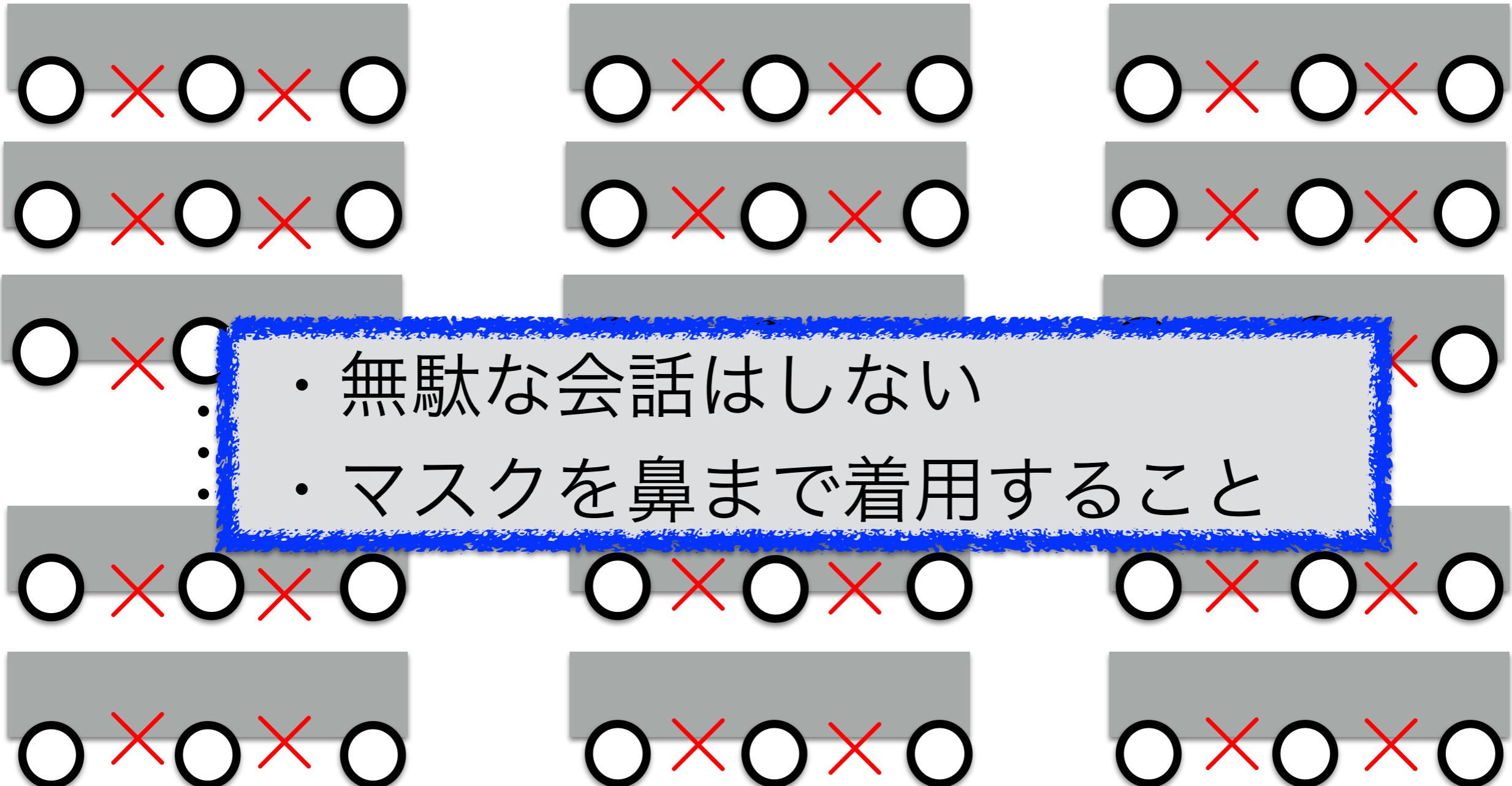


# Lecture 10

プログラミング演習Ⅰ その10

# 教室での座席について

前



後

- ・ 無駄な会話はしない
- ・ マスクを鼻まで着用すること

# 注意事項

1. 授業のスライドはHPからダウンロードすることができます、毎回の演習問題の回答例などは、自分で打ってください。
2. スライド以外の授業内容もあるので、ノートを取ること。
3. 授業中にスマートフォンやLineなどは使用禁止（演習問題の回答の写真撮影も不可！）
4. 偽造出席や出席名簿改竄（両者）＝ F.

