NumPyとpandas

1ブロックずつ実行してみましょう

はコメントを表します

NumPy の利用

NumPyを使うと、配列・行列の計算が可能となります。

NumPyでは ndarray というデータ形式(numpy配列)を扱います。
これは、Rのベクトルや行列に相当します。

1次元の配列を作る方法を以下に示します。

In [3]:
```
import numpy as np    #numpyをnpという名前で読み込む

#numpy配列を作成（リストから変換）
a = np.array([1, 2, 3])
A = np.array([4, 5, 6])

#Rのベクトルと同様に計算が可能
print("a+A", type(a+A), " : ", a+A)   #要素どうしの足し算
print("a*A", type(a*A), " : ", a*A)   #要素どうしの掛け算
print("a*5", type(a*5), " : ", a*5)   #5を掛ける
```

```
a+A <class 'numpy.ndarray'>  :  [5 7 9]
a*A <class 'numpy.ndarray'>  :  [ 4 10 18]
a*5 <class 'numpy.ndarray'>  :  [ 5 10 15]
```

2次元の行列を作る方法を以下に示します。

In [4]:
```
#リストを2次元のnumpy配列（行列）に変換
print("¥n", np.array([[1,2,3], [4,5,6], [7,8,9]] ))  #行列を作成

#.arange()でnumpy配列を作成
Nums = np.arange(4, 63, 2, dtype=np.int32)  #4から62まで間隔2で作成
print("¥narray:¥n", Nums )

Nums = np.insert(Nums, 0, 2)     #0番目に2を挿入
Nums = np.append(Nums, 64)       #最後に64を追加
print("array:¥n",    Nums )

Nums = np.reshape(Nums, (8, 4)) #8x4の行列に変換
print("¥narray:¥n", Nums )

print("¥nshape: ",  Nums.shape )      #.shapeはnumpy配列が持つメソッド
print("object type: ", type(Nums) )  #関数type()で型を表示
```

```
[[1 2 3]
 [4 5 6]
 [7 8 9]]

array:
 [ 4  6  8 10 12 14 16 18 20 22 24 26 28 30 32 34 36 38 40 42 44 46 48 50
 52 54 56 58 60 62]
array:
 [ 2  4  6  8 10 12 14 16 18 20 22 24 26 28 30 32 34 36 38 40 42 44 46 48
 50 52 54 56 58 60 62 64]

array:
 [[ 2  4  6  8]
 [10 12 14 16]
 [18 20 22 24]
 [26 28 30 32]
 [34 36 38 40]
 [42 44 46 48]
 [50 52 54 56]
 [58 60 62 64]]

shape:  (8, 4)
object type:  <class 'numpy.ndarray'>
```

添字を使って要素を参照できます。

In [5]:
```
print( Nums[2] )      #row 2（2次元の場合は行が1つの要素）
print( Nums[2:4] )    #row 2 と 3
print( Nums[1][2] )   #row 1, column 2（1番目の中の2番目）
print( Nums[1, 2] )   #row 2, column 1（カンマで区切る方法）
print( Nums[:, 2] )   #全てのrow, column 2
print( Nums[2, :] )   #row 2, 全てのcolumns
```

```
[18 20 22 24]
[[18 20 22 24]
 [26 28 30 32]]
14
14
[ 6 14 22 30 38 46 54 62]
[18 20 22 24]
```

In [6]:
```
M = np.array([[2,3],[-1,-2]])
M
```

Out[6]:
```
array([[ 2,  3],
       [-1, -2]])
```

In [7]:
```
# 行列式

print("Mの行列式　＝ ",np.linalg.det(M))
```

```
Mの行列式　＝  -1.0
```

In [8]:
```
# 行列ランク

print("Mのランク　＝ ",np.linalg.matrix_rank(M))
```

```
Mのランク　＝  2
```

In [12]:
```
# 逆行列
print(M)
Minv = np.linalg.inv(M)
print("Mの逆行列　＝ ")
print(Minv)
```

```
[[ 2  3]
 [-1 -2]]
Mの逆行列　＝
[[ 2.  3.]
 [-1. -2.]]
```

In [14]:
```
# M * Inv(M) = I
np.matmul(M,Minv)
```

Out[14]:
```
array([[1., 0.],
       [0., 1.]])
```

In [15]:
```
# 固有値行列と固有ベクトル
print("Mの固有値行列と固有ベクトル　＝ ")
print(np.linalg.eig(M))
```

```
Mの固有値行列　＝
(array([ 1., -1.]), array([[ 0.9486833 , -0.70710678],
       [-0.31622777,  0.70710678]]))
```

In [17]:
```
M
```

Out[17]:
```
array([[ 2,  3],
       [-1, -2]])
```

$$\begin{cases} 2x + 3y = 8 \\ -x - 2y = 3 \end{cases}$$

In [16]:
```
# 連立方程式
b = [8,3]
solution = np.linalg.solve(M,b)
print(solution)
```

```
[ 25. -14.]
```

