ベクトルと行列

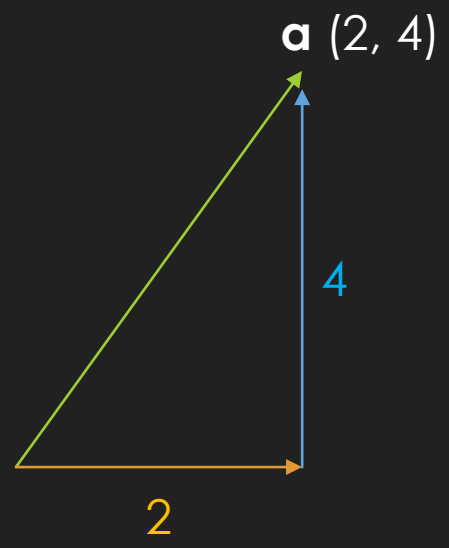
情報科　飯島　涼

本日の内容

- numpy のインストール
- ベクトル
- 行列

ベクトルとは?

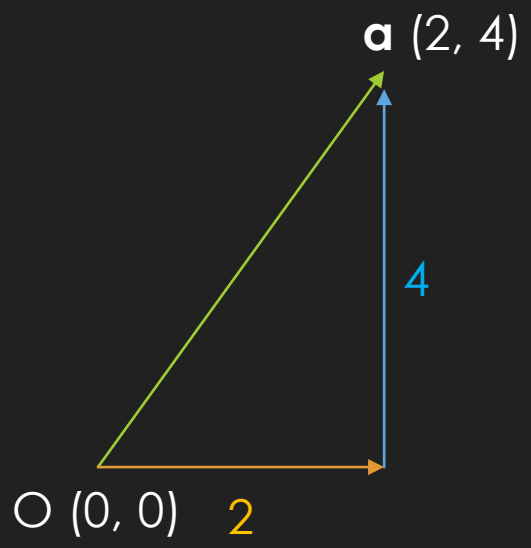
向きと大きさを表す量のこと　（太字のアルファベットでベクトルを表すことにします）



※ 始点、終点の位置の情報は持たない

位置ベクトルとは?

- 始点が原点（0, 0）となるベクトルのこと（ベクトルの大きさで点の位置を表せる= 位置ベクトル）



具体的な応用例（情報系の中のみ）

- ゲームエンジン上でのキャラクター、オブジェクトの位置の情報
- PCの画面上でのwindowの位置指定、描画

ベクトルの基本的な演算

```
1 ! pip install numpy
2 import numpy as np
3
4 a = np.array([2, 2])
5 b = np.array([2, -1])
6
7 print(a+b)
8 print(a-b)
9 print(2*a) # ベクトルの定数倍
10
```

```
Requirement already satisfied: numpy in /usr/local/lib/python3.6/dist-packages (1.18.5)
[4 1]
[0 3]
[4 4]
```

ベクトルの絶対値・内積

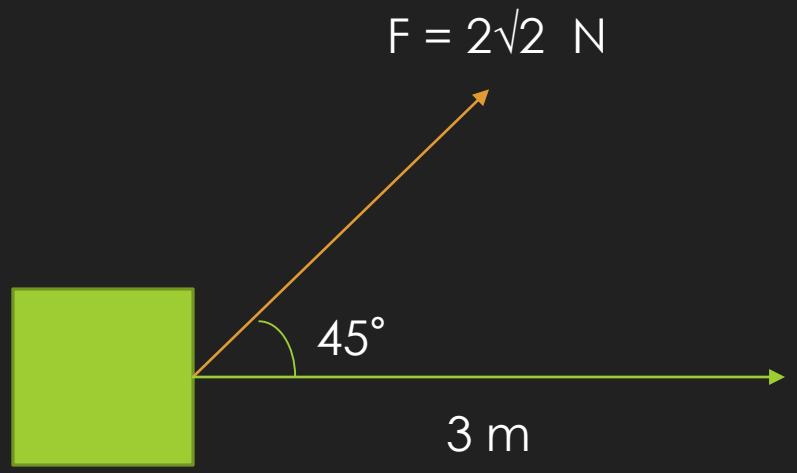
```python
import numpy as np

a = np.array([2, 2])
b = np.array([2, -1])

norm_a = np.linalg.norm(a)
norm_b = np.linalg.norm(b)
print("a, bの絶対値:", norm_a,":", norm_b)

dot_ab = np.dot(a, b)
print("a, bの内積:", dot_ab)
```

```
a, bの絶対値: 2.8284271247461903 : 2.23606797749979
a, bの内積: 2
```

ベクトル同士の角度を求めるには?

