Lesson 11 Advanced Pandas Data Frames 4 Last update: Apr 18, 2021

Concatenating Data Frames

Data frames can be joined together by columns and rows. Usually the rows are equal or the columns are equal. If there is no match between rows and columns then NaN's are produced. We use the pandas **concat** function to concatenate two data frames together

Example Equal Rows and Columns

Make data frame 1:

```
import pandas as pd

df1 = pd.DataFrame({'A': [1, 2, 3, 4],
                    'B': [5, 6, 7, 8],
                    'C': [9, 10, 11, 12],
                    'D': [13, 14, 15, 16]})
print(df1)
```

```
   A  B   C   D
0  1  5   9  13
1  2  6  10  14
2  3  7  11  15
3  4  8  12  16
```

Make data frame 2

```
df2 = pd.DataFrame({'A': [17, 18, 19, 20],
                    'B': [21, 22, 23, 24],
                    'C': [25, 26, 27, 28],
                    'D': [29, 30, 31, 32]})
print(df2)
```

```
    A   B   C   D
0  17  21  25  29
1  18  22  26  30
2  19  23  27  31
3  20  24  28  32
```

To concatenate two data frames together we first put our data frames in a list:

```
data_frames = [df1, df2]
```

We then call the pandas **concat** methods to concatenate the data frames together. We set **ignore index** to true, so that we get unique consecutive indexes, or else we would have duplicate indexes.

```
df3 = pd.concat(data_frames,ignore_index=True)
print(df3)
```

```
    A   B   C   D
0   1   5   9  13
1   2   6  10  14
2   3   7  11  15
3   4   8  12  16
4  17  21  25  29
5  18  22  26  30
6  19  23  27  31
7  20  24  28  32
```

Advanced Pandas 4 Homework Question 1

Make data frames with different rows and columns, and concatenate them together. Try data frames with indexes and no indexes.

You can add index to a data frame when you make by using an index key like this:

```
df1 = pd.DataFrame({'A': [1, 2, 3, 4],
                    'B': [5, 6, 7, 8],
                    'C': [9, 10, 11, 12],
                    'D': [13, 14, 15, 16]},
                    index=[0, 1, 2, 3])
```

Merging data frames on a common key

We can merge two data frames by a common key, or index column. The resulting data frame will have columns from one data frame and the columns from the other data frame but has a common key or index column.

Here is a left data frame with common keys [1,2,3,4]

```
left = pd.DataFrame({'key': [1,2,3,4],
          'A': [1,2,3,4],
          'B': [5,6,7,8]})
print(left)
```

```
   key  A  B
0    1  1  5
1    2  2  6
2    3  3  7
3    4  4  8
```

Here is the right data frame with common keys [1,2,3,4]

```
right = pd.DataFrame({'key': [1,2,3,4],
        'C': [9,10,11,12],
        'D': [13,14,15,16]})
print(right)
```

```
   key   C   D
0    1   9  13
1    2  10  14
2    3  11  15
3    4  12  16
```

```
result = pd.merge(left, right, on='key')
print(result)
```

```
   key  A  B   C   D
0    1  1  5   9  13
1    2  2  6  10  14
2    3  3  7  11  15
3    4  4  8  12  16
```

Advanced Pandas 4 Homework Question 2

Try merging the data frames using different keys, or semi different keys.
Try merging the data frames having common keys but different indexes.

Merge methods

The **'how'** parameter : specifies the merge method one of: 'left', 'right', 'outer', 'inner' and 'cross' the default **'how'** is 'inner'.

Merge method	**Description**
left	Use keys from left frame only
right	Use keys from right frame only
outer	Use union of keys from both frames
inner	Use intersection of keys from both frames (default)
cross	creates the cartesian product from both frames, preserves the order of the left keys

To get the full effect of joining with 'how' we need dataframes with some different keys.

```
left = pd.DataFrame({'key': [1,2,3,4],
                     'A': [1,2,3,4],
                     'B': [5,6,7,8]})
print(left)
```

```
   key  A  B
0    1  1  5
1    2  2  6
2    3  3  7
3    4  4  8
```

Data frame 2 Using different keys:

```
right = pd.DataFrame({'key': [1,2,5,6],
                      'C': [9,10,11,12],
                      'D': [13,14,15,16]})
print(right)
```

```
   key   C   D
0    1   9  13
1    2  10  14
2    5  11  15
3    6  12  16
```

```
result = pd.merge(left, right, on='key', how='outer')
print(result)
```

```
   key    A    B     C     D
0    1  1.0  5.0   9.0  13.0
1    2  2.0  6.0  10.0  14.0
2    3  3.0  7.0   NaN   NaN
3    4  4.0  8.0   NaN   NaN
4    5  NaN  NaN  11.0  15.0
5    6  NaN  NaN  12.0  16.0
```

Rows 2 and 3 are NaN for C and D because **right** dataframe does not have keys 3 and 4.

Rows 4 and 5 are NaN for A and B because **left** dataframe does not have keys 3 and 4

Advanced Pandas 4 Homework question 3

Experiment with the 'how' merge method parameter's outer, inner, left, right and cross. Make sure the data frames have some different keys.

