

「Python プログラミング基礎その2」

九州大学 大学院システム情報科学研究所
情報知能工学部門
データサイエンス実践特別講座
末廣大貴, Diego Thomas, 正井克俊

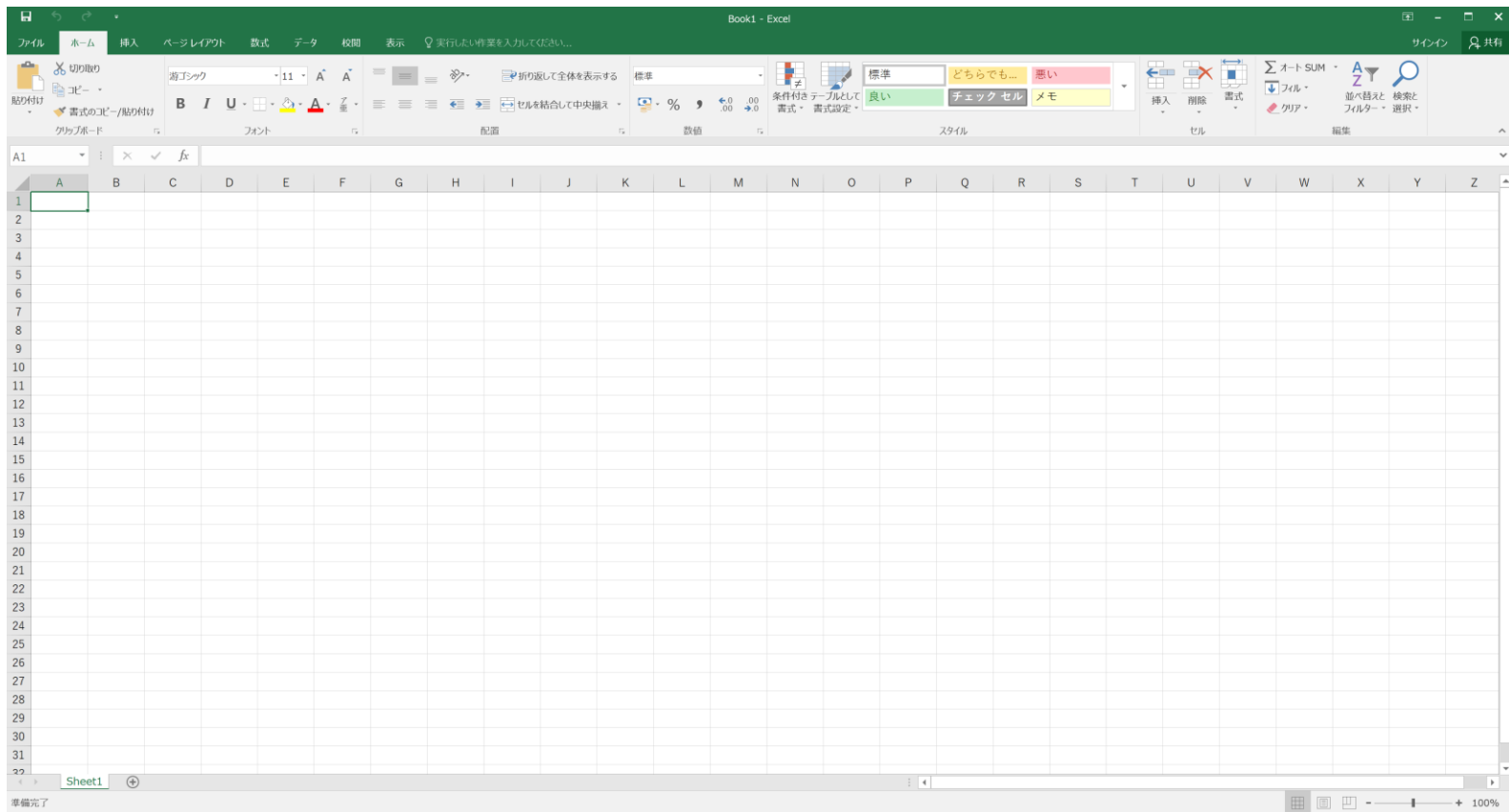
復習

CSV file format

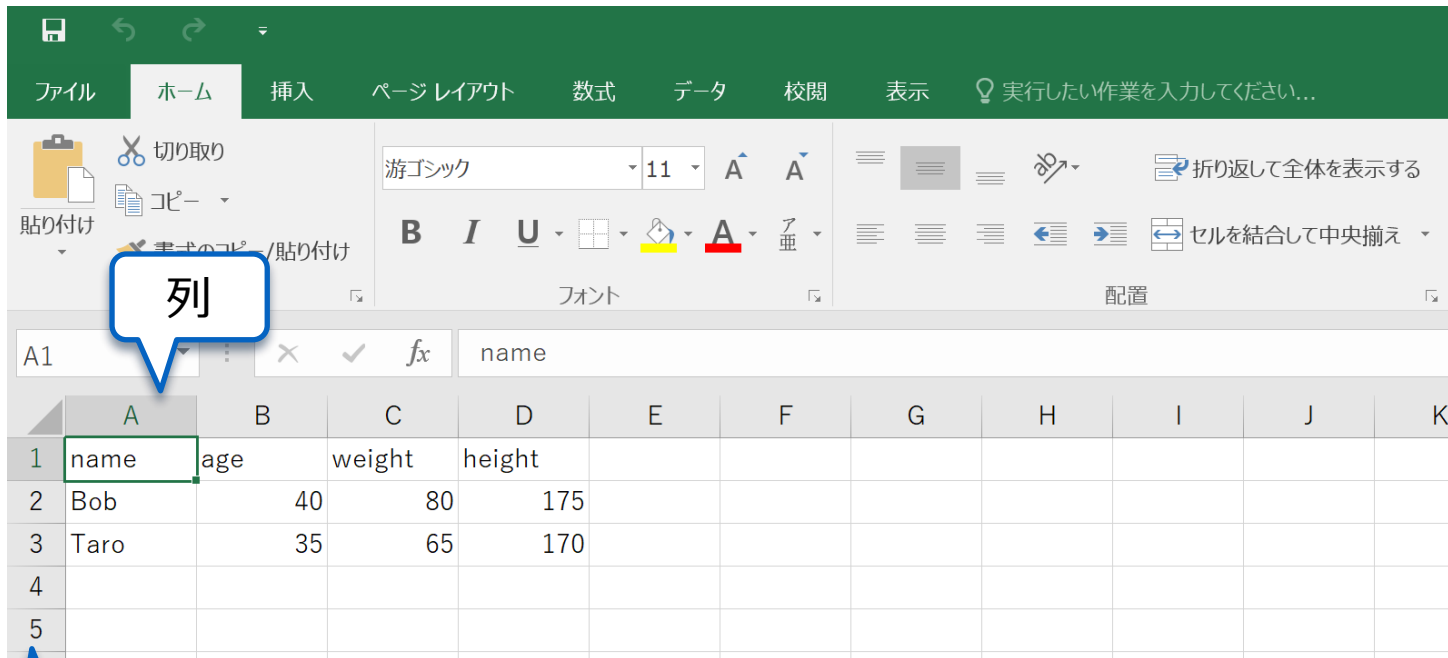
- CSV = comma separated values
- File format specifications:
 - 項目ごとにコンマ “,” で区切られたデータ
 - 1行が一まとめのデータで、改行で行を識別
- Example:
name, age, weight, height
Bob, 40, 80, 175
Taro, 35, 65, 170
. . .

Excel で csv ファイル

● Excel



Excel で csv ファイル



The screenshot shows the Microsoft Excel interface with a CSV file open. The ribbon is set to 'ホーム' (Home). The font settings are '游ゴシック' (Yu Gothic) at size 11. The data is as follows:

	A	B	C	D	E	F	G	H	I	J	K
1	name	age	weight	height							
2	Bob	40	80	175							
3	Taro	35	65	170							
4											
5											

列

行

Excel で csv ファイル

● CSV fileとして保存

ファイル → “名前を付けて保存”

開く
書き保存
名前を付けて保存
印刷
共有
エクスポート
発行
閉じる
アカウント
オプション

デスクトップ

ブックの保護

問題のチェック

ブックの管理

ブラウザーの表示オプション

フォルダの選択

ファイルフォーマットの選択

csv (カンマ区切り)

適当に名前をつけて終了

ExcelでCSVファイルを作ってみよう！

