行列の演算

ここでは行列について成立する演算について説明します．

- 行列のスカラー倍
- 行列の足し算
- 行列の引き算
- 行列の積
- 行列とベクトルの積

これらの演算について順を追って学習します．

行列のスカラー倍

行列とスカラーとの間で掛け算を行うことができます．これを行列のスカラー倍といい，行列の各成分に対してスカラーの値を掛けます．今，(m,n)型行列$\mathbf{A}=(a_{ij})$に対して，左からスカラーkを掛けてみます．

$$k\mathbf{A}=k\begin{pmatrix}a_{11}&a_{12}&\cdots&a_{1n}\\a_{21}&a_{22}&\cdots&a_{2n}\\\vdots&\vdots&\ddots&\vdots\\a_{m1}&a_{m2}&\cdots&a_{mn}\end{pmatrix}=\begin{pmatrix}ka_{11}&ka_{12}&\cdots&ka_{1n}\\ka_{21}&ka_{22}&\cdots&ka_{2n}\\\vdots&\vdots&\ddots&\vdots\\ka_{m1}&ka_{m2}&\cdots&ka_{mn}\end{pmatrix}$$

このようにスカラーと行列の掛け算は，$k\mathbf{A}$というように，それらを並べて記載します．行列の各成分内の掛け算はスカラーどうしの掛け算なので左右を交換できます．これによりスカラーと行列との掛け算も左右を入替えても同じ結果となります．すなわち，$k\mathbf{A}=\mathbf{A}k$が成立します．

行列の加減算

行列どうしの加減算を行うには，行列の型が等しいことが前提となります．行列どうしの加減算は，各成分ごとの加減算で定義します．そのために行列の型が異なると演算の相手となる成分が見つからないので計算が出来ません．それでは，$\mathbf{A}=(a_{ij})_{mn}$と$\mathbf{B}=(b_{ij})_{mn}$との足し算は下記のようになります．

$$\mathbf{A}+\mathbf{B}=\begin{pmatrix}a_{11}&a_{12}&\cdots&a_{1n}\\a_{21}&a_{22}&\cdots&a_{2n}\\\vdots&\vdots&\ddots&\vdots\\a_{m1}&a_{m2}&\cdots&a_{mn}\end{pmatrix}+\begin{pmatrix}b_{11}&b_{12}&\cdots&b_{1n}\\b_{21}&b_{22}&\cdots&b_{2n}\\\vdots&\vdots&\ddots&\vdots\\b_{m1}&b_{m2}&\cdots&b_{mn}\end{pmatrix}$$

$$=\begin{pmatrix}a_{11}+b_{11}&a_{12}+b_{12}&\cdots&a_{1n}+b_{1n}\\a_{21}+b_{21}&a_{22}+b_{22}&\cdots&a_{2n}+b_{2n}\\\vdots&\vdots&\ddots&\vdots\\a_{m1}+b_{m1}&a_{m2}+b_{m2}&\cdots&a_{mn}+b_{mn}\end{pmatrix}$$

スカラーの交換則により行列どうしの足し算は可換$\mathbf{A}+\mathbf{B}=\mathbf{B}+\mathbf{A}$となります．

同様に引き算についても各成分ごとの引き算として定義されます．

$$\mathbf{A}-\mathbf{B}=\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix}-\begin{pmatrix} b_{11} & b_{12} & \cdots & b_{1n} \\ b_{21} & b_{22} & \cdots & b_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ b_{m1} & b_{m2} & \cdots & b_{mn} \end{pmatrix}$$

$$=\begin{pmatrix} a_{11}-b_{11} & a_{12}-b_{12} & \cdots & a_{1n}-b_{1n} \\ a_{21}-b_{21} & a_{22}-b_{22} & \cdots & a_{2n}-b_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1}-b_{m1} & a_{m2}-b_{m2} & \cdots & a_{mn}-b_{mn} \end{pmatrix}$$

