', 'KY', 'MS', 'TN', 'AR', 'LA', 'OK', 'TX'])
```

```
In [4]: midwest = pd.Series(['IL', 'IN', 'MN', 'MO', 'NE', 'ND',
...: 'SD', 'IA', 'KS', 'MI', 'OH', 'WI'])
```

```
In [5]: west = pd.Series(['AZ', 'CO', 'ID', 'MT', 'NV', 'NM',
...: 'UT', 'WY', 'AK', 'CA', 'HI', 'OR', 'WA'])
```



Using .append()

```
In [6]: east = northeast.append(south)
```

```
In [7]: print(east)
```

```
0    CT          7    DC
1    ME          8    WV
2    MA          9    AL
3    NH          10   KY
4    RI          11   MS
5    VT          12   TN
6    NJ          13   AR
7    NY          14   LA
8    PA          15   OK
0    DE          16   TX
1    FL
2    GA
3    MD
4    NC
5    SC
6    VA
dtype: object
```



The appended Index

```
In [8]: print(east.index)
Int64Index([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  0,  1,  2,  3,  4,
             5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16], dtype='int64')

In [9]: print(east.loc[3])
3    NH
3    MD
dtype: object
```



Using .reset_index()

```
In [10]: new_east = northeast.append(south).reset_index(drop=True)
```

```
In [11]: print(new_east.head(11))
```

```
0      CT
1      ME
2      MA
3      NH
4      RI
5      VT
6      NJ
7      NY
8      PA
9      DE
10     FL
dtype: object
```

```
In [12]: print(new_east.index)
```

```
RangeIndex(start=0, stop=26, step=1)
```



Using concat()

```
In [13]: east = pd.concat([northeast, south])
```

```
In [14]: print(east.head(11))
```

```
0    CT  
1    ME  
2    MA  
3    NH  
4    RI  
5    VT  
6    NJ  
7    NY  
8    PA  
0    DE  
1    FL  
dtype: object
```

```
In [15]: print(east.index)
```

```
Int64Index([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  0,  1,  2,  3,  4,  
            5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16], dtype='int64')
```



Using ignore_index

```
In [16]: new_east = pd.concat([northeast, south],  
...:                           ignore_index=True)
```

```
In [17]: print(new_east.head(11))  
0      CT  
1      ME  
2      MA  
3      NH  
4      RI  
5      VT  
6      NJ  
7      NY  
8      PA  
9      DE  
10     FL  
dtype: object
```

```
In [18]: print(new_east.index)  
RangeIndex(start=0, stop=26, step=1)
```



MERGING DATAFRAMES WITH PANDAS

Let's practice!



MERGING DATAFRAMES WITH PANDAS

Appending & concatenating DataFrames



Loading population data

```
In [1]: import pandas as pd  
  
In [2]: pop1 = pd.read_csv('population_01.csv', index_col=0)  
  
In [3]: pop2 = pd.read_csv('population_02.csv', index_col=0)  
  
In [4]: print(type(pop1), pop1.shape)  
<class 'pandas.core.frame.DataFrame'> (4, 1)  
  
In [5]: print(type(pop2), pop2.shape)  
<class 'pandas.core.frame.DataFrame'> (4, 1)
```



Examining population data

```
In [6]: print(pop1)
```

```
2010 Census Population
```

Zip Code ZCTA	2010 Census Population
66407	479
72732	4716
50579	2405
46241	30670

```
In [7]: print(pop2)
```

```
2010 Census Population
```

Zip Code ZCTA	2010 Census Population
12776	2180
76092	26669
98360	12221
49464	27481



Appending population DataFrames

```
In [8]: pop1.append(pop2)
Out[8]:
      2010 Census Population
Zip Code ZCTA
66407              479
72732              4716
50579              2405
46241             30670
12776              2180
76092             26669
98360              12221
49464             27481
```

```
In [9]: print(pop1.index.name, pop1.columns)
Zip Code ZCTA Index(['2010 Census Population'], dtype='object')
```

```
In [10]: print(pop2.index.name, pop2.columns)
Zip Code ZCTA Index(['2010 Census Population'], dtype='object')
```



Population & unemployment data

```
In [11]: population = pd.read_csv('population_00.csv',
...:                               index_col=0)
```

```
In [12]: unemployment = pd.read_csv('unemployment_00.csv',
...:                               index_col=0)
```

```
In [13]: print(population)
           2010 Census Population
      Zip Code ZCTA
57538
59916
37660
2860
```

Zip Code ZCTA	2010 Census Population
57538	322
59916	130
37660	40038
2860	45199

```
In [14]: print(unemployment)
           unemployment participants
      Zip
2860        0.11       34447
46167       0.02       4800
1097        0.33        42
80808       0.07       4310
```

Zip	unemployment	participants
2860	0.11	34447
46167	0.02	4800
1097	0.33	42
80808	0.07	4310



Appending population & unemployment