# 本日の演習の流れ

1. Pythonによる配列の利用
2. Pythonによるグラフの表示
3. Pythonのnumpyの利用について

course page

<http://amth.mind.meiji.ac.jp/courses/PE1/>

# Pythonによる配列の利用

1

Pythonの配列はリスト(list)と呼ばれている。リストを作るには、角括弧にコンマで区切った値を指定する。以下の文は、リストを作り、それをラベルsimplelistで参照する。

```
>>> simplelist = [1, 2, 3]
```

C言語と同じく、simplelist[0]は最初の数、simplelist[1]が2番目の数、simplelist[2]が3番目の数を指す。

```
>>> simplelist[0]  
1  
>>> simplelist[1]  
2  
>>> simplelist[2]  
3
```

# Pythonによる配列の利用

リストのデータの型に特に制約はなく、計算式を含む、任意の型をデータにできる。以下を実行してみる。

```
[>>> my_list = [4, 6, "big fish", 1+math.sqrt(2), [3,4]]  
[>>> my_list[2]="penguin"  
[>>> print(my_list)  
[4, 6, 'penguin', 2.414213562373095, [3, 4]]
```

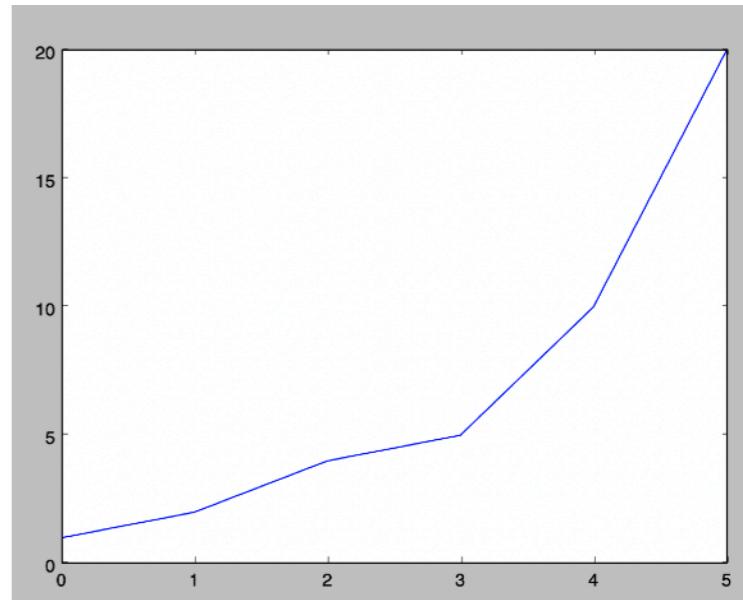
この例では、最初に定義したリストのmy\_listのデータのうち、3番目のデータを文字列「penguin」に変更している。

# Pythonによるグラフの表示

2 次のプログラムを実行してみよう。

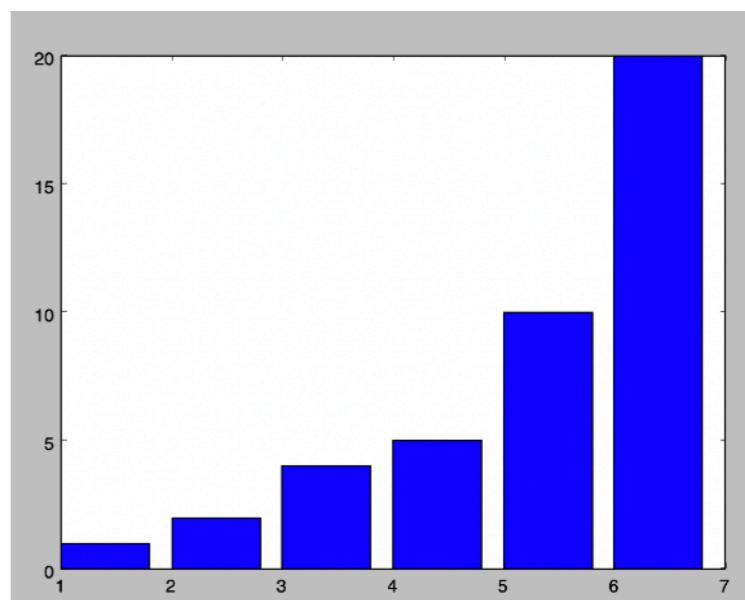
## 折れ線グラフ(line graph)

```
1 import matplotlib as mpl  
2 import matplotlib.pyplot as plt  
3 import numpy as np  
4 a = [1,2,4,5,10,20]  
5 plt.plot(a)  
6 plt.show()
```