In [18]:
```
np.allclose(np.dot(M, solution), b)
```

Out[18]:
```
True
```

In [19]:
```
b1=[1,2]
np.allclose(np.dot(M,b1), b)
```

Out[19]:
```
False
```

演習

$$M = \begin{pmatrix} 1 & 2 & -1 \\ 3 & 1 & 1 \\ 1 & -1 & 2 \end{pmatrix}$$

- 行列式
- ランク
- 逆行列
- M * Inv(M) = I を確かめよう
- 固有値行列と固有ベクトル

$$\begin{cases} x + 2y - z = -3 \\ 3x + y + z = 4 \\ x - y + 2z = 6 \end{cases}$$

- 連立方程式の解を求めよう

In [ ]:
```

```

In [ ]:

pandas の利用

pandasを使うと、データフレームの利用が可能となります。

Starbucks店の飲み物はサイズによる価格が異なる。

コップサイズ(cup)と液量（液量オンス:fl.oz）と価格(USD)のデータをデータフレームに入れる。

1 fl.ozの価格を計算し、新たな列(OuncePrice (cents))に入れる。

データフレームを作る方法を以下に示します。

In [18]:

```
import pandas as pd    #pandasをpdという名前で読み込む
from numpy import nan  #numpyの欠損値機能を使う
import numpy as np

#データフレームの作成
DFSize = pd.DataFrame({"cup"   : ["Kids", "Short", "Medium", "Tall", "Grand"],
                       "fl.oz" : [7, 10, 14, 18, 24],
                       "USD"   : [nan, 2.45, 2.85, 3.25, 3.65]} )
#列の参照
print( DFSize['cup'] )

#データフレームの列どうしの演算
DFSize['OuncePrice(cents)'] = np.round(DFSize['USD'] / DFSize['fl.oz']*100)
DFSize['OuncePrice(cents)']=DFSize['OuncePrice(cents)'].astype(pd.Int64Dtype())
print( "¥nobject type:¥n", type(DFSize) )  #型を表示
display( DFSize )                          #データフレームを表示
```

```
0      Kids
1     Short
2    Medium
3      Tall
4     Grand
Name: cup, dtype: object

object type:
 <class 'pandas.core.frame.DataFrame'>
```

	cup	fl.oz	USD	OuncePrice(cents)
0	Kids	7	NaN	<NA>
1	Short	10	2.45	25
2	Medium	14	2.85	20
3	Tall	18	3.25	18
4	Grand	24	3.65	15

添字を使って要素を参照できます。
数字による参照では、.iloc を使います。

In [19]:

```
#添え字での参照（.ilocを使う）
print( "¥n", DFSize.iloc[1, 2] )     #row 1, column 2
print( "¥n", DFSize.iloc[0:2, :] )   #row 0:2, 全てのcolumn
print( "¥n", DFSize.iloc[:, 2] )     #全てのrow、column 2
```

```
2.45

    cup  fl.oz   USD  OuncePrice(cents)
0   Kids      7   NaN                <NA>
1  Short     10  2.45                  25

 0     NaN
1    2.45
2    2.85
3    3.25
4    3.65
Name: USD, dtype: float64
```

以下では、条件に該当する行を抽出しています。

In [20]:

```
#条件式での抽出
print( "¥n", DFSize[DFSize.cup == "Tall"] ) #Cupが"Tall"のものを抽出
print( "¥n", DFSize[DFSize.USD <= 3.0] )    #価格が3.0以下のものを抽出
```

```
    cup  fl.oz   USD  OuncePrice(cents)
3  Tall     18  3.25                  18

       cup  fl.oz   USD  OuncePrice(cents)
1    Short     10  2.45                  25
2  Medium     14  2.85                  20
```

In [ ]:

演習

このデータを日本で使う単位に変換しましょう。

コップのサイズは[S, M, L]あります。["Short", "Medium", "Tall"]に相当する。新たな列Japan_cupでこのサイズを入れる。対応がなければ、nanを入れる。

液量（液量オンス:fl.oz）はmlに変換し、新な列mlに入れる。

1USDは110円とする。それぞれの価格を変換し、新たな列(Yen)で入れる。

OuncePrice(cents)も新たな列OuncePrice(yen)で入れる。

データフレームは以下の様に示します。

In [40]:
```
DFSize
```

Out[40]:

	cup	fl.oz	USD	OuncePrice(cents)	Japan_cup	ml	Yen	OuncePrice(Yen)
0	Kids	7	NaN	<NA>	NaN	207.0	<NA>	<NA>
1	Short	10	2.45	25	S	296.0	269	27
2	Medium	14	2.85	20	M	414.0	313	22
3	Tall	18	3.25	18	L	532.0	357	19
4	Grand	24	3.65	15	NaN	710.0	401	16

In [ ]:
```
print(DFSize.dtypes)
```

cup object fl.oz int64 USD float64 OuncePrice(cents) Int64 Japan_cup object ml float64 Yen Int64 OuncePrice(Yen) Int64 dtype: object

In [39]: