レシピ46 デカルト積の作成（P122）

理解にしにくかったため、丸写し

- SeriesやDataFrameが2つ互いに働きあう場合、それぞれのオブジェクトのインデックス（行、カラムも）は
- 演算の開始前にアラインメント（整列）される。
- このインデックスアラインメントは黙って行われ、pandasの初心者がとても驚く
- アラインメントでは、インデックスが同じものでない限り、常にデカルト積？が作られる

- デカルト積（直積）は数学用語で、集合論でよく使う
- 2集合のデカルト積とは、両方の集合の要素のあらゆる組み合わせすべてを指す
- 例えば、トランプの52枚は、13枚のランク（A,2,3....,Q,K）と4つのマークとのデカルト積だ

- デカルト積は、常に意図した結果であるとは限らないが、意図しない結果を招かないように
- いつどのようんにして作られるのかを知っておくことが非常に重要
- このレシピでは、重複があるものの全く同じではないインデックスをもつ2つのSeriesが
- 足し合わせると予測しない結果になることを学ぶ

（1）一部が同じ値の異なるインデックスをもつ2つのSeriesを作る

In [2]:
```
import pandas as pd
import numpy as np
s1 = pd.Series(index=list('aaab'), data=np.arange(4))
s1
```

Out[2]:
```
a    0
a    1
a    2
b    3
dtype: int64
```

In [20]:
```
s2 = pd.Series(index=list('cababb'), data=np.arange(6))
s2
```

Out[20]:
```
c    0
a    1
b    2
a    3
b    4
b    5
dtype: int64
```

（2）2つのSeriesを足し合わせてデカルト積を作る

In [21]:
```
s1 + s2
```

Out[21]:
```
a    1.0
a    3.0
a    2.0
a    4.0
a    3.0
a    5.0
b    5.0
b    7.0
b    8.0
c    NaN
dtype: float64
```

解説

- 数学的なデカルト積は、この2つのpandasオブジェクトのとは少し異なる
- s1のラベルa（3つ）が、s2のラベルa（2つ）と対になる
- 結果のSeriesには、対が6つのaラベル、3つのbラベル、1つのcラベルになる
- デカルト積は、同じインデックスラベルで起こる
- ラベルcの要素はSeriesのs2だけなので、pandasはs1にアラインメントするラベルがないので
- その値をデフォルトで欠損値にする
- インデックスラベルが、あるオブジェクトにしかない場合、pandasはデフォルトで欠損値にする
- これは、どちらのSeriesも値が整数値しかないのに、結果のSeriesのデータ型をfloatにする
- こうなるのは、Numpyの欠損値オブジェクト、np.nanがintではなくfloatでしか存在しないため
- SeriesとDataFrameは、同種数データセットでは、これはあまり違いをもたらさないが
- 巨大なデータセットではメモリ使用量に深刻な影響を与える

補足

- この例で、インデックスが同じ順序で同じ要素を含んでいたら例外的なことになる
- その場合、デカルト積はつくられない。インデックスがそのままの位置でアラインメントする

- 次のコードでは、要素が全く同じアラインメントをして、データ型が整数のままであることに注意

In [3]:
```
s1 = pd.Series(index=list('aaabb'), data=np.arange(5))
s2 = pd.Series(index=list('aaabb'), data=np.arange(5))
s1 + s2
```

Out[3]:
```
a    0
a    2
a    4
b    6
b    8
dtype: int64
```

- インデックスの要素は同じだが順序が異なる場合、デカルト積が作られる
- s2の順序を変えて同じ処理をする

In [4]:
```
s1 = pd.Series(index=list('aaabb'), data=np.arange(5))
s2 = pd.Series(index=list('bbaaa'), data=np.arange(5))
s1 + s2
```

Out[4]:
```
a    2
a    3
a    4
a    3
a    4
a    5
a    4
a    5
a    6
b    3
b    4
b    4
b    5
dtype: int64
```

- 同じ演算でpandasが非常に異なる結果を出すのが興味深い
- もしもpandasがデカルト積しか返せないとしたら、DataFrameのカラムの足し合わせという
- ごく単純な操作でも、要素の個数が爆発的に増えてしまう
- このレシピでは、Seriesの要素数が異なる。
- Ptyhonやその他のプログラミング言語では配列のようなデータ構造で
- 演算の次元の要素数が同じでないと演算を許さない
- pandasでは演算の前にインデックスをアラインメントすることで、演算を許す