- Excelを開いて、次の内容を自分で書いてみよう！
- Bob, Taroに加えて、1行分、適当に自分でデータを書いて、CSVファイルとしてsaveしてみよう！

name, age, weight, height

Bob, 40, 80, 175

Taro, 35, 65, 170

(ここに好きなデータを記載)

Fileのアップロード

introduction.ipynb ☆

ファイル 編集 表示 挿入 ランタイム ツール

+ コード + テキスト

[12] 1
2
3
4
4
2
3
4

▶ # 外部データの読み込み

```
# pandas
import pandas as pd
data = pd.read_csv(data_dir+"some.csv")
# 好きな行を見る
print(data.iloc[0])
print(data.iloc[0:2])
# 好きな列を見る
print(data.iloc[:,0])
print(data.iloc[:,0:3])

print(data.columns)

# 項目を指定して見る
print(data.name)
print(data.height)

# 項目を指定して好きな行を見る
print(data.iloc[0].weight)
print(data.weight[1])

for i in range(data.shape[0]):
    print(data.iloc[i])

# 行を追加
s = pd.Series(['Jiro',30,75,180],index=data.index)
data = data.append(s,ignore_index=True)

# 新規列名を指定して追加
data['BMI'] = [20, 30, 25]
print(data)

# 行、列を指定して編集
data.iloc[0,4] = 40
print(data)
```

introduction.ipynb ☆

ファイル 編集 表示 挿入 ランタイム ツール ヘルプ

+ コード + テキスト

目次 コードスニペット ファイル X

📁 アップロード 🔄 更新 🗑️ ドライブをマウント

📁 ..

▶ 📁 sample_data

[12] 1
2
3
4
4
2
3
4

▶ # 外部データの読み込み

```
# pandas
import pandas as pd
data = pd.read_csv(data_dir+"some.csv")
# 好きな行を見る
print(data.iloc[0])
print(data.iloc[0:2])
# 好きな列を見る
print(data.iloc[:,0])
print(data.iloc[:,0:3])

print(data.columns)

# 項目を指定して見る
print(data.name)
print(data.height)

# 項目を指定して好きな行を見る
print(data.iloc[0].weight)
print(data.weight[1])

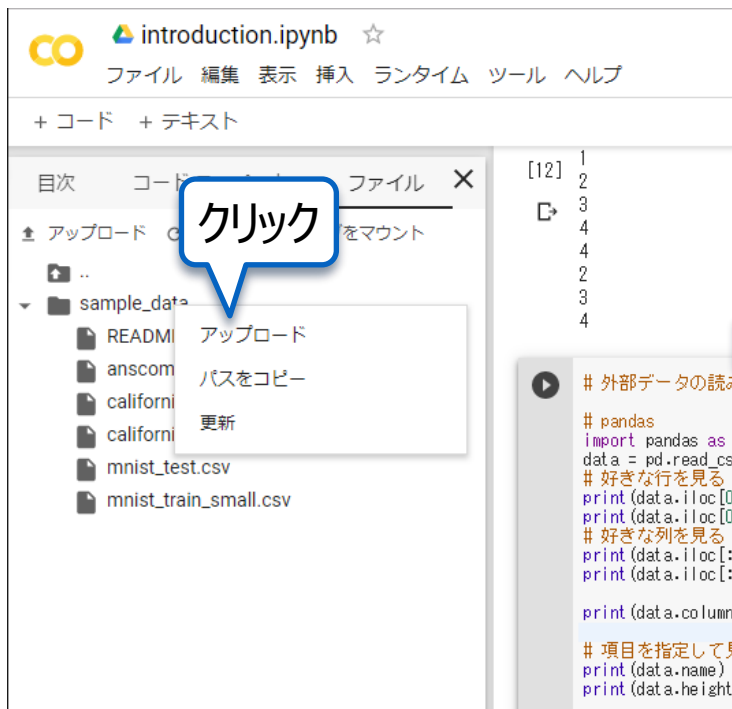
for i in range(data.shape[0]):
    print(data.iloc[i])

# 行を追加
s = pd.Series(['Jiro',30,75,180],index=data.index)
data = data.append(s,ignore_index=True)
```



右クリック

Fileのアップロード



introduction.ipynb ☆
ファイル 編集 表示 挿入 ランタイム ツール ヘルプ

+ コード + テキスト

目次 コード ファイル ×

アップロード をマウント

sample_data

- README アップロード
- anscom バスをコピー
- californi 更新
- californi
- mnist_test.csv
- mnist_train_small.csv

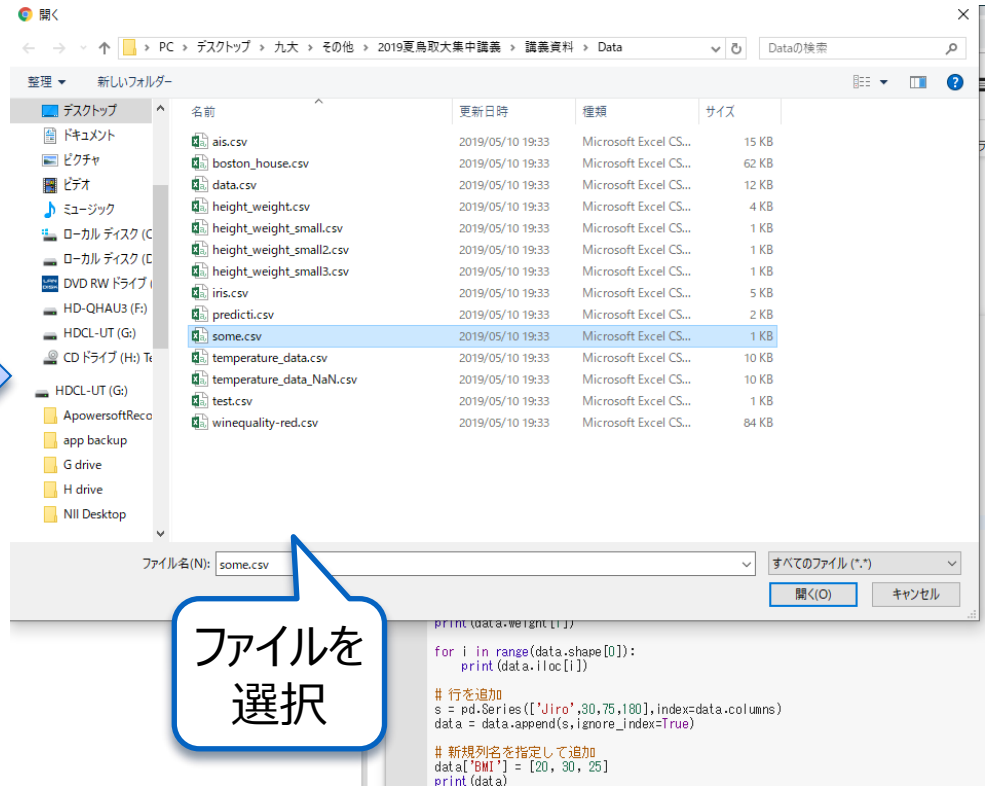
[12] 1
2
3
4
4
4
2
3
4

```
# 外部データの読み込み
# pandas
import pandas as pd
data = pd.read_csv('some.csv')
# 好きな行を見る
print(data.iloc[0])
# 好きな列を見る
print(data.iloc[:,0])
print(data.iloc[:,1])

print(data.columns)

# 項目を指定して見
print(data.name)
print(data.height)
```

クリック



開く

PC > デスクトップ > 九大 > その他 > 2019夏鳥取大集中講義 > 講義資料 > Data

名前	更新日時	種類	サイズ
ais.csv	2019/05/10 19:33	Microsoft Excel CS...	15 KB
boston_house.csv	2019/05/10 19:33	Microsoft Excel CS...	62 KB
data.csv	2019/05/10 19:33	Microsoft Excel CS...	12 KB
height_weight.csv	2019/05/10 19:33	Microsoft Excel CS...	4 KB
height_weight_small.csv	2019/05/10 19:33	Microsoft Excel CS...	1 KB
height_weight_small2.csv	2019/05/10 19:33	Microsoft Excel CS...	1 KB
height_weight_small3.csv	2019/05/10 19:33	Microsoft Excel CS...	1 KB
iris.csv	2019/05/10 19:33	Microsoft Excel CS...	5 KB
predicti.csv	2019/05/10 19:33	Microsoft Excel CS...	2 KB
some.csv	2019/05/10 19:33	Microsoft Excel CS...	1 KB
temperature_data.csv	2019/05/10 19:33	Microsoft Excel CS...	10 KB
temperature_data_NaN.csv	2019/05/10 19:33	Microsoft Excel CS...	10 KB
test.csv	2019/05/10 19:33	Microsoft Excel CS...	1 KB
winequality-red.csv	2019/05/10 19:33	Microsoft Excel CS...	84 KB

ファイル名(N): some.csv

すべてのファイル (*.*)