## 棒グラフ(bar graph)

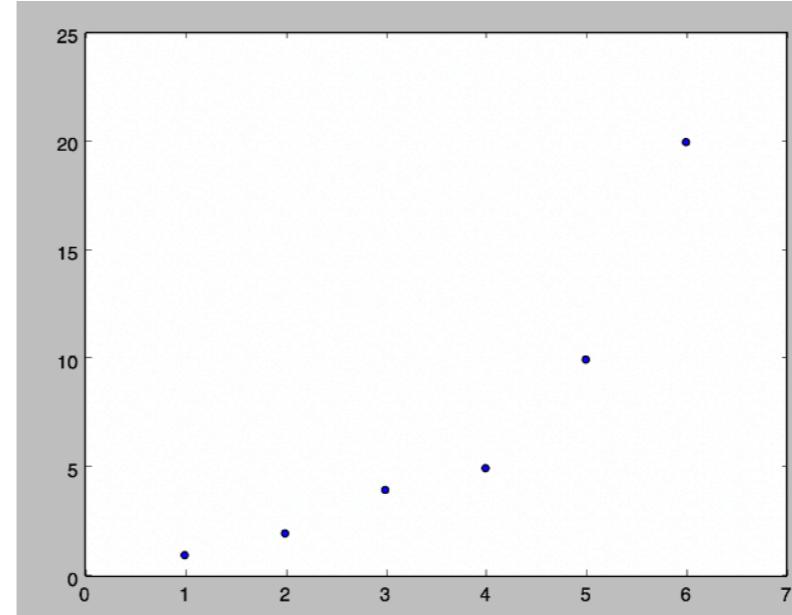
```
1 import matplotlib as mpl  
2 import matplotlib.pyplot as plt  
3 import numpy as np  
4 a = [1,2,4,5,10,20]  
5 x = list(range(1,7))  
6 plt.bar(x,a)  
7 plt.show()
```



# Pythonによるグラフの表示

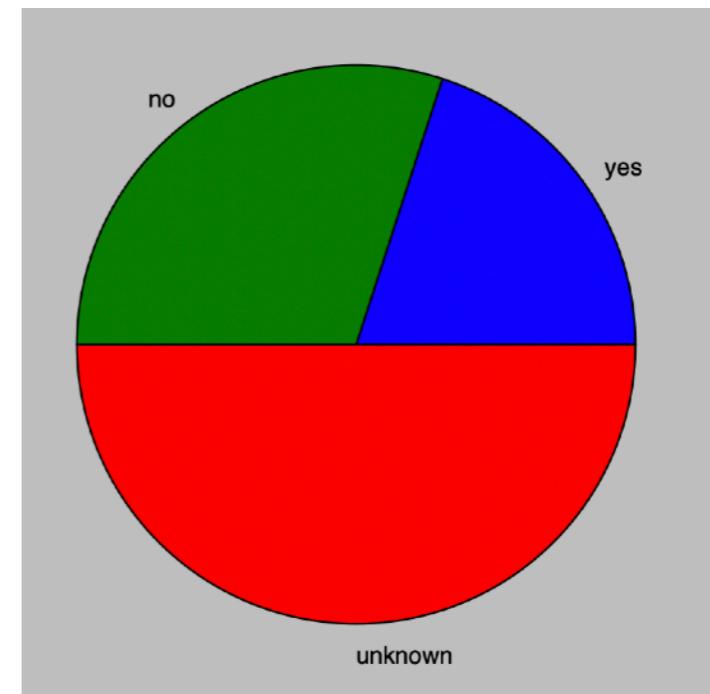
## 散布図(scatter plot)

```
1 import matplotlib as mpl  
2 import matplotlib.pyplot as plt  
3 import numpy as np  
4 a = [1,2,4,5,10,20]  
5 x = list(range(1,7))  
6 plt.scatter(x,a)  
7 plt.show()
```



## 円グラフ(pie chart)

```
1 import matplotlib as mpl  
2 import matplotlib.pyplot as plt  
3 import numpy as np  
4 plt.axis('equal')  
5 data =[20,30,50]  
6 label =["yes","no","unknown"]  
7 plt.pie(data,labels=label)  
8 plt.show()
```



# numpyの利用について

3 Numpyを入れるとデータ処理（配列のsort等）やベクトル、行列の計算が簡単にできる。以下を試してみよう。

(a) `>>> import numpy as np`  
`>>> np.random.rand()` ← rand() ~ 一様分布  
0.9422169198710257  
(uniform distribution)  
`>>> np.random.randn()` ← randn() ~ 正規分布  
-1.0082794189486057  
(normal distribution)