内積の「意味」を物理の知識で考える

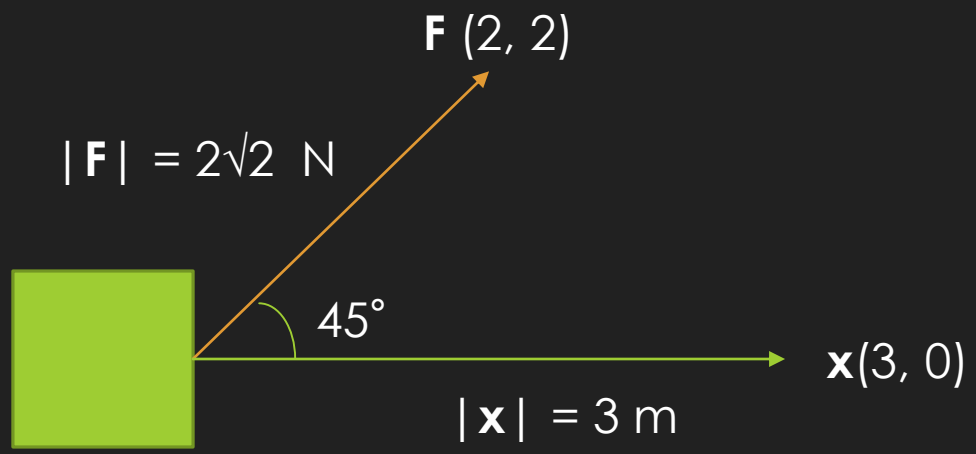
Q:水平面から45°の角度をつけて、 F = 2√2 N **の大きさ**で引っ張った時、水平方向に3m 動いたときの仕事は?