二つの行列のサイズが異なる場合に加減算は定義されないことに注意してください．

行列の積

行列の演算において行列の積は最も特徴的で，行列の数学的構造を理解する上で最も重要な演算です．行列の積が行えるための条件があります．それは，左側の行列の列数と右側の行列の行数が一致することです．二つの行列を$\mathbf{A}=(a_{ij})_{lm}$と$\mathbf{B}=(b_{ij})_{mn}$として行列の積を次のように定義します．

$$\mathbf{AB}=\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1m} \\ a_{21} & a_{22} & \cdots & a_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ a_{l1} & a_{l2} & \cdots & a_{lm} \end{pmatrix}\begin{pmatrix} b_{11} & b_{12} & \cdots & b_{1n} \\ b_{21} & b_{22} & \cdots & b_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ b_{m1} & b_{m2} & \cdots & b_{mn} \end{pmatrix}$$

$$=\begin{pmatrix} \sum_{k=1}^{m} a_{1k}b_{k1} & \sum_{k=1}^{m} a_{1k}b_{k2} & \cdots & \sum_{k=1}^{m} a_{1k}b_{kn} \\ \sum_{k=1}^{m} a_{2k}b_{k1} & \sum_{k=1}^{m} a_{2k}b_{k2} & \cdots & \sum_{k=1}^{m} a_{2k}b_{kn} \\ \vdots & \vdots & \ddots & \vdots \\ \sum_{k=1}^{m} a_{lk}b_{k1} & \sum_{k=1}^{m} a_{lk}b_{k2} & \cdots & \sum_{k=1}^{m} a_{lk}b_{kn} \end{pmatrix}$$

覚え方としては，横×縦と理解してください．この定義により，左側の列数と右側の行数が一致することが前提条件であることが分かります．また行列の積は可換ではないこと $\mathbf{AB}\neq\mathbf{BA}$ に注意してください．

行列の積の定義では総和の記号（Σ）が使われているので，拒絶感を持つ方もいるでしょう．理解を深めるために，2次行列の場合の積を総和を展開した形を確認してみましょう．このパターンをご覧いただいて，行列の積についての計算の形を理解しましょう．

$$\mathbf{AB}=\begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}\begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix}=\begin{pmatrix} \begin{pmatrix} a_{11} & a_{12} \end{pmatrix}\begin{pmatrix} b_{11} \\ b_{21} \end{pmatrix} & \begin{pmatrix} a_{11} & a_{12} \end{pmatrix}\begin{pmatrix} b_{12} \\ b_{22} \end{pmatrix} \\ \begin{pmatrix} a_{21} & a_{22} \end{pmatrix}\begin{pmatrix} b_{11} \\ b_{21} \end{pmatrix} & \begin{pmatrix} a_{21} & a_{22} \end{pmatrix}\begin{pmatrix} b_{12} \\ b_{22} \end{pmatrix} \end{pmatrix}$$

$$=\begin{pmatrix} a_{11}b_{11}+a_{12}b_{21} & a_{11}b_{12}+a_{12}b_{22} \\ a_{21}b_{11}+a_{22}b_{21} & a_{21}b_{12}+a_{22}b_{22} \end{pmatrix}$$

行列とベクトルの積

n次の列ベクトルは$(n,1)$型の行列と見なすことができます．これにより行列とベクトルとの積を定義することができます．

行列 $\mathbf{A}=\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix}$ と列ベクトル$\mathbf{v}=\begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{pmatrix}$ との積$\mathbf{Av}$は下記のようになります．

$$\mathbf{Av}=\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix}\begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{pmatrix}=\begin{pmatrix} \sum_{k=1}^{n} a_{1k}v_k \\ \sum_{k=1}^{n} a_{2k}v_k \\ \vdots \\ \sum_{k=1}^{n} a_{mk}v_k \end{pmatrix}$$

