

Combining the Data

Concatenation and Merging

Concatenation

- We require to **concatenate** the data horizontally or vertically
- This feature is similar to R feature of `cbind()` and `rbind()`
- We have two alternatives for this task: `.append()` and `concat()`

Syntax:

```
data1.append(data2).reset_index(drop=True)
```

```
pandas.concat([col1,col2,...], axis=0, ignore_index=False)
```

Column-wise concatenation

	Name	Age	City
0	Raghav	32	Mumbai
1	Ganesh	45	Mumbai
2	Pallavi	31	Mumbai

+

	Name	Age	City
0	Harpreet	45	Delhi
1	Nitin	21	Delhi
2	Jeevan	24	Delhi
3	Garima	34	Delhi



	Name	Age	City
0	Raghav	32	Mumbai
1	Ganesh	45	Mumbai
2	Pallavi	31	Mumbai
3	Harpreet	45	Delhi
4	Nitin	21	Delhi
5	Jeevan	24	Delhi
6	Garima	34	Delhi

Using .append()

- Consider the data frames `mum` and `delhi` as shown

mum

	Name	Age	City
0	Raghav	32	Mumbai
1	Ganesh	45	Mumbai
2	Pallavi	31	Mumbai

delhi

	Name	Age	City
0	Harpreet	45	Delhi
1	Nitin	21	Delhi
2	Jeevan	24	Delhi
3	Garima	34	Delhi

- If we consider applying the `append()` for the two, we get the following result

```
In [3]: comb1 = mum.append(delhi)
```

```
...: comb1
```

```
Out[3]:
```

	Name	Age	City
0	Raghav	32	Mumbai
1	Ganesh	45	Mumbai
2	Pallavi	31	Mumbai
0	Harpreet	45	Delhi
1	Nitin	21	Delhi
2	Jeevan	24	Delhi
3	Garima	34	Delhi

Notice that the indices of rows have not been given as desired

Resetting the index

- We can reset the index of the data frame by calling `.reset_index()` on the data frame object

```
In [5]: comb1 = mum.append(delhi).reset_index()  
...: comb1
```

```
Out[5]:  
   index  Name  Age  City  
0       0  Raghav  32  Mumbai  
1       1  Ganesh  45  Mumbai  
2       2  Pallavi  31  Mumbai  
3       0 Harpreet  45  Delhi  
4       1   Nitin  21  Delhi  
5       2  Jeevan  24  Delhi  
6       3  Garima  34  Delhi
```

Notice still that the indices of rows have been given as desired but a column of index has been written. It can be avoided by specifying `drop=True` option in `.reset_index()`

```
In [32]: comb1 = mum.append(delhi).reset_index(drop=True)  
...: comb1
```

```
Out[32]:  
   Name  Age  City  
0  Raghav  32  Mumbai  
1  Ganesh  45  Mumbai  
2  Pallavi  31  Mumbai  
3 Harpreet  45  Delhi  
4   Nitin  21  Delhi  
5  Jeevan  24  Delhi  
6  Garima  34  Delhi
```

Using pandas . concat ()

- Similarly, we can get result with pandas . concat () as

```
In [6]: comb2 = pd.concat([mum, delhi])
...: comb2
Out[6]:
```

	Name	Age	City
0	Raghav	32	Mumbai
1	Ganesh	45	Mumbai
2	Pallavi	31	Mumbai
0	Harpreet	45	Delhi
1	Nitin	21	Delhi
2	Jeevan	24	Delhi
3	Garima	34	Delhi

- And, we can get rid of index with the option ignore_index=True

```
In [7]: comb2 = pd.concat([mum, delhi],ignore_index=True)
...: comb2
Out[7]:
```

	Name	Age	City
0	Raghav	32	Mumbai
1	Ganesh	45	Mumbai
2	Pallavi	31	Mumbai
3	Harpreet	45	Delhi
4	Nitin	21	Delhi
5	Jeevan	24	Delhi
6	Garima	34	Delhi

Row-wise concatenation

In [29]: aster
Out[29]:

