

# 関数グラフの勘所



第一回全国高校おっぱい関数選手権大会

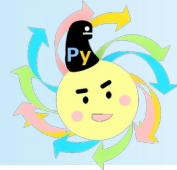
おっぱい関数甲子園 出場校募集

結論

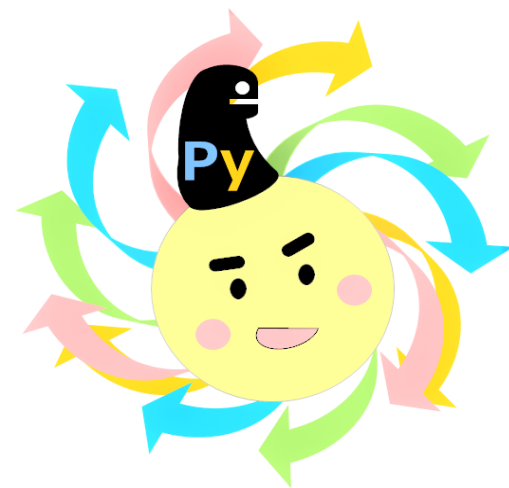
**numpy** は超便利！

どんどん使おう！

# おまえだれよ



- 田中丸 祐治 (たなかまる ゆうじ)
- Python好きの  
日曜大工的なんちゃって  
データサイエンティスト  
(要は、ただのサラリーマン)
- 備忘録代わりにTwitter やってます : @malo21st





関数

$$y = f(x)$$

従属変数

独立変数



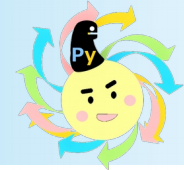
$y = f(x)$  ① 独立変数の用意

$y = f(x)$  ② 関数の計算と従属変数の出力

③ データを設定してグラフ化



# ① 独立変数 **X** の用意



(例) -1 から +1 まで、0.1 刻みの配列を作る

```
1 x_lst = []
2 for x in range(-10, 11):
3     x_lst.append(x/10)
4 print(x_lst)
```

```
[-1.0, -0.9, -0.8, -0.7, -0.6, -0.5, -0.4, -0.3, -0.2, -0.1, 0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0]
```

```
1 x_lst = [x/10 for x in range(-10, 11)]
2 print(x_lst)
```

```
[-1.0, -0.9, -0.8, -0.7, -0.6, -0.5, -0.4, -0.3, -0.2, -0.1, 0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0]
```

```
1 import numpy as np
2
3 x_seq = np.linspace(-1, 1, 21)
4 x_seq
```

```
array([-1. , -0.9, -0.8, -0.7, -0.6, -0.5, -0.4, -0.3, -0.2, -0.1, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0])
```

```
1 x_seq = np.arange(-1, 1.1, 0.1)
2 x_seq
```

```
array([-1.00000000e+00, -9.00000000e-01, -8.00000000e-01, -7.00000000e-01, -6.00000000e-01, -5.00000000e-01, -4.00000000e-01, -3.00000000e-01, -2.00000000e-01, -1.00000000e-01, -2.22044605e-16, 2.00000000e-01, 3.00000000e-01, 4.00000000e-01, 5.00000000e-01, 6.00000000e-01, 7.00000000e-01, 8.00000000e-01, 9.00000000e-01, 1.00000000e+00])
```

## ② 関数 **f** の計算 と 従属変数 **y** の出力



(例) 関数  $y = x^2$  を求めよ。ただし、 $-1 \leq x \leq 1$

```
1 y_lst = []
2 for x in x_lst:
3     y_lst.append(x**2)
4 print(y_lst)
```

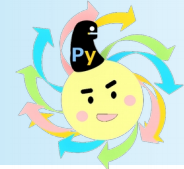
```
[1.0, 0.81, 0.6400000000000001, 0.48999999999999994, 0.25, 0.16000000000000003, 0.09, 0.04000000000000002, 0.0, 0.010000000000000002, 0.01, 0.09, 0.16000000000000003, 0.25, 0.36, 0.49, 0.6400000000000001, 0.81, 1.0]
```

```
1 y_lst = [x**2 for x in x_lst]
2 print(y_lst)
```

```
[1.0, 0.81, 0.6400000000000001, 0.48999999999999994, 0.25, 0.16000000000000003, 0.09, 0.04000000000000002, 0.0, 0.010000000000000002, 0.01, 0.09, 0.16000000000000003, 0.25, 0.36, 0.49, 0.6400000000000001, 0.81, 1.0]
```

```
1 import numpy as np
2
3
4 x_seq = np.arange(-1, 1.1, 0.1)
5
6 y_seq = x_seq ** 2
7
8 print(y_seq)
9
```

```
[1.00000000e+00  8.10000000e-01  6.40000000e-01
 3.60000000e-01  2.50000000e-01  1.60000000e-01
 4.00000000e-02  1.00000000e-02  4.93038066e-32
 4.00000000e-02  9.00000000e-02  1.60000000e-01
 3.60000000e-01  4.90000000e-01  6.40000000e-01
 1.00000000e+00]
```