[def] 仕事: = 動いた距離 x 動く方向にかかった力

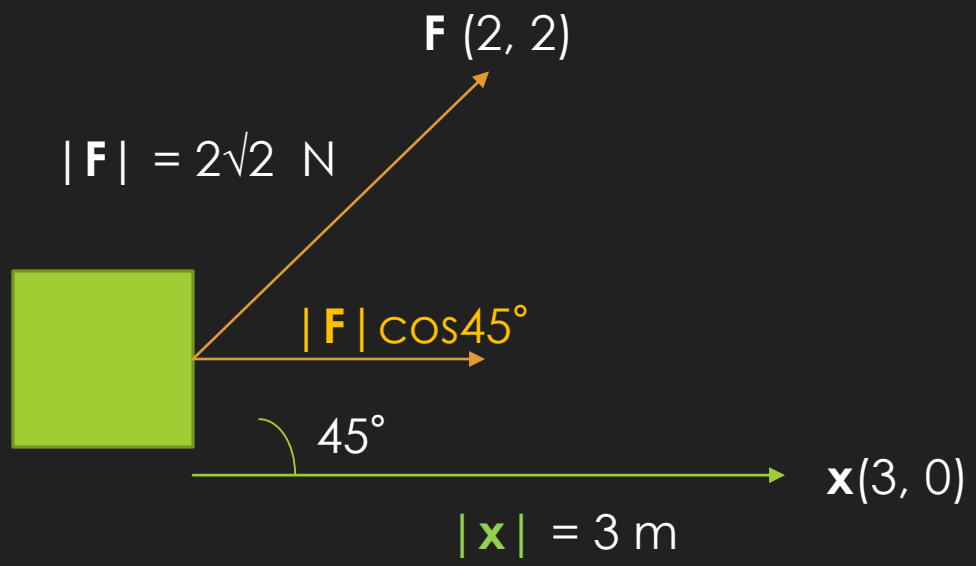
内積ってなんの意味がある？

それぞれの矢印をベクトルで表現してみる



内積ってなんの意味がある？

それぞれの矢印をベクトルで表現してみる



[def] 仕事: = 動いた距離 x 動く方向にかかった力

= | x | | F | cos 45°

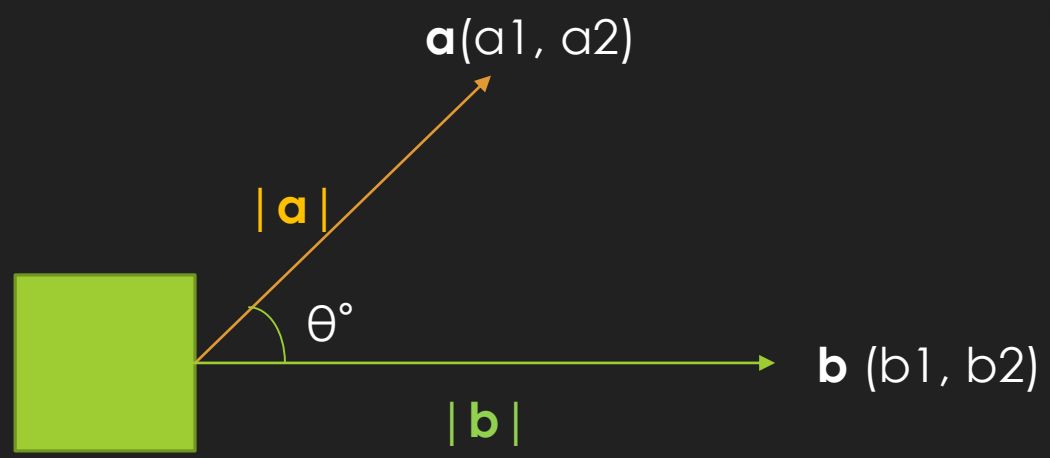
物理における内積＝仕事　ベクトル間の角度

$\boldsymbol{a} = \begin{pmatrix} a_1 \\ a_2 \end{pmatrix}, \mathbf{b} = \begin{pmatrix} b \\ b_2 \end{pmatrix}$

$$\boldsymbol{a} \cdot \mathbf{b} = a_1 b_1 + a_2 b_2 = |\boldsymbol{a}||\boldsymbol{b}| cos\theta$$

$$cos\theta = \frac{\boldsymbol{a} \cdot \mathbf{b}}{|\boldsymbol{a}||\boldsymbol{b}|}$$

$$\theta = \arccos \frac{\boldsymbol{a} \cdot \mathbf{b}}{|\boldsymbol{a}||\boldsymbol{b}|}$$



【演習１】ベクトル間の角度

```python
import numpy as np
import math
a = np.array([2, 2])
b = np.array([2, -1])

norm_a = np.linalg.norm(a)
norm_b = np.linalg.norm(b)
print("a, bの絶対値:", norm_a,":", norm_b)

dot_ab = np.dot(a, b)
print("a, bの内積:", dot_ab)

```

```
a, bの絶対値: 2.8284271247461903 : 2.23606797749979
a, bの内積: 2
ベクトル間の角度:
```

【解答１】ベクトル間の角度

```python
import numpy as np
import math
a = np.array([2, 2])
b = np.array([2, -1])

norm_a = np.linalg.norm(a)
norm_b = np.linalg.norm(b)
print("a, bの絶対値:", norm_a,":", norm_b)

dot_ab = np.dot(a, b)
print("a, bの内積:", dot_ab)

theta_rad = math.acos( dot_ab/(norm_a * norm_b) )
print("ベクトル間の角度: ", math.degrees(theta_rad))
```

```
a, bの絶対値: 2.8284271247461903 : 2.23606797749979
a, bの内積: 2
ベクトル間の角度:  71.56505117707799
```

行列

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

- データの保存
- 積和を要するデータの格納
- 統計・機械学習モデルの表現 <= 今注目されている部分

Pythonにおける行列

```python
import numpy as np

A = np.matrix([[1, 3], [5, 7]])
B = np.matrix([[2, 4], [6, 8]])

# 内積
print("AB= ¥n", A*B)

print("BA= ¥n", B*A)
```

```
AB=
 [[20 28]
 [52 76]]
BA=
 [[22 34]
 [46 74]]
```

$$A = \begin{pmatrix} 1 & 3 \\ 5 & 7 \end{pmatrix}$$

$$B = \begin{pmatrix} 2 & 4 \\ 6 & 8 \end{pmatrix}$$

```python
import numpy as np

A = np.matrix([[5, 3], [2, 1]])

# 逆行列を求める
Ainv = np.linalg.inv(A)
print(Ainv)
```

```
[[-1.  3.]
 [ 2. -5.]]
```

【演習2】連立方程式を解く

$$5x + 3y = 9$$
$$2x + y = 4$$

- を、行列を用いて表現し、解を求めてください。

- Numpy 以外のライブラリの使用は禁止

【演習2】連立方程式を解く

```python
import numpy as np

A = np.matrix([[5, 3], [2, 1]])
B = np.matrix([[9], [4]])
# 逆行列を求める
Ainv = np.linalg.inv(A)
print(Ainv)

# 連立方程式の解出力
print(Ainv * B )
```

```
[[-1.  3.]
 [ 2. -5.]]
[[ 3.]
 [-2.]]
```

【演習2.2】連立方程式を解く

$$5x - 4y + 6z = 8$$
$$7x - 6y + 10z = 14$$
$$4x + 9y + 7z = 74$$

手計算では面倒くさくなり始める => Python使うか!