Example using a how:

```
result = pd.merge(left, right, on='key', how='outer')
print(result)
```

MERGE METHOD CASES

Merge case joins	Description
one-to-one **1:1**	joining two Data Frame objects on their indexes that contain unique values
one-to-many **1:m**	joining an index (unique) to one or more columns in a different Data Frame.
many-to-one **m:1**	Joining a one or more columns to an index (unique) in a different Data Frame.
many-to-many **m:m**	joining columns on columns.

Many to one example

We have two data frames each with the same key column. Each data frame has different columns names. What we want to do is make a third data frame that merges the two data frames to the same key values**.** Many data rows with the same column key to other rows the same column key (many to one)

Make data frame 1:

```
df1 = pd.DataFrame({'key': [1, 1, 3, 4],
                    'A': [5, 6, 7, 8],
                    'B': [9, 10, 11, 12]} )
print(df1)
```

```
   Key  A   B
0    1  5   9
1    1  6  10
2    3  7  11
3    4  8  12
```

Make data frame 2

```
df2 = pd.DataFrame({'key': [1, 2, 3, 4],
                    'C': [25, 26, 27, 28],
                    'D': [29, 30, 31, 32]})
print(df2)
```

```
   Key   C   D
0    1  25  29
1    2  26  30
2    3  27  31
3    4  28  32
```

```
df3 = pd.merge(df1, df2, left_on="key", right_on="key", how="left", validate="m:1")
print(df3)
```

```
   Key  A   B   C   D
0    1  5   9  25  29
1    1  6  10  25  29
2    3  7  11  27  31
3    4  8  12  28  32
```

Note: The result dataframe only has keys 1,1,3 and 4. key 2 is not present because key 2 is not in the first data frame. For many to 1 relationships the keys of the first data frame must be present in second data frame to be able to merge.

Advanced Pandas 4 Homework Question 4

Try some of the other merge method cases like m:m.

JOINING ON A INDEX

We use the pandas **join** method to join two data frames using the index.

`DataFrame.join`(*other, on=None, how='left', lsuffix='', rsuffix='', sort=False*)

Parameter	Description
other	DataFrame, Series, or list of DataFrame
on	optional Column or index level name(s) in the caller to join on the index in *other*, otherwise joins index-on-index.
how	'left', 'right', 'outer', 'inner' default 'left'
lsuffix	Suffix to use from left frame's overlapping columns. Default is ''
rsuffix	Suffix to use from right frame's overlapping columns. Default is ''
sort	Order result lexicographically by the join key. If False, the order of the join key depends on the join type (how keyword).

The pandas **join** method returns a dataframe containing columns from both the caller and other parameter. The default is **left join**

We make a data frame with columns 'A',' B' and indexes 1, 2, 3

```
left = pd.DataFrame({'A': [1, 2, 3],
                     'B': [4, 5, 6]},
                    index=[1, 2, 3])
print(left)
```

```
   A  B
1  1  4
2  2  5
3  3  6
```

We then make another data frame with columns ‘C’,’ D’ and indexes 1, 2,3

```
right = pd.DataFrame({'C': [7, 8, 9],
                      'D': [10, 11, 12]},
                     index=[1, 2, 3])

print(right)
```

```
   C   D
1  7  10
2  8  11
3  9  12
```

Join the data frames together using the indexes:

```
result = left.join(right)
print(result)
```

```
   A  B  C   D
1  1  4  7  10
2  2  5  8  11
3  3  6  9  12
```

using lsuffix and rsuffix

lsuffix and rsuffix specifies the suffixes of the added columns in situations where the column values are different, and additional columns are made, example:

```
left = pd.DataFrame({'A': [1, 2, 3],
                     'B': [4, 5, 6]},
                     index=[1, 2, 3])

right = pd.DataFrame({'A': [7, 8, 9],
       'C': [10, 11, 12]},
       index=[1, 2, 3])
```

```
result = left.join(right,lsuffix='left', rsuffix='right')
print(result)
```

```
  Aleft  B  Aright  C
1    1  4     7  10
2    2  5     8  11
3    3  6     9  12
```

Advanced Pandas4 Homework Question 5

Try merging data frames with different 'how' join cases: right, inner, outer and see what happens. Try joining on a key using 'on'. Use lsuffix and rsuffix.

Example: **result = left.join(right,how="inner")**

LOOKUP VALUES FROM DIFFERENT DATA FRAMES AND PUT INTO ANOTHER DATA FRAME

Let's say you have a data frame with a list of person names and what country they belong where each county has a country id code.

```
# country  ids, persons and countries
df1 = pd.DataFrame({"id":[1,2,3,4,5,1,2,3,4,4],
            "person":["tom","mary","bill","joe","todd","sue","george","jane","dave","bobo"],
             "country":["Canada","USA","UK","Mexico","Australia","Canada",
                 "USA","UK","Mexico","Mexico"]})
print(df1)
```

```
    id  person    country
0    1     tom     Canada
1    2    mary        USA
2    3    bill         UK
3    4     joe     Mexico
4    5    todd  Australia
5    1     sue     Canada
6    2  george        USA
7    3    jane         UK
8    4    dave     Mexico
9    4    bobo     Mexico
```

Let say you have a another data frame with a country id and a city name. The country id indicates which country the city belongs to.

```
# country id's and cities
df2 = pd.DataFrame({"id":[1,2,3,4,5],
            "city":["Ottawa","Washington","London","Mexico","Sydney"]})
  print(df2)
```

```
   id        city
0   1      Ottawa
1   2  Washington
2   3      London
3   4      Mexico
4   5      Sydney
```

The id tells you what country the city is in.