開く(O) キャンセル

```
print(data.weight[1])

for i in range(data.shape[0]):
    print(data.iloc[i])

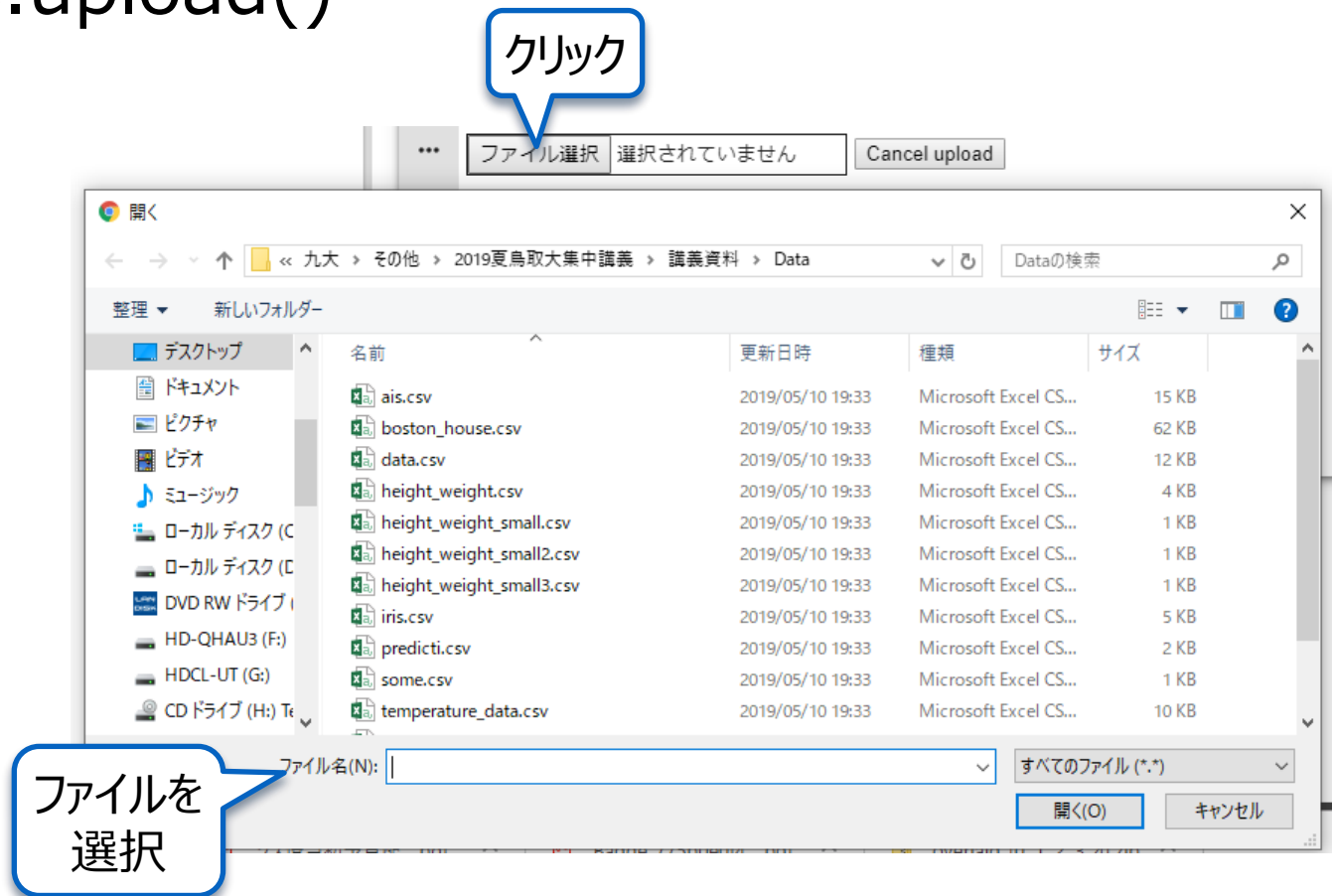
# 行を追加
s = pd.Series(['Jiro',30,75,180], index=data.columns)
data = data.append(s, ignore_index=True)

# 新規列名を指定して追加
data['BMI'] = [20, 30, 25]
print(data)
```

ファイルを選択

Fileのアップロード: コマンド

from google.colab import files
files.upload()



PythonによるCSVファイルの読み込み

- CSVファイル読み込み用の道具をPython上で呼び出す

```
import pandas as pd
```

CSVファイルの読み込み

● CSVファイルを読み出すおまじない

注) 他のフォルダにファイルがある場合
`pd.read_csv("path名¥ファイル名")`
Windowsでは、¥¥とバックスラッシュ2つ
でないと動かない場合もある。

CSVファイルの中身の
データが格納

```
Name, age, weight, height  
Bob, 40, 80, 175  
Taro, 35, 65, 170
```

読み込むCSVファイルのパス

```
data = pd.read_csv("some.csv")
```

読み込むCSVファイルの名前

以下を事前に行うことを忘れないように！

- (1) **`import pandas as pd`**
- (2) "some.csv" をコードと同じフォルダに置く

データの見方1

●好きな行を見る

```
In [399]: print(data.iloc[0])
name      Bob
age       40
weight    80
height   175
Name: 0, dtype: object
```

```
In [400]: print(data.iloc[0:2])
   name  age  weight  height
0  Bob   40     80    175
1  Taro  35     65    170
```

↑ 0,1行目を見る

●好きな列を見る

```
In [401]: print(data.iloc[:,0])
0      Bob
1      Taro
Name: name, dtype: object
```

```
In [402]: print(data.iloc[:,0:3])
   name  age  weight
0  Bob   40     80
1  Taro  35     65
```

↑ 0,1,2列目を見る

データの見方2

- 項目(index)を指定してみる

↓ 項目を指定

```
In [406]: print(data.name)
0      Bob
1      Taro
Name: name, dtype: object
```

```
In [407]: print(data.height)
0      175
1      170
Name: height, dtype: int64
```

- 項目を指定して好きな行を見る

どちらでもO.K.

```
In [125]: print(data.iloc[0].weight)
80
```

```
In [126]: print(data.weight[1])
65
```

行を指定して項目を指定

項目を指定して行を指定

データの見方2

- 項目(index)を指定してみる

↓ 項目を指定

```
In [406]: print(data.name)
0      Bob
1      Taro
Name: name, dtype: object
```

```
In [407]: print(data.height)
0      175
1      170
Name: height, dtype: int64
```

- 項目を指定して好きな行を見る

どちらでもO.K.

```
In [125]: print(data.iloc[0].weight)
80
```

```
In [126]: print(data.weight[1])
65
```

行を指定して項目を指定

項目を指定して行を指定

一個一個見てたらキリがない！

データの可視化（プロット）

- プロットには色々な方法があります
 - 折れ線グラフ
 - ヒストグラム
 - 散布図
 - etc.