【解答2.2】やり方は全く同じ

```python
import numpy as np

A = np.matrix([[5, -4, 6], [7, -6, 10],[4, 9, 7]])
B = np.matrix([[8], [14], [74]])
# 逆行列を求める
Ainv = np.linalg.inv(A)
print(Ainv)

# 連立方程式の解出力
print(Ainv * B )
```

```
[[ 1.29411765 -0.80392157  0.03921569]
 [ 0.08823529 -0.10784314  0.07843137]
 [-0.85294118  0.59803922  0.01960784]]
[[2.]
 [5.]
 [3.]]
```

行列による一次変換

行列の役割: 図形の移動、回転、縮小・拡大が可能

一次変換

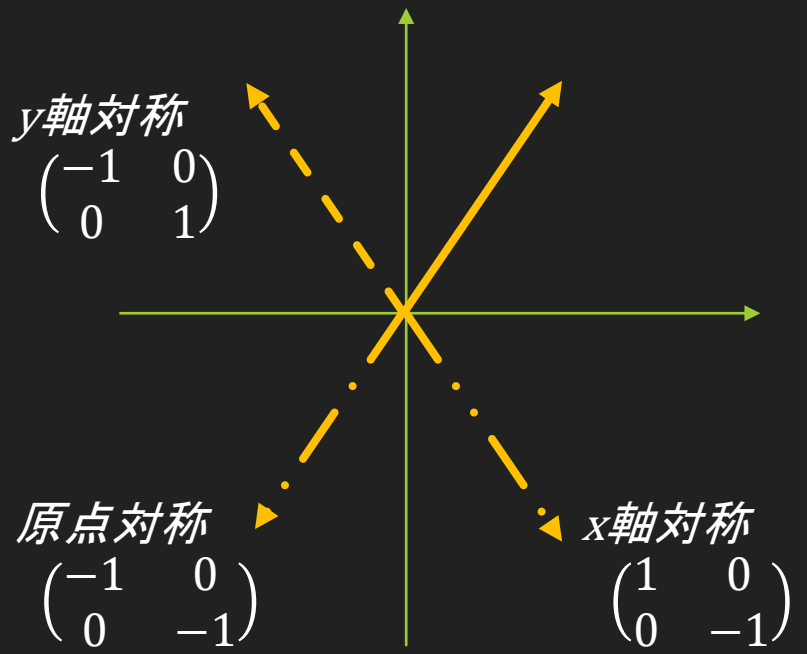
ベクトルに行列をかけて、その性質を見てみる

ここから先: ベクトルは、2行1列の行列とみなして、すべての計算を行列として扱います

$$\boldsymbol{a} = \begin{pmatrix} 2 \\ 3 \end{pmatrix} \qquad \boldsymbol{b} = \begin{pmatrix} 1 \\ -1 \end{pmatrix}$$

一次変換の行列

以下の行列をベクトルに左からかけると、示したような効果が表れる



一次変換　例

```python
import numpy as np

a = np.matrix([[2], [4]])

x_sy = np.matrix([[1, 0], [0, -1]])
y_sy = np.matrix([[-1, 0], [0, 1]])
o_sy = np.matrix([[-1, 0], [0, -1]])

print("元のベクトル:¥n", a)
print("x軸対称:¥n", x_sy*a)
print("y軸対称:¥n",y_sy*a)
print("原点対称:¥n",o_sy*a)
```

```
元のベクトル:
 [[2]
 [4]]
x軸対称:
 [[ 2]
 [-4]]
y軸対称:
 [[-2]
 [ 4]]
原点対称:
 [[-2]
 [-4]]
```