# (補足) 関数 $f$ の計算 と 従属変数 $y$ の出力

```
1 def bmi(height, weight):
2     return weight / (height/100)**2
3
4 print("### 数値 ###")
5 bmi_val = bmi(167, 72)
6 print( bmi_val )
7
8 print("\n### numpyの配列 ###")
9 import numpy as np
10 height_seq = np.array([167, 183, 154])
11 weight_seq = np.array([72, 67, 56])
12 bmi_seq = bmi(height_seq, weight_seq)
13 print( bmi_seq )
14
15 print("\n### pandasのシリーズ(series) ###")
16 import pandas as pd
17 df = pd.DataFrame({'h':height_seq, 'w':weight_seq})
18 bmi_ser = bmi(df.h, df.w)
19 print( bmi_ser )
```

### 数値 ###

25.816630212628635

### numpyの配列 ###

[25.81663021 20.00656932 23.61275089]

### pandasのシリーズ(series) ###

0 25.816630

1 20.006569

2 23.612751

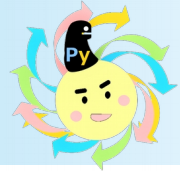
dtype: float64

1	df
	h w
0	167 72
1	183 67
2	154 56

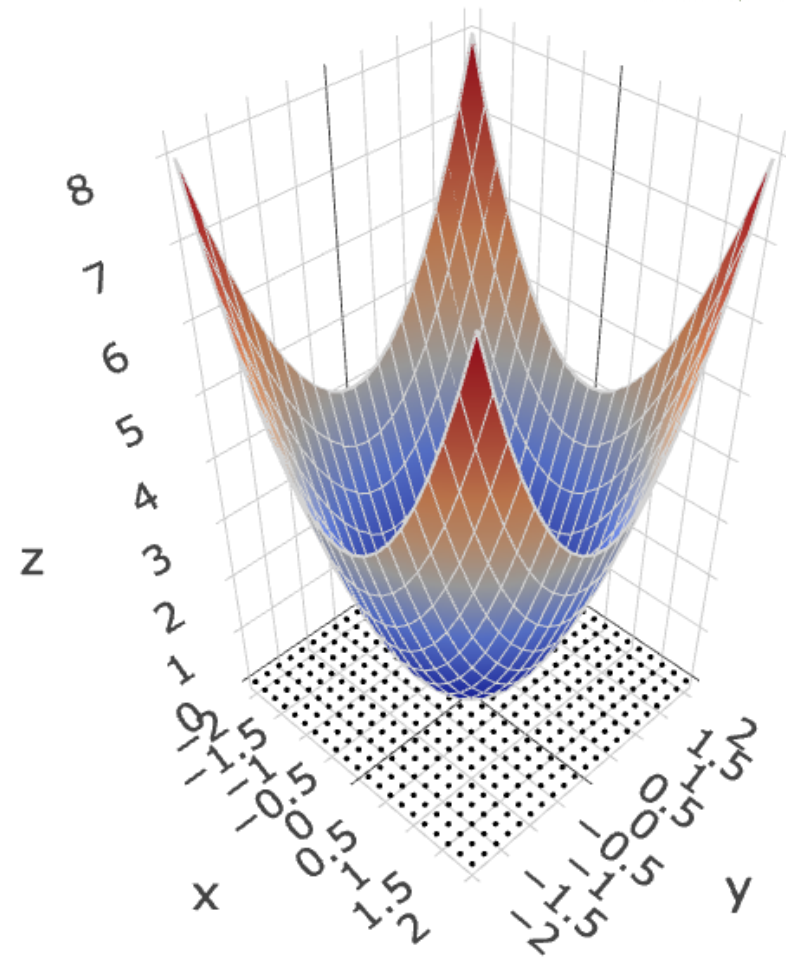
1	df.h	# df['h']
0	167	
1	183	
2	154	
Name: h, dtype: int64		
1	df.w	# df['w']
0	72	
1	67	
2	56	
Name: w, dtype: int64		