```
In [15]: population.append(unemployment)
```

```
Out[15]:
```

	2010 Census Population	participants	unemployment
57538	322.0	NaN	NaN
59916	130.0	NaN	NaN
37660	40038.0	NaN	NaN
2860	45199.0	NaN	NaN
2860	NaN	34447.0	0.11
46167	NaN	4800.0	0.02
1097	NaN	42.0	0.33
80808	NaN	4310.0	0.07



Repeated index labels

```
In [15]: population.append(unemployment)
```

```
Out[15]:
```

```
   2010 Census Population participants unemployment
57538              322.0        NaN        NaN
59916              130.0        NaN        NaN
37660            40038.0        NaN        NaN
2860             45199.0        NaN        NaN
2860                NaN    34447.0       0.11
46167                NaN     4800.0       0.02
1097                 NaN      42.0       0.33
80808                NaN     4310.0       0.07
```



Concatenating rows

```
In [16]: pd.concat([population, unemployment], axis=0)
```

```
Out[16]:
```

	2010 Census	Population	participants	unemployment
57538		322.0	NaN	NaN
59916		130.0	NaN	NaN
37660		40038.0	NaN	NaN
2860		45199.0	NaN	NaN
2860		NaN	34447.0	0.11
46167		NaN	4800.0	0.02
1097		NaN	42.0	0.33
80808		NaN	4310.0	0.07



Concatenating columns

```
In [17]: pd.concat([population, unemployment], axis=1)
```

```
Out[17]:
```

```
      2010 Census Population  unemployment  participants
1097          NaN        0.33         42.0
2860        45199.0       0.11       34447.0
37660       40038.0       NaN        NaN
46167          NaN        0.02       4800.0
57538        322.0       NaN        NaN
59916        130.0       NaN        NaN
80808          NaN        0.07       4310.0
```



MERGING DATAFRAMES WITH PANDAS

Let's practice!



MERGING DATAFRAMES WITH PANDAS

Concatenation, keys, & MultiIndexes



Loading rainfall data

```
In [1]: import pandas as pd  
  
In [2]: file1 = 'q1_rainfall_2013.csv'  
  
In [3]: rain2013 = pd.read_csv(file1, index_col='Month', parse_dates=True)  
  
In [4]: file2 = 'q1_rainfall_2014.csv'  
  
In [5]: rain2014 = pd.read_csv(file2, index_col='Month', parse_dates=True)
```



Examining rainfall data

```
In [6]: print(rain2013)
```

```
Precipitation
```

```
Month
```

```
Jan      0.096129
Feb      0.067143
Mar      0.061613
```

```
In [7]: print(rain2014)
```

```
Precipitation
```

```
Month
```

```
Jan      0.050323
Feb      0.082143
Mar      0.070968
```



Concatenating rows

```
In [8]: pd.concat([rain2013, rain2014], axis=0)
```

```
Out[8]:
```

```
Precipitation
Jan      0.096129
Feb      0.067143
Mar      0.061613
Jan      0.050323
Feb      0.082143
Mar      0.070968
```



Using multi-index on rows

```
In [7]: rain1314 = pd.concat([rain2013, rain2014], keys=[2013, 2014], axis=0)
```

```
In [8]: print(rain1314)
```

```
Precipitation
2013 Jan      0.096129
          Feb      0.067143
          Mar      0.061613
2014 Jan      0.050323
          Feb      0.082143
          Mar      0.070968
```



Accessing a multi-index

```
In [9]: print(rain1314.loc[2014])
      Precipitation
Jan      0.050323
Feb      0.082143
Mar      0.070968
```



Concatenating columns

```
In [10]: rain1314 = pd.concat([rain2013, rain2014], axis='columns')
```

```
In [11]: print(rain1314)
```

	Precipitation	Precipitation
Jan	0.096129	0.050323
Feb	0.067143	0.082143
Mar	0.061613	0.070968



Using a multi-index on columns

```
In [12]: rain1314 = pd.concat([rain2013, rain2014], keys=[2013, 2014], axis='columns')
```

```
In [13]: print(rain1314)
```

	2013	2014
Precipitation	Precipitation	
Jan	0.096129	0.050323
Feb	0.067143	0.082143
Mar	0.061613	0.070968

```
In [14]: rain1314[2013]
```

```
Out[14]:
```

	Precipitation
Jan	0.096129
Feb	0.067143
Mar	0.061613



pd.concat() with dict

```
In [15]: rain_dict = {2013: rain2013, 2014: rain2014}
```

```
In [16]: rain1314 = pd.concat(rain_dict, axis='columns')
```

```
In [17]: print(rain1314)
```

	2013	2014
Precipitation	Precipitation	
Jan	0.096129	0.050323
Feb	0.067143	0.082143
Mar	0.061613	0.070968



MERGING DATAFRAMES WITH PANDAS

Let's practice!



MERGING DATAFRAMES WITH PANDAS

Outer & inner joins



Using with arrays