またm次の行ベクトルは$(1,m)$型の行列と見なすことが出来るので行列に左から掛けることができます．

行ベクトル$\mathbf{w}=(w_1 w_2 \cdots w_m)$と行列$\mathbf{A}=(a_{ij})_{mn}$との積$\mathbf{wA}$は下記のようになります．

$$\mathbf{wA}=\begin{pmatrix} w_1 & v_2 & \cdots & v_m \end{pmatrix}\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix}=\begin{pmatrix} \sum_{k=1}^{m} w_k a_{k1} & \sum_{k=1}^{m} w_k a_{k2} & \cdots & \sum_{k=1}^{m} w_k a_{kn} \end{pmatrix}$$

行列演算の実際

それでは行列の演算を具体的な例で確認してみましょう． Pythonでの計算のために配列計算ライブラリーNumPyをimportします．

```
import numpy as np
```

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

行列のスカラー倍

まず最初は行列のスカラー倍について計算します．スカラーを$k=5$とし，行列 $\mathbf{A}=\begin{pmatrix}1&2&3&4\\5&6&7&8\\9&10&11&12\end{pmatrix}$ にスカラーを掛けてみます．

$$k\mathbf{A}=5\begin{pmatrix}1&2&3&4\\5&6&7&8\\9&10&11&12\end{pmatrix}=\begin{pmatrix}5\times1&5\times2&5\times3&5\times4\\5\times5&5\times6&5\times7&5\times8\\5\times9&5\times10&5\times11&5\times12\end{pmatrix}=\begin{pmatrix}5&10&15&20\\25&30&35&40\\45&50&55&60\end{pmatrix}$$

Pythonでスカラーと行列の掛け算を行うにはアスタリスク記号による2項演算「k*A」と記載します．まず，スカラー定数kと行列Aを作成します．行列はNumPyの配列オブジェクトです．

```
k = 5
A = np.array([[1,2,3,4],[5,6,7,8],[9,10,11,12]])
k*A
```

```
In [2]: k = 5
        A = np.array([[1,2,3,4],[5,6,7,8],[9,10,11,12]])
        k*A
```

```
Out[2]: array([[ 5, 10, 15, 20],
               [25, 30, 35, 40],
               [45, 50, 55, 60]])
```

Pythonによる計算結果は手計算による結果と一致しました．スカラーと行列の掛け算は左右入替えても同じ答えになります．念のために「A*k」についても計算してみます．

```
A*k
```

```
In [3]: A*k
```

```
Out[3]: array([[ 5, 10, 15, 20],
               [25, 30, 35, 40],
               [45, 50, 55, 60]])
```

また，行列をスカラーで割ることも可能です．この場合，行列の各成分の値をスカラーで割ったものが求める行列となります．行列のスカラー倍の定義で解釈すると，スカラーの逆数を掛けることと等しい計算結果となります．

$$\mathbf{A}\div k=\mathbf{A}\frac{1}{k}=\begin{pmatrix}\frac{1}{k}a_{11}&\frac{1}{k}a_{12}&\cdots&\frac{1}{k}a_{1n}\\\frac{1}{k}a_{21}&\frac{1}{k}a_{22}&\cdots&\frac{1}{k}a_{2n}\\\vdots&\vdots&\ddots&\vdots\\\frac{1}{k}a_{m1}&\frac{1}{k}a_{m2}&\cdots&\frac{1}{k}a_{mn}\end{pmatrix}$$

Pythonでも同じ結果を得ることができます．割り算は数値の場合と同じくスラッシュを用いて「A/k」と記載します．

```
A/k
```

In [4]:
```
A/k
```

Out[4]:
```
array([[ 0.2,  0.4,  0.6,  0.8],
       [ 1. ,  1.2,  1.4,  1.6],
       [ 1.8,  2. ,  2.2,  2.4]])
```

行列の理論においてスカラーを行列で割ることはできません．しかし，PythonのNumPyにおいて，スカラーkを行列$\mathbf{A}$で割る操作「k/A」を実行してもエラーになりません．この場合は，行列の各成分でkを割った値を持つ行列が生成されます．具体的に試してみましょう．

```
k/A
```

