# レシピ63 都市間の航空便の総数（P179）

- flightsデータセットには発着空港のデータがある
- 例えば、ヒューストンからアトランタへの便数を数えるのは簡単だが
- 都市間の全便数を数えるのは難しい

- このレシピでは、発着がどちらかに関係なく、都市間の全便数を数える
- そのために、発着空港を英字順にソートして、2空港の組み合わせが常に同じになるようにする
- それから、この新たなカラム配置を使ってグループ分けを行い数える

In [181…
```
print('flights.csv')
flights.head()
```

Out[181…
flights.csv

| | MONTH | DAY | WEEKDAY | AIRLINE | ORG_AIR | DEST_AIR | SCHED_DEP | DEP_DELAY | AIR_TIME | DIST | SCHED_ARR | ARR_DELAY | DIVERTED | CANCELLED |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 1 | 1 | 4 | WN | LAX | SLC | 1625 | 58.0 | 94.0 | 590 | 1905 | 65.0 | 0 | 0 |
| **1** | 1 | 1 | 4 | UA | DEN | IAD | 823 | 7.0 | 154.0 | 1452 | 1333 | -13.0 | 0 | 0 |
| **2** | 1 | 1 | 4 | MQ | DFW | VPS | 1305 | 36.0 | 85.0 | 641 | 1453 | 35.0 | 0 | 0 |
| **3** | 1 | 1 | 4 | AA | DFW | DCA | 1555 | 7.0 | 126.0 | 1192 | 1935 | -7.0 | 0 | 0 |
| **4** | 1 | 1 | 4 | WN | LAX | MCI | 1720 | 48.0 | 166.0 | 1363 | 2225 | 39.0 | 0 | 0 |

---

（1）発着空港ごとに全便数を数える

In [182…
```
flights_ct = flights.groupby(['ORG_AIR', 'DEST_AIR']).size()
flights_ct
```

Out[182…
```
ORG_AIR  DEST_AIR
ATL      ABE          31
         ABQ          16
         ABY          19
         ACY           6
         AEX          40
                      ..
SFO      SNA         122
         STL          20
         SUN          10
         TUS          20
         XNA           2
Length: 1130, dtype: int64
```

---

（2）ヒューストン（IAH）とアトランタ（ATL）間の全便数を両方向で選ぶ

- MultiIndexにおける列や値の取得は、`.loc[]`を利用する
- 参考HP：https://note.nkmk.me/python-pandas-multiindex-indexing/

In [183…
```
# 例） MultiIndexの'ORG_AIR'='ATL'を取得したい場合
display(flights_ct.loc['ATL'])
```

```
DEST_AIR
ABE    31
ABQ    16
ABY    19
ACY     6
AEX    40
       ..
TUS    10
TYS    60
VLD    23
VPS    65
XNA    54
Length: 159, dtype: int64
```

In [184…
```
# 例） MultiIndexで、ORG：'ATL'> DEST：'ABE'の値を取得したい場合
flights_ct.loc['ATL', 'ABE']
```

Out[184…
```
31
```

In [185…
```
# 結果、IAHとATLの全便数を取得する場合はこうなる、正しく選択するためにここでは「タプル」のリストを使う
flights_ct.loc[[('ATL', 'IAH'), ('IAH', 'ATL')]]
```

Out[185…
```
ORG_AIR  DEST_AIR
ATL      IAH         121
IAH      ATL         148
dtype: int64
```

---

（3）

- この2つの数値を足して,同様に全便数を求めることができるが、もっとよいやり方がある
- ATLとIAHであればユニークに組み合わせは1つだが、2つできてしまっている、これを1つだけにしたい
- 発着の2空港を英字順にソートすれば、ラベルを1つに出来る
- そのためには、DataFrameの`apply`を使う。ただし、groupbyのapplyではない。ここではグループ分けはしない
- applyにsorted関数を渡す、計算方向はaxis=1(idx側)
- sortedに渡される行はSeriesになる

In [186…
```
# cf
flights[['ORG_AIR', 'DEST_AIR']]
```

Out[186…

| | ORG_AIR | DEST_AIR |
|---|---|---|
| **0** | LAX | SLC |
| **1** | DEN | IAD |
| **2** | DFW | VPS |
| **3** | DFW | DCA |
| **4** | LAX | MCI |
| **…** | … | … |
| **58487** | SFO | DFW |
| **58488** | LAS | SFO |
| **58489** | SFO | SBA |
| **58490** | MSP | ATL |
| **58491** | SFO | BOI |