```
In [1]: import numpy as np
```

```
In [2]: import pandas as pd
```

```
In [3]: A = np.arange(8).reshape(2,4) + 0.1
```

```
In [4]: print(A)
[[ 0.1  1.1  2.1  3.1]
 [ 4.1  5.1  6.1  7.1]]
```

```
In [5]: B = np.arange(6).reshape(2,3) + 0.2
```

```
In [6]: print(B)
[[ 0.2  1.2  2.2]
 [ 3.2  4.2  5.2]]
```

```
In [7]: C = np.arange(12).reshape(3,4) + 0.3
```

```
In [8]: print(C)
[[ 0.3  1.3  2.3  3.3]
 [ 4.3  5.3  6.3  7.3]
 [ 8.3  9.3 10.3 11.3]]
```



Stacking arrays horizontally

```
In [6]: np.hstack([B, A])
Out[6]:
array([[ 0.2,  1.2,  2.2,  0.1,  1.1,  2.1,  3.1],
       [ 3.2,  4.2,  5.2,  4.1,  5.1,  6.1,  7.1]])
```

```
In [7]: np.concatenate([B, A], axis=1)
Out[7]:
array([[ 0.2,  1.2,  2.2,  0.1,  1.1,  2.1,  3.1],
       [ 3.2,  4.2,  5.2,  4.1,  5.1,  6.1,  7.1]])
```



Stacking arrays vertically

```
In [8]: np.vstack([A, C])
Out[8]:
array([[ 0.1,    1.1,    2.1,    3.1],
       [ 4.1,    5.1,    6.1,    7.1],
       [ 0.3,    1.3,    2.3,    3.3],
       [ 4.3,    5.3,    6.3,    7.3],
       [ 8.3,    9.3,   10.3,   11.3]])
```

```
In [9]: np.concatenate([A, C], axis=0)
Out[9]:
array([[ 0.1,    1.1,    2.1,    3.1],
       [ 4.1,    5.1,    6.1,    7.1],
       [ 0.3,    1.3,    2.3,    3.3],
       [ 4.3,    5.3,    6.3,    7.3],
       [ 8.3,    9.3,   10.3,   11.3]])
```



Incompatible array dimensions

```
In [11]: np.concatenate([A, B], axis=0) # incompatible columns
```

```
-----  
ValueError                                Traceback (most recent call last)  
----> 1 np.concatenate([A, B], axis=0) # incompatible columns
```

ValueError: all the input array dimensions except for the concatenation axis must match exactly

```
In [12]: np.concatenate([A, C], axis=1) # incompatible rows
```

```
-----  
ValueError                                Traceback (most recent call last)  
----> 1 np.concatenate([A, C], axis=1) # incompatible rows
```

ValueError: all the input array dimensions except for the concatenation axis must match exactly



Population & unemployment data

```
In [13]: population = pd.read_csv('population_00.csv',
...:                               index_col=0)
```

```
In [14]: unemployment = pd.read_csv('unemployment_00.csv',
...:                                   ...index_col=0)
```

```
In [15]: print(population)
2010 Census Population
Zip Code ZCTA
57538                  322
59916                  130
37660                 40038
2860                  45199
```

```
In [16]: print(unemployment)
unemployment participants
Zip
2860      0.11      34447
46167     0.02      4800
1097      0.33       42
80808    0.07      4310
```



Converting to arrays

```
In [17]: population_array = np.array(population)
```

```
In [18]: print(population_array) # Index info is lost
[[ 322]
 [ 130]
[40038]
[45199]]
```

```
In [19]: unemployment_array = np.array(unemployment)
```

```
In [20]: print(population_array)
[[ 1.10000000e-01    3.44470000e+04]
 [ 2.00000000e-02    4.80000000e+03]
 [ 3.30000000e-01    4.20000000e+01]
 [ 7.00000000e-02    4.31000000e+03]]
```



Manipulating data as arrays

```
In [21]: print(np.concatenate([population_array, unemployment_array],  
...:                         axis=1))  
[[ 3.22000000e+02   1.10000000e-01   3.44470000e+04]  
[ 1.30000000e+02   2.00000000e-02   4.80000000e+03]  
[ 4.00380000e+04   3.30000000e-01   4.20000000e+01]  
[ 4.51990000e+04   7.00000000e-02   4.31000000e+03]]
```



Joins

- Joining tables: Combining rows of multiple tables
- Outer join
 - Union of index sets (all labels, no repetition)
 - Missing fields filled with NaN
- Inner join
 - Intersection of index sets (only common labels)



Concatenation & inner join

```
In [22]: pd.concat([population, unemployment], axis=1, join='inner')
```

```
Out[22]:
```

	2010 Census Population	unemployment	participants
2860	45199	0.11	34447



Concatenation & outer join

```
In [23]: pd.concat([population, unemployment], axis=1, join='outer')
```

```
Out[23]:
```

	2010 Census Population	unemployment	participants
1097	NaN	0.33	42.0
2860	45199.0	0.11	34447.0
37660	40038.0	NaN	NaN
46167	NaN	0.02	4800.0
57538	322.0	NaN	NaN
59916	130.0	NaN	NaN
80808	NaN	0.07	4310.0



Inner join on other axis

```
In [24]: pd.concat([population, unemployment], join='inner', axis=0)
Out[24]:
Empty DataFrame
Columns: []
Index: [2860, 46167, 1097, 80808, 57538, 59916, 37660, 2860]
```



MERGING DATAFRAMES WITH PANDAS

Let's practice!