(b) `>>> x = np.random.rand(2)` ←  $x \in \mathbf{R}^2$   
`>>> x`  
`array([ 0.90143478, 0.06021542])`  
`>>> x[0]`  
0.90143478423721635  
`>>> x[1]`  
0.060215417526835235

# numpyの利用について

- (c) `>>> v = [1,1]`  $\longleftrightarrow v \cdot v \in \mathbf{R}$   
`>>> np.dot(v,v)`  
2  
dot ~ dot product ~ 内積
- (d) `>>> C = [[1,2],[3,4]]`  $\longleftrightarrow C \in \mathbf{R}^{2 \times 2}$   
`>>> D = [[1,0],[0,1]]`  $\longleftrightarrow D \in \mathbf{R}^{2 \times 2}$   
`>>> np.dot(C,D)`  $\longleftrightarrow AB$   
`array([[1, 2],`  
 `[3, 4]])`
- (e) `>>> np.add(C,D)`  $\longleftrightarrow C + D$   
`array([[2, 2],`  
 `[3, 5]])`

# numpyの利用について

(f)  $\ggg \text{np.subtract}(C, D) \quad \leftarrow C - D$

```
array([[0, 2],  
       [3, 3]])
```

(g)  $\ggg \text{np.linalg.inv}(C) \quad \leftarrow C^{-1}$

```
array([-2., 1.,  
      [1.5, -0.5]])
```

linalg ~ linear algebra ~ 線形代数

$a \text{ e-}b \implies a \times 10^b$

(h)  $\ggg N = \text{np.linalg.inv}(C)$

```
>>> np.dot(C, N)
```

```
array([[ 1.00000000e+00, 1.11022302e-16],  
      [ 0.00000000e+00, 1.00000000e+00]])
```

(i)  $\ggg v = [1, 1] \quad \leftarrow Cv \in \mathbf{R}^2$

```
>>> np.dot(C, v)
```

```
array([3, 7])
```

# numpyの利用について

```
(j)  >>> A = np.random.rand(3,3)           ←  $A \in \mathbf{R}^{3 \times 3}$ 
>>> print(A)
[[ 0.32285224  0.03981743  0.07404115]
 [ 0.06502553  0.36005369  0.08960305]
 [ 0.63606001  0.52256122  0.84691684]]
```

```
>>> B = np.random.rand(3,3)           ←  $B \in \mathbf{R}^{3 \times 3}$ 
>>> print(B)
[[ 0.07763798  0.39508383  0.77555902]
 [ 0.84706611  0.78036553  0.27230131]
 [ 0.21600063  0.3476755   0.17589932]]
```

```
>>> M = np.dot(A,B)           ←  $M = AB$ 
>>> M
array([[ 0.07478653,  0.18436815,  0.2742571 ],
       [ 0.32939205,  0.33781681,  0.16423534],
       [ 0.67496089,  0.95353803,  0.78456828]])
```