はじめに

- 今回はサンプルとして次の2つのデータを使ってみます
- 那覇, 福岡, 札幌の気温データ
→ temperature_data.csv
<http://www.data.jma.go.jp/gmd/risk/obsdl/index.php>
- 体重身長データ (オーストラリア国立スポーツ研究所の男女202人のアスリートのデータ)
→ height_weight.csv
<http://www.statsci.org/data/oz/ais.html>

おまじないと準備

- import pandas as pd
- データの読み込み

```
In [66]: temper_data=pd.read_csv("../DS_enshu1/temperature_data.csv")
```

```
In [67]: HW_data=pd.read_csv("../DS_enshu1/height_weight.csv")
```



ファイルの場所は各自置いた場所を
指定しよう

読み込んだデータの中身

```
In [68]: temper_data
Out[68]:
```

```
   date  Naha  Fukuoka  Sapporo
0  2016/1/1  18.1    7.3    -1.1
1  2016/1/2  20.2   11.6    1.6
2  2016/1/3  21.2   11.6    0.3
3  2016/1/4  19.8   11.0   -1.7
4  2016/1/5  22.5    9.9   -3.9
5  2016/1/6  19.5    9.5   -2.3
6  2016/1/7  19.1    9.1   -2.7
7  2016/1/8  16.6    7.5   -2.5
8  2016/1/9  17.1    7.9   -3.6
9  2016/1/10 18.3    7.9   -4.2
10 2016/1/11 19.4    8.3   -6.7
11 2016/1/12 18.2    7.5   -6.4
12 2016/1/13 15.9    5.8   -4.2
13 2016/1/14 15.1    6.7   -3.7
14 2016/1/15 16.6    5.8   -6.3
15 2016/1/16 17.5    7.5   -5.0
16 2016/1/17 19.5    7.1   -5.1
17 2016/1/18 17.6    6.7   -5.9
18 2016/1/19 15.3    2.3   -1.4
19 2016/1/20 15.0    2.7   -0.5
20 2016/1/21 17.3    2.8   -2.1
21 2016/1/22 16.8    5.8   -2.7
22 2016/1/23 14.7    3.5   -3.7
23 2016/1/24  9.2   -2.0   -5.7
24 2016/1/25 10.4    1.2   -6.7
25 2016/1/26 12.6    4.4    0.0
26 2016/1/27 15.7    6.6   -1.7
27 2016/1/28 20.2    8.3   -3.9
28 2016/1/29 22.2   11.0   -5.0
29 2016/1/30 18.6   10.8   -5.8
...
336 2016/12/2 21.5   11.6    1.1
337 2016/12/3 22.5   10.6    3.6
338 2016/12/4 24.1   11.3    7.1
339 2016/12/5 23.3   12.4    6.3
340 2016/12/6 21.3   11.3   -2.1
341 2016/12/7 19.8    8.8   -5.0
342 2016/12/8 20.7   10.6   -2.5
343 2016/12/9 21.6   12.4   -0.3
344 2016/12/10 20.9   10.6   -2.6
345 2016/12/11 20.3    9.4   -4.6
346 2016/12/12 21.9   11.0   -4.1
347 2016/12/13 22.8   12.3    0.2
348 2016/12/14 20.0   10.8   -2.0
349 2016/12/15 17.4    7.9   -4.5
350 2016/12/16 16.6    6.1   -5.3
351 2016/12/17 17.8    7.4   -3.6
352 2016/12/18 20.7    8.9    1.0
353 2016/12/19 22.2   12.4    1.0
354 2016/12/20 23.4   14.0   -0.1
355 2016/12/21 23.6   16.1    2.7
356 2016/12/22 22.7   15.8    0.3
357 2016/12/23 18.9    9.7   -1.4
358 2016/12/24 18.5    9.1   -4.4
359 2016/12/25 20.2   10.8   -2.3
360 2016/12/26 22.3   13.3    1.3
361 2016/12/27 18.5   10.7   -2.5
362 2016/12/28 16.9    6.8   -6.4
363 2016/12/29 17.9    6.8   -3.6
364 2016/12/30 17.9    6.6   -2.8
365 2016/12/31 19.3    7.3   -2.6
```