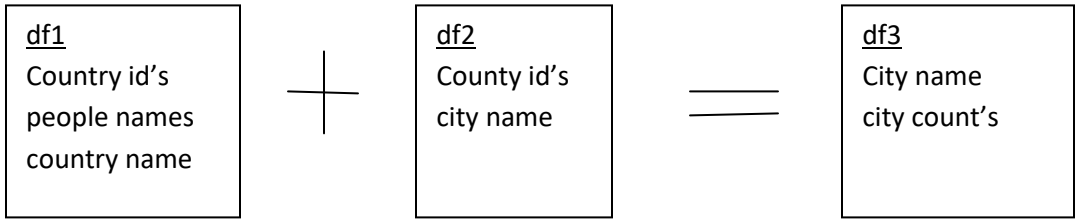
Let's say you have another data frame with a list of cities and a added column with empty values to count the cities belonging to certain people.

```
# city counts
df3 = pd.DataFrame({"city":["Ottawa","Washington","London","Mexico","Sydney"]})

# add count column to result dataframe
df3['count'] = 0
print(df3)
```

```
         city  count
0      Ottawa      0
1  Washington      0
2      London      0
3      Mexico      0
4      Sydney      0
```

Our goal is to count all the people in each city using the cities and country codes from data frame 1 (df1) and using the cities and country codes from data frame 2 (df2) and then make a third data frame df3 that has the counts of the people in each city.

Goal: count the number of people in each city using df1 and df2



Here are the steps:

(1) read each row of df1 and extract the county id;
(2) look up the county id in df2 and extract the city name
(3) look up the city in df3 and increment the count

Here is the rest of the code:

```
import matplotlib.pyplot as plt

# for each row in data frame 1
for i in range(len(df1)):

    # get country id code at index i
    id = df1.at[i, 'id' ]

    # get row in city data frame for id code
    cityrow = df2.loc[df2['id'] == id]
```

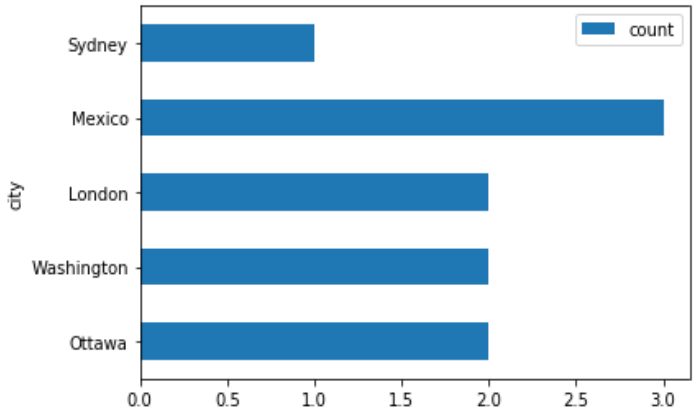
```
# check if got a row
if len(cityrow) == 1:
    # get city for from city row
    city = cityrow.iloc[0]['city']

    # get row index in cities count data frame
    cityindex = df3.loc[df3['city'] == city].index[0]

    # increment count for this country
    df3.at[cityindex,'count'] += 1

# print cities count results
print(df3)

# plot cities and count
df3.plot.barh(x='city',y='count')
plt.show()
```

Here are the city counts:

```
          city  count
0       Ottawa      2
1   Washington      2
2       London      2
3       Mexico      3
4       Sydney      1
```

We now have a count of how many people there are in each city using the country id's.

You should get a horizontal bar chart like this:



Advanced Pandas4 Homework Question 6

Take the previous countries and cities program and use **merge** or **join** to merge df1 and df2 into df3. Sort the cities by name and count.

citycounts = df3['city'].value_counts().sort_index()

citycounts = df3['city'].value_counts().sort_values()

You may get something like this:

```
   id  person    country        city
0   1     tom     Canada      Ottawa
1   2    mary        USA  Washington
2   3    bill         UK      London
3   4     joe     Mexico      Mexico
4   5    todd  Australia      Sydney
5   1     sue     Canada      Ottawa
6   2  george        USA  Washington
7   3    jane         UK      London
8   4    dave     Mexico      Mexico
9   4    bobo     Mexico      Mexico
```

```
sort by city name
London       2
Mexico       3
Ottawa       2
Sydney       1
Washington   2
Name: city, dtype: int64

sort by city count
Sydney       1
Ottawa       2
London       2
Washington   2
Mexico       3
Name: city, dtype: int64
```

Call your homework file advpandas4_homework.py

END