In [5]:
```
k/A
```

Out[5]:
```
array([[ 5.        ,  2.5       ,  1.66666667,  1.25      ],
       [ 1.        ,  0.83333333,  0.71428571,  0.625     ],
       [ 0.55555556,  0.5       ,  0.45454545,  0.41666667]])
```

このように数学理論上とは異なることもプログラミングの中では実用化されていることが多くあります．プログラミングは実践主義であり便利なので採用されているものなので，数学理論ではない部分は誤解の無いようにしましょう．

行列の加減算

次の2つの行列の加減算を計算します．

$$\mathbf{A} = \begin{pmatrix} 10 & 20 & 30 \\ 40 & 50 & 60 \\ 70 & 80 & 90 \end{pmatrix}, \mathbf{B} = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$$

まず最初に足し算を行います．

$$\mathbf{A} + \mathbf{B} = \begin{pmatrix} 10 & 20 & 30 \\ 40 & 50 & 60 \\ 70 & 80 & 90 \end{pmatrix} + \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} = \begin{pmatrix} 10+1 & 20+2 & 30+3 \\ 40+4 & 50+5 & 60+6 \\ 70+7 & 80+8 & 90+9 \end{pmatrix} = \begin{pmatrix} 11 & 22 & 33 \\ 44 & 55 & 66 \\ 77 & 88 & 99 \end{pmatrix}$$

これをPythonで実行すると次のようになります．まず，行列AとBをNumPyの配列として定義して，それらを足し合わせます．

```
A = np.array([[10,20,30],[40,50,60],[70,80,90]])
B = np.array([[1,2,3],[4,5,6],[7,8,9]])
A+B
```

In [6]:
```
A = np.array([[10,20,30],[40,50,60],[70,80,90]])
B = np.array([[1,2,3],[4,5,6],[7,8,9]])
A+B
```

Out[6]:
```
array([[11, 22, 33],
       [44, 55, 66],
       [77, 88, 99]])
```

次に引き算について確認します．

$$\mathbf{A}-\mathbf{B}=\begin{pmatrix}10&20&30\\40&50&60\\70&80&90\end{pmatrix}-\begin{pmatrix}1&2&3\\4&5&6\\7&8&9\end{pmatrix}=\begin{pmatrix}10-1&20-2&30-3\\40-4&50-5&60-6\\70-7&80-8&90-9\end{pmatrix}=\begin{pmatrix}9&18&27\\36&45&54\\63&72&81\end{pmatrix}$$

これをPythonで実行する場合は数値と同じく「A-B」となります．

```
A-B
```

```
In [7]: A-B
```

```
Out[7]: array([[ 9, 18, 27],
               [36, 45, 54],
               [63, 72, 81]])
```

このように行列の加減算は，同じ型の行列，すなわち，行の数と列の数が同じ場合に限り，成分どうしの加減算で計算することができます．

PythonのNumPyにおける計算も基本的には数学的定義に従っています．しかし，数学的厳密さからすると仕様が緩いところが多く見られます．例えば次の計算は数学的には定義されていませんが，Pythonでは実施することができます．

$$\begin{pmatrix}10&20&30\\40&50&60\\70&80&90\end{pmatrix}+\begin{pmatrix}2&4&6\end{pmatrix}$$

実際にPythonで実行してみると，次のCodeセルにあるように各行に$\begin{pmatrix}2&4&6\end{pmatrix}$を足す仕様になっていることが分かります．変数を定義せずに，配列の計算を直接記載してみます．

```
np.array([[10,20,30],[40,50,60],[70,80,90]]) + np.array([2,4,6])
```

```
In [8]: np.array([[10,20,30],[40,50,60],[70,80,90]]) + np.array([2,4,6])
```

```
Out[8]: array([[12, 24, 36],
               [42, 54, 66],
               [72, 84, 96]])
```

このように仕様が緩いので，プログラムの記載ミスがエラーにならず計算が進んでしまい間違えた結果を信じてしまう危険性もありますので，十分注意してください．

行列の積

行列の積は，左の行列の一行と右の行列の一列の成分を順番に取り出して積をとり総和をとる，という操作を全ての行と列について行っていきます．簡単な行列から始めて徐々に慣れていきましょう．