	Sales
Month	
January	17.5
February	21.8
March	24.5
April	25.1

+

In [30]: rose
Out[30]:

	Sales
Month	
January	7.8
February	3.5
March	5.9
April	6.8



In [31]: flower_sales
Out[31]:

	aster	rose
	Sales	Sales
Month		
January	17.5	7.8
February	21.8	3.5
March	24.5	5.9
April	25.1	6.8

Using pandas . concat ()

```
In [5]: flower_sales = pd.concat([aster,rose],axis='columns')
```

```
In [6]: flower_sales
```

```
Out[6]:
```

	Sales	Sales
Month		
January	17.5	7.8
February	21.8	3.5
March	24.5	5.9
April	25.1	6.8

```
In [7]: flower_sales = pd.concat([aster,rose],keys = ['aster','rose'],axis='columns')
```

```
In [8]: flower_sales
```

```
Out[8]:
```

	aster	rose
	Sales	Sales
Month		
January	17.5	7.8
February	21.8	3.5
March	24.5	5.9
April	25.1	6.8

Merging the Data

- We can join(SQL type join) the data sets with the help of function `pandas.DataFrame.merge()`

Syntax :

```
pandas.DataFrame.merge(right, how='inner', on=None, left_on=None, right_on=None,...)
```

Where

right: right Data frame, how: type of join, on:Column or index level names to join on, left_on: Column or index level names to join on in the left DataFrame, right_on: Column or index level names to join on in the right DataFrame,

Inner Join

- By default, the function does inner join

```
In [16]: demog
```

```
Out[16]:
```

	Name	Age
0	Nandan	33
1	Girish	31
2	Sonali	26
3	Sheetal	29

```
In [17]: rating
```

```
Out[17]:
```

	Name	Rating
0	Nandan	5
1	Kapil	6
2	Girish	8
3	Sonali	7



```
In [18]: dem_rat = demog.merge(rating,on="Name")
```

```
In [19]: dem_rat
```

```
Out[19]:
```

	Name	Age	Rating
0	Nandan	33	5
1	Girish	31	8
2	Sonali	26	7

Outer Join

```
In [16]: demog
```

```
Out[16]:
```

	Name	Age
0	Nandan	33
1	Girish	31
2	Sonali	26
3	Sheetal	29

```
In [17]: rating
```

```
Out[17]:
```

	Name	Rating
0	Nandan	5
1	Kapil	6
2	Girish	8
3	Sonali	7



```
In [21]: dem_rat = demog.merge(rating,how='left')
```

```
In [22]: dem_rat
```

```
Out[22]:
```

	Name	Age	Rating
0	Nandan	33	5.0
1	Girish	31	8.0
2	Sonali	26	7.0
3	Sheetal	29	NaN

Right Join

```
In [16]: demog
```

```
Out[16]:
```

	Name	Age
0	Nandan	33
1	Girish	31
2	Sonali	26
3	Sheetal	29

```
In [17]: rating
```

```
Out[17]:
```

	Name	Rating
0	Nandan	5
1	Kapil	6
2	Girish	8
3	Sonali	7



```
In [23]: dem_rat = demog.merge(rating,how='right')
```

```
In [24]: dem_rat
```

```
Out[24]:
```

	Name	Age	Rating
0	Nandan	33.0	5
1	Girish	31.0	8
2	Sonali	26.0	7
3	Kapil	NaN	6

Outer Join

```
In [16]: demog
```

```
Out[16]:
```

	Name	Age
0	Nandan	33
1	Girish	31
2	Sonali	26
3	Sheetal	29

```
In [17]: rating
```

```
Out[17]:
```

	Name	Rating
0	Nandan	5
1	Kapil	6
2	Girish	8
3	Sonali	7



```
In [27]: dem_rat = demog.merge(rating,how='outer')
```

```
In [28]: dem_rat
```

```
Out[28]:
```

	Name	Age	Rating
0	Nandan	33.0	5.0
1	Girish	31.0	8.0
2	Sonali	26.0	7.0
3	Sheetal	29.0	NaN
4	Kapil	NaN	6.0

Questions?