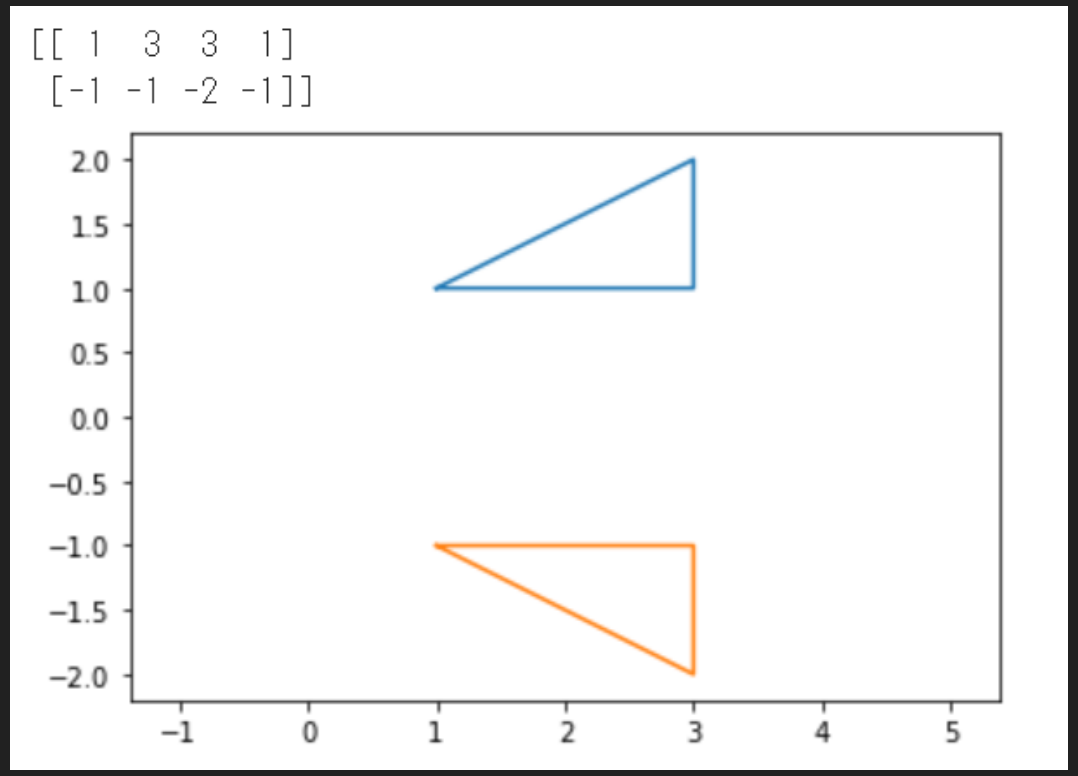
【演習4】ベクトルの縮小・拡大・回転を表現するにはどのような行列が必要か？

- 実際に実装してみて、縮小・拡大・回転が確認できることを示すプログラムを作成してください。

x軸対称の三角形を描画するグラフ

```python
import numpy as np
import matplotlib.pyplot as plt
import math

# 三角形の頂点を1列ずつ並べていく
p = np.matrix([[1, 3, 3, 1], [1, 1, 2, 1]])

A = np.matrix([[1, 0], [0, -1]])

# 一次変換 x軸対称
p2 = A * p
print(p2)

p = np.array(p)
p2 = np.array(p2)

plt.plot(p[0, :], p[1, :])
plt.plot(p2[0, :], p2[1, :])
plt.axis("equal")
plt.show()
```

```
[[ 1  3  3  1]
 [-1 -1 -2 -1]]
```



【自由課題・簡単】3Dゲームの世界

3次元の座標上に配置されたキャラクターの位置を座標で管理するゲームがある。ここで、常にキャラクターたちは地平におり、目線の高さは同じという仮定をおいて、常にz=0 (または固定値、const)の空間を仮定して話を進める。

今、プレイヤーは 座標(5, 4) に位置しており、座標(7, 2) に位置するNPC(ノンプレイヤーキャラクター)に話しかけようとしている。NPCが向いている**向き**を単位ベクトル(1, 0)で表現したとき、NPCとプレイヤーが向かい合うためには、NPCはどの程度回転する必要があるか。

ここで、プレイヤーはすでに最適な角度を向いている状態になっているものとする。

NPCが向いている**向き**を単位ベクトル(1, 0)と表しているので、この単位ベクトルが、NPCのいる座標を始点として考えるとわかりやすい

【自由課題・難】 図形の縮小・拡大・回転

三角形を縮小・拡大・回転させて、matplotlibで表示させるプログラムを作成してください。

ヒント: 図形のそれぞれの頂点や中心などの特徴的な個所の座標を、位置ベクトルとして表現して並べた行列を作る。

三角形はどのような形・位置にあっても構わない。自分がプログラムで表示しやすくなるように作って構わない。