[366 rows x 4 columns]

```
In [70]: HW_data
Out[70]:
```

```
  Unnamed: 0  Sex  Ht  Wt
0           1    1  195.9  78.9
1           2    1  189.7  74.4
2           3    1  177.8  69.1
3           4    1  185.0  74.9
4           5    1  184.6  64.6
5           6    1  174.0  63.7
6           7    1  186.2  75.2
7           8    1  173.8  62.3
8           9    1  171.4  66.5
9          10    1  179.9  62.9
10         11    1  193.4  96.3
11         12    1  188.7  75.5
12         13    1  169.1  63.0
13         14    1  177.9  80.5
14         15    1  177.5  71.3
15         16    1  179.6  70.5
16         17    1  181.3  73.2
17         18    1  179.7  68.7
18         19    1  185.2  80.5
19         20    1  177.3  72.9
20         21    1  179.3  74.5
21         22    1  175.3  75.4
22         23    1  174.0  69.5
23         24    1  183.3  66.4
24         25    1  184.7  79.7
25         26    1  180.2  73.6
26         27    1  180.2  78.7
27         28    1  176.0  75.0
28         29    1  156.0  49.8
29         30    1  179.7  67.2
...
173        173    0  178.5  71.0
174        174    0  171.3  69.1
175        175    0  178.0  62.9
176        176    0  189.1  94.8
177        177    0  195.4  94.6
178        178    0  179.1  108.2
179        179    0  180.1  97.9
180        180    0  179.6  75.2
181        181    0  174.7  74.8
182        182    0  192.7  94.2
183        183    0  179.3  76.1
184        184    0  197.5  94.7
185        185    0  182.7  86.2
186        186    0  190.5  79.6
187        187    0  191.0  85.3
188        188    0  179.6  74.4
189        189    0  192.6  93.5
190        190    0  194.1  87.6
191        191    0  193.0  85.4
192        192    0  193.9  101.0
193        193    0  187.7  74.9
194        194    0  185.3  87.3
195        195    0  191.5  90.0
196        196    0  184.6  94.7
197        197    0  179.9  76.3
198        198    0  183.9  93.2
199        199    0  183.5  80.0
200        200    0  183.1  73.8
201        201    0  178.4  71.1
202        202    0  190.8  76.7
```

[202 rows x 4 columns]

- 「データフレーム」という形式で保存
- 行列（縦が行，横が列）
- 気温データは
 - 縦が時間
 - 横が地方
- 身長体重データは
 - 縦が人（サンプル）
 - 横が項目（性別，身長，体重）

データの見方

●好きな行をみる

```
In [79]: temper_data.iloc[0]
Out[79]:
date      2016/1/1
Naha      18.1
Fukuoka   7.3
Sapporo   -1.1
Name: 0, dtype: object
```

```
In [82]: temper_data.iloc[:3]
Out[82]:
      date  Naha  Fukuoka  Sapporo
0  2016/1/1  18.1     7.3    -1.1
1  2016/1/2  20.2    11.6     1.6
2  2016/1/3  21.2    11.6     0.3
```

●好きな列をみる

↑ 0,1,2行目をみる

```
In [80]: temper_data.iloc[:,0]
```

```
In [84]: temper_data.iloc[:,1:3]
```

↑ 1,2列目をみる

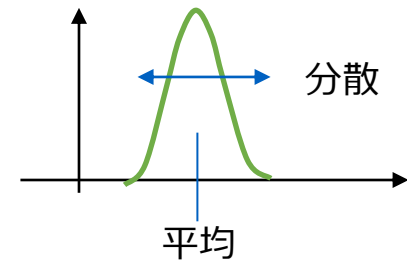
簡単な統計値を出す

● 平均値

```
In [94]: temper_data.mean()  
Out[94]:  
Naha      24.131421  
Fukuoka   18.091803  
Sapporo   9.295902  
dtype: float64
```

● 分散

```
In [95]: temper_data.var()  
Out[95]:  
Naha      24.559640  
Fukuoka   62.146179  
Sapporo   95.555079  
dtype: float64
```



● 標準偏差

```
In [96]: temper_data.std()  
Out[96]:  
Naha      4.955768  
Fukuoka   7.883285  
Sapporo   9.775228  
dtype: float64
```

まとめてみることもできます

- 統計量のまとめを表示

```
temper_data.describe()
```

	Naha	Fukuoka	Sapporo
count	366.000000	366.000000	366.000000
mean	24.131421	18.091803	9.295902
std	4.955768	7.883285	9.775228
min	9.200000	-2.000000	-6.800000
25%	20.225000	11.025000	0.300000
50%	25.050000	18.500000	8.650000
75%	28.600000	24.650000	18.050000
max	30.800000	31.400000	27.100000