最初は2行2列の正方行列どうしの積を行いましょう．

$$\mathbf{A}=\begin{pmatrix}1&2\\3&4\end{pmatrix},\mathbf{B}=\begin{pmatrix}2&-3\\-1&5\end{pmatrix}$$

計算手順に沿って詳細に記載します．

$$\mathbf{AB} = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} \begin{pmatrix} 2 & -3 \\ -1 & 5 \end{pmatrix} = \begin{pmatrix} 1 \times 2 + 2 \times (-1) & 1 \times (-3) + 2 \times 5 \\ 3 \times 2 + 4 \times (-1) & 3 \times (-3) + 4 \times 5 \end{pmatrix} = \begin{pmatrix} 2 - 2 & -3 + 10 \\ 6 - 4 & -9 + 20 \end{pmatrix}$$
$$= \begin{pmatrix} 0 & 7 \\ 2 & 11 \end{pmatrix}$$

これをPythonで実行してみましょう．まずは，掛け合わせる行列をNumPyの配列として変数を定義します．行列の積を行うには専用のdot()メソッドを使用します．これを*ドット積*と呼んでいます．

```
A = np.array([[1,2],[3,4]])
B = np.array([[2,-3],[-1,5]])
A.dot(B)
```

```
In [9]: A = np.array([[1,2],[3,4]])
        B = np.array([[2,-3],[-1,5]])
        A.dot(B)

Out[9]: array([[ 0,  7],
               [ 2, 11]])
```

単位行列

次は3行3列の正方行列の積ですが，単位行列と呼ばれる特別な行列の積について計算してみます．

$$\mathbf{I}_3 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \mathbf{A} = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$$

ここで$\mathbf{I}_3$は3次元の単位行列と言います．単位行列という名前の理由は，任意の行列に掛けても変化させない働きがあるためです．

実際に$\mathbf{I}_3\mathbf{A}$を計算して確かめてみます．

$$\mathbf{I}_3\mathbf{A} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$$
$$= \begin{pmatrix} 1 \times 1 + 0 \times 4 + 0 \times 7 & 1 \times 2 + 0 \times 5 + 0 \times 8 & 1 \times 3 + 0 \times 6 + 0 \times 9 \\ 0 \times 1 + 1 \times 4 + 0 \times 7 & 0 \times 2 + 1 \times 5 + 0 \times 8 & 0 \times 3 + 1 \times 6 + 0 \times 9 \\ 0 \times 1 + 0 \times 4 + 1 \times 7 & 0 \times 2 + 0 \times 5 + 1 \times 8 & 0 \times 3 + 0 \times 6 + 1 \times 9 \end{pmatrix}$$
$$= \begin{pmatrix} 1 + 0 + 0 & 2 + 0 + 0 & 3 + 0 + 0 \\ 0 + 4 + 0 & 0 + 5 + 0 & 0 + 6 + 0 \\ 0 + 0 + 7 & 0 + 0 + 8 & 0 + 0 + 9 \end{pmatrix} = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$$

行列I3とAをNumPyの配列で定義して，ドット積を実行します．なお，n次の単位行列は，numpy.identity(次元数) 関数で生成できます．

```
I3 = np.identity(3)
A = np.array([[1,2,3],[4,5,6],[7,8,9]])
I3.dot(A)
```

In [10]:
```
I3 = np.identity(3)
A  = np.array([[1,2,3],[4,5,6],[7,8,9]])
I3.dot(A)
```

Out[10]:
```
array([[ 1.,  2.,  3.],
       [ 4.,  5.,  6.],
       [ 7.,  8.,  9.]])
```

計算結果は行列$\mathbf{A}$と等しくなっています．単位行列は左右どちらから掛けても結果は同じです．試しに$\mathbf{AI}_3$を実行してみます．

```
A.dot(I3)
```