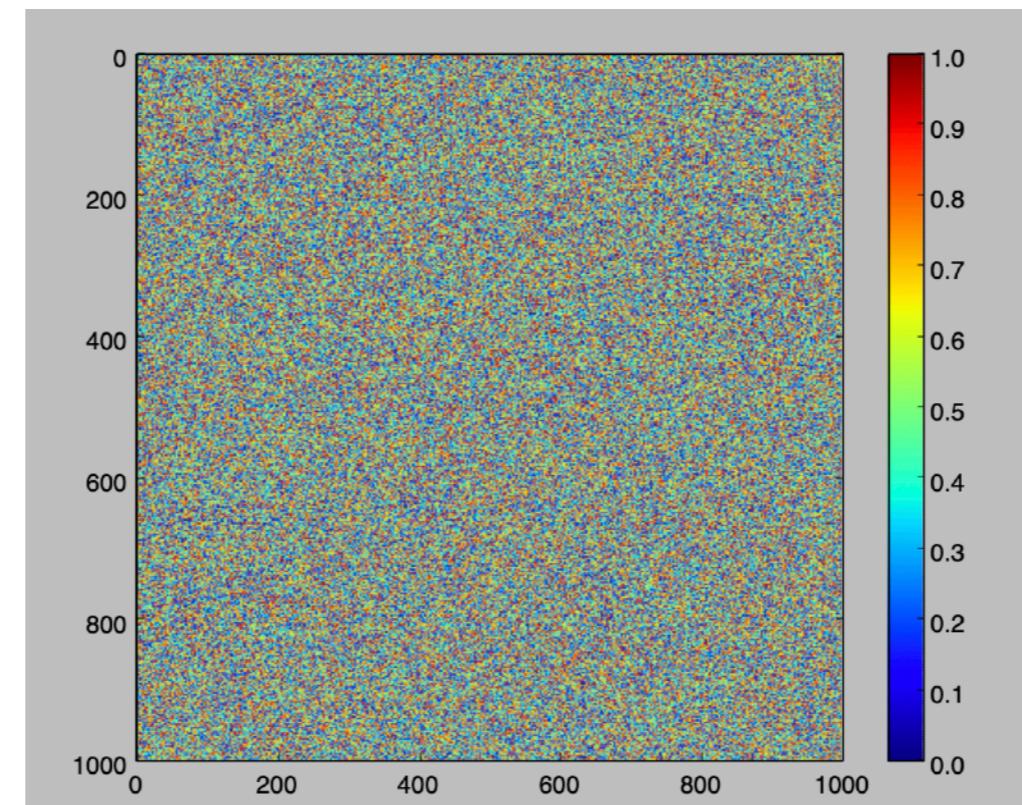
# numpyの利用について

- (k) [">>>> np.linalg.eigvals(C) ← eigvals ~ eigenvalues ~ 固有値  
array([-0.37228132, 5.37228132])  
 $\det(C - \lambda I) = 0$
- (l) [">>>> np.linalg.det(C) ← det ~ determinant ~ 行列式  
-2.000000000000004
- (m) [">>>> np.linalg.eigvals(A)  
array([ 1.00127506, 0.23328225, 0.29526547])
- (n) [">>>> np.linalg.det(A)  
0.068968019332671909

# numpyの利用について

(o) `S = np.random.rand(1000,1000)`  $\leftarrow S \in \mathbf{R}^{1000 \times 1000}$

(p) `>>> plt.imshow(S)`  
`<matplotlib.image.AxesImage object at 0x114e24290>`  
`>>> plt.colorbar()`  
`<matplotlib.colorbar.Colorbar instance at 0x114ec0cf8>`  
`>>> plt.show()`



# Pythonによる乱数生成

```
(q) >>> np.linalg.eigvals(S)
array([ 4.99633539e+02+0.j           ,  7.81088402e+00+5.13770989j,
       7.81088402e+00-5.13770989j, -2.54295801e+00+9.03587653j,
      -2.54295801e+00-9.03587653j, -7.46205401e+00+5.47670587j,
      -7.46205401e+00-5.47670587j, -6.56763583e+00+6.50007918j,
      -6.56763583e+00-6.50007918j, -8.76540677e+00+2.86394097j,
      -8.76540677e+00-2.86394097j,  5.01438469e+00+7.60280261j,
                                         .
                                         .
                                         .
      1.42936759e-01+0.36477186j,  1.42936759e-01-0.36477186j,
     -6.35766696e-01+0.20233929j, -6.35766696e-01-0.20233929j,
     -4.93136832e-01+0.3442304j , -4.93136832e-01-0.3442304j ,
     -1.77903623e-01+0.j         , -9.83810744e-03+0.j        ])
```

# Pythonによる乱数生成

(r) ソートの処理も簡単！

```
>>> r = np.random.rand(6)
>>> print(r)
[ 0.55494205  0.02610252  0.23251614  0.32455951  0.49574795  0.70076266]

>>> np.sort(r)
array([ 0.02610252,  0.23251614,  0.32455951,  0.49574795,  0.55494205,
       0.70076266])
```

# Pythonによる乱数生成

## サイコロ投げをシミュレーションする

6面サイコロ投げをシミュレーションするには、1から6の乱数を生成する方法が要ります。

- ・ Python標準ライブラリのrandomモジュールは、乱数を生成する様々な関数を提供しています。
- ・ randint()関数は2つの整数を引数として、その2つの数の（両端を含む）間のランダムな整数を返します。

4

```
>>> import random  
>>> random.randint(1,6)
```

(再度呼び出すと、異なる数を返します)

- ・ randint()は、下限を最初に与えるものと想定しているので、rand(6,1)が不当であることに注意して下さい。

# Pythonによる乱数生成

5 次のプログラムは、目の合計が20になるまで、6面のサイコロを振り続ける簡単なサイコロゲームです。プログラムを理解した上で、実行して下さい。

```
1 import matplotlib.pyplot as plt
2 import random
3
4 def roll():
5     return random.randint(1,6)
6
7 target_score = 20
8 score = 0
9 num_rolls = 0
10 while score < target_score:
11     die_roll = roll()
12     num_rolls = num_rolls + 1
13     print('Rolled: {}'.format(die_roll))
14     score = score + die_roll
15
16 print('You scored {} in {} rolls'.format(score,num_rolls))
```

# Pythonによる乱数生成



乱数を使用したゲームを作成せよ。