# 3次元の関数グラフ

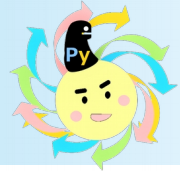


```
1 import numpy as np
2
3 # ① 独立変数の用意
4 x = np.linspace(-2, 2, 21)
5 y = np.arange(-2, 2.2, 0.2)
6 x_grd, y_grd = np.meshgrid(x, y)
7
8 # ② 関数の計算 と 従属変数 の出力
9 z_grd = x_grd ** 2 + y_grd ** 2
```

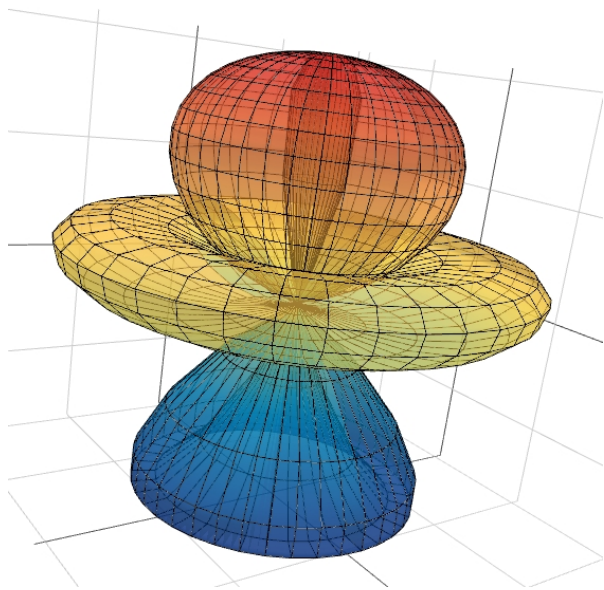




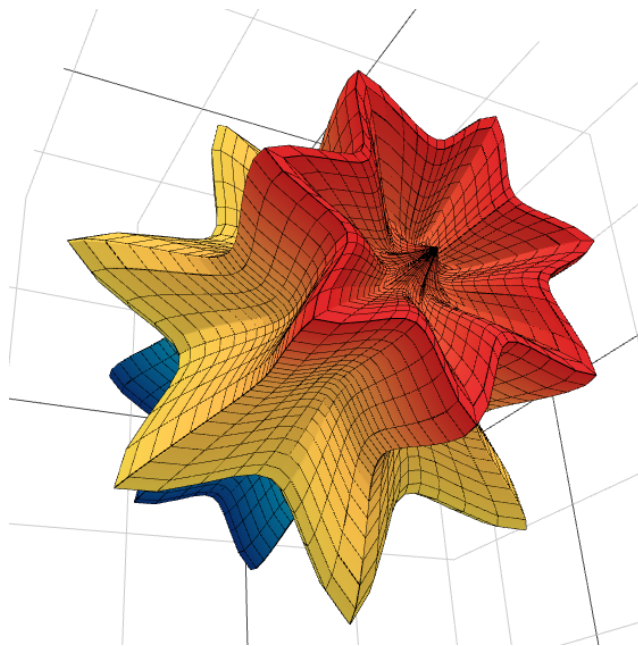
# その他の関数グラフ



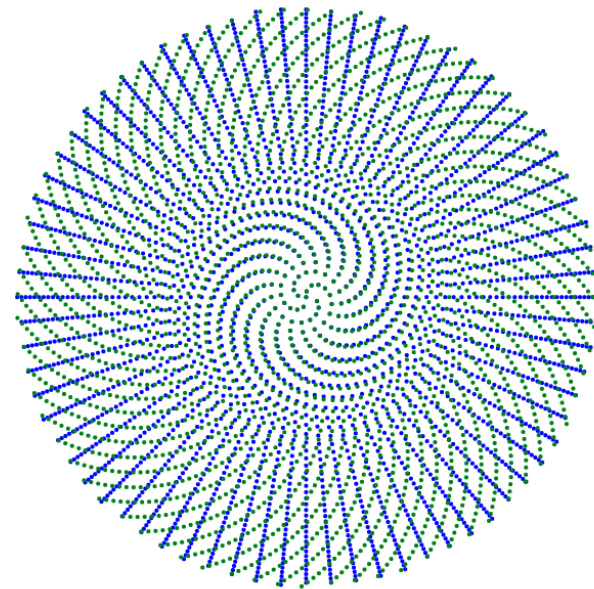
- 極座標

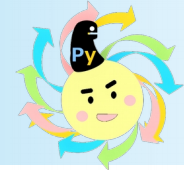


- Supershape



- フェルマー螺旋

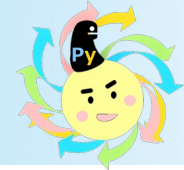




# numpy が 超便利！

# どんどん使おう！





# ご清聴ありがとうございました

本日の資料：

<https://github.com/malo21st/PyFukuoka191212>