In [11]:
```
A.dot(I3)
```

Out[11]:
```
array([[ 1.,  2.,  3.],
       [ 4.,  5.,  6.],
       [ 7.,  8.,  9.]])
```

単位行列は線形代数において重要な役割を果たします．ここに幾つかの次元における単位行列を掲載いたします．

$$\mathbf{I}_2 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \mathbf{I}_3 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \mathbf{I}_4 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \mathbf{I}_5 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

行と列の数が同じ行列を正方行列と言いますが，このように単位行列は，その対角線成分が1でその他の成分が0になっている正方行列です．一般に行列の次元が明確な場合は添え字を省略して単に$\mathbf{I}$と記載します．

行を置換する基本行列

次の行列を行列$\mathbf{A}$の左から掛けてみて，計算結果がどうなるか確認しましょう．

$$\mathbf{P}_3(1,2) = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \mathbf{P}_3(1,3) = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix}, \mathbf{P}_3(2,3) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

まず，$\mathbf{P}_3(1,2)\mathbf{A}$を計算しましょう．

$$\begin{aligned} \mathbf{P}_3(1,2)\mathbf{A} &= \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} \\ &= \begin{pmatrix} 0\times1+1\times4+0\times7 & 0\times2+1\times5+0\times8 & 0\times3+1\times6+0\times9 \\ 1\times1+0\times4+0\times7 & 1\times2+0\times5+0\times8 & 1\times3+0\times6+0\times9 \\ 0\times1+0\times4+1\times7 & 0\times2+0\times5+1\times8 & 0\times3+0\times6+1\times9 \end{pmatrix} \\ &= \begin{pmatrix} 0+4+0 & 0+5+0 & 0+6+0 \\ 1+0+0 & 2+0+0 & 3+0+0 \\ 0+0+7 & 0+0+8 & 0+0+9 \end{pmatrix} = \begin{pmatrix} 4 & 5 & 6 \\ 1 & 2 & 3 \\ 7 & 8 & 9 \end{pmatrix} \end{aligned}$$

計算結果を見ると行列$\mathbf{A}$の1行目と2行目を入替えていることが分かります．行列$\mathbf{P}_3(1,2)$自身も単位行列から1行目と2行目を入替えた行列になっています．

この計算をPythonで実行してみましょう．

In [12]:
```
P12 = np.array([[0,1,0],[1,0,0],[0,0,1]])
P12.dot(A)
```

Out[12]:
```
array([[4, 5, 6],
       [1, 2, 3],
       [7, 8, 9]])
```

行列の積の左右を入替えて$\mathbf{A}\mathbf{P}_3(1,2)$とすると，列の入替えになります．

$$\mathbf{A}\mathbf{P}_3(1,2) = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$
$$= \begin{pmatrix} 1\times0+2\times1+3\times0 & 1\times1+2\times0+3\times0 & 1\times0+2\times0+3\times1 \\ 4\times0+5\times1+6\times0 & 4\times1+5\times0+6\times0 & 4\times0+5\times0+6\times1 \\ 7\times0+8\times1+9\times0 & 7\times1+8\times0+9\times0 & 7\times0+8\times0+9\times1 \end{pmatrix}$$
$$= \begin{pmatrix} 0+2+0 & 1+0+0 & 0+0+3 \\ 0+5+0 & 4+0+0 & 0+0+6 \\ 0+8+0 & 7+0+0 & 0+0+9 \end{pmatrix} = \begin{pmatrix} 2 & 1 & 3 \\ 5 & 4 & 6 \\ 8 & 7 & 9 \end{pmatrix}$$

In [13]:
```
A.dot(P12)
```

Out[13]:
```
array([[2, 1, 3],
       [5, 4, 6],
       [8, 7, 9]])
```

一般にn次正方行列のi列目とj列目を入替える左行列を$P_n(i,j)$と表記します．このように，行列を掛けるという操作は変換を表すものです．翻って連立方程式に戻って考えると$P_n(i,j)$を左から掛けることは連立方程式の並び順でｉ番目の式とｊ番目の式を入替える操作を表しています．