58492 rows × 2 columns

In [187…
```
sorted(flights.loc[0, ['ORG_AIR', 'DEST_AIR']])
```

Out[187…
```
['LAX', 'SLC']
```

---

- このレシピで最も重要！
- 発着と到着で2つのラベルがあるが、それを1つにまとめたい（発着A→到着B、発着B→や到着A、の2つを1つに）
- 発着の2空港を英字順にソートすれば、ラベルを1つに出来る
- それにはDataFrameの`apply`メソッドを使う。これはgroupbyのapplyメソッドとは違う。今回はgroupを使わない
- DataFrameのapplyメソッドには関数を渡さないといけない
- この場合、組み込みの`sorted`関数を使う
- デフォルトではこの関数はSeriesのカラムに適用される。
- よって計算の方向を`axis='index'`を使って行方向に変える（注：**axis=1がcolumnsではないメソッド例の1つ**）
- 先頭行をSeriesとしてsortedに渡した例（下図）：

:
```
sorted(flights.loc[0, ['ORG_AIR', 'DEST_AIR']])
```

:
```
['LAX', 'SLC']
```

- applyメソッドはsortedをこのように全行をイテレーションする
- 完了したら、どの行も独立にソート済み、カラム名は無意味になる

## ここから後はテキストどおりにいかなかった！

In [200…
```
# テキストと結果が違う。axis='index'が正しい。axis=1だとエラーになる（次の行参照）
# flights_sort = flights[['ORG_AIR', 'DEST_AIR']].apply(sorted, axis='index')
flights_sort = flights[['ORG_AIR', 'DEST_AIR']].apply(sorted, axis=1)
flights_sort
```

Out[200…
```
0        [LAX, SLC]
1        [DEN, IAD]
2        [DFW, VPS]
3        [DCA, DFW]
4        [LAX, MCI]
            ...
58487    [DFW, SFO]
58488    [LAS, SFO]
58489    [SBA, SFO]
58490    [ATL, MSP]
58491    [BOI, SFO]
Length: 58492, dtype: object
```

参考）

:
```
# cf.applyでaxis=1とするとNG!! Seriesを取得してしまう。sortedが効いてない
flights_sort = flights[['ORG_AIR', 'DEST_AIR']].apply(sorted, axis=1)
flights_sort
```

:
```
0        [LAX, SLC]
1        [DEN, IAD]
2        [DFW, VPS]
3        [DCA, DFW]
4        [LAX, MCI]
            ...
58487    [DFW, SFO]
58488    [LAS, SFO]
58489    [SBA, SFO]
58490    [ATL, MSP]
58491    [BOI, SFO]
Length: 58492, dtype: object
```

In [174…
```
# cf
flights.loc[0, ['ORG_AIR','DEST_AIR']]
```

Out[174…
```
ORG_AIR     LAX
DEST_AIR    SLC
Name: 0, dtype: object
```

In [175…
```
# cf
flights.loc[1, ['ORG_AIR','DEST_AIR']]
```

Out[175…
```
ORG_AIR     DEN
DEST_AIR    IAD
Name: 1, dtype: object
```

---

（4）各行を独立にソートしたので、カラム名が正しくない。より一般的な名前に変えて、都市間の全便数を求める

In [193…
```
rename_dict = {'ORG_AIR':'AIR1','DEST_AIR':'AIR2'}
# 上述したが、flights_sortがDFでなく、Seriesになっていた場合、この行でエラーになる
flights_sort = flights_sort.rename(columns=rename_dict)
flights_ct2 = flights_sort.groupby(['AIR1', 'AIR2']).size()
flights_ct2
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-193-d9bfe85019e1> in <module>
      1 rename_dict = {'ORG_AIR':'AIR1','DEST_AIR':'AIR2'}
      2 # 上述したが、flights_sortがDFでなく、Seriesになっていた場合、この行でエラーになる
----> 3 flights_sort = flights_sort.rename(columns=rename_dict)
      4 flights_ct2 = flights_sort.groupby(['AIR1', 'AIR2']).size()
      5 flights_ct2

TypeError: rename() got an unexpected keyword argument 'columns'
```

---

（5）アトランタとヒューストンの全便数を選び、（2）の値の和と合致するか検証する

In [ ]:
```
flights_ct2.loc[('ATL','IAH')]
```

---

（6）

---

（補足）