データの25%は～以下
データの50%は～以下
データの75%は～以下（上位25%は～以上）

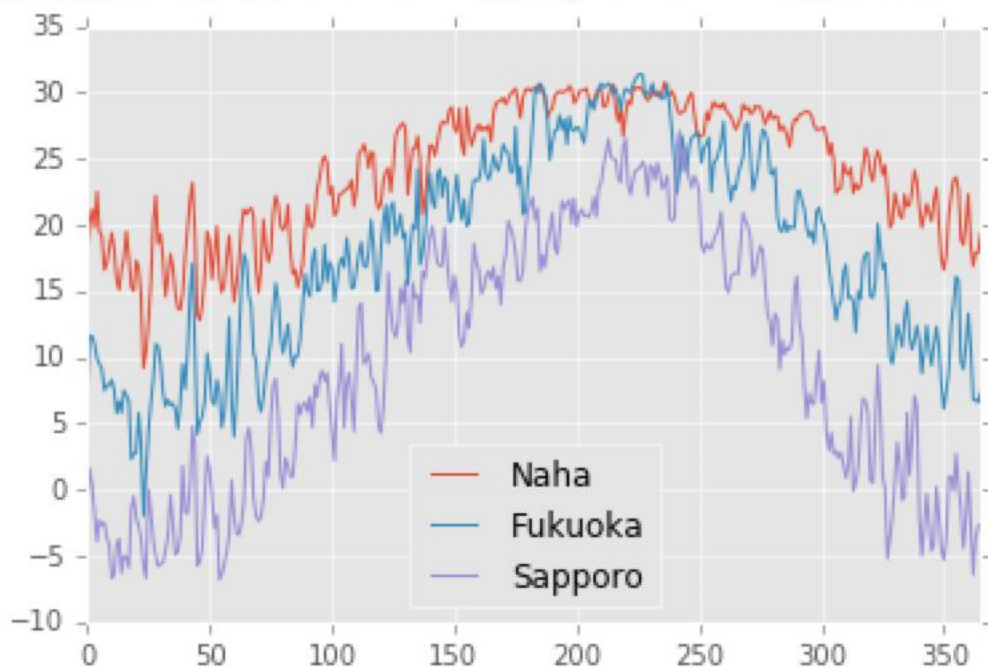
データの可視化（プロット）

折れ線グラフ

●折れ線グラフ

```
In [97]: temper_data.plot()
```

```
Out[97]: <matplotlib.axes._subplots.AxesSubplot at 0x120dc2050>
```



グラフの横軸が行の番号 (index) でわかりにくい

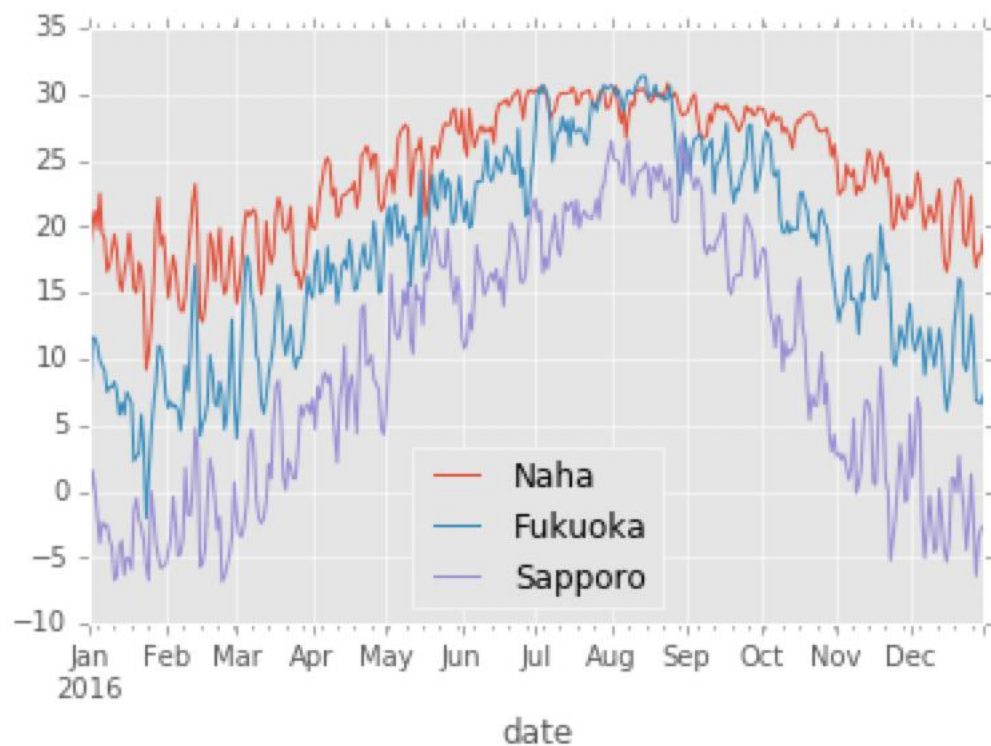
折れ線グラフ

●折れ線グラフ

```
In [98]: temper_data.index = pd.to_datetime(temper_data.date)
```

```
In [99]: temper_data.plot()
```

```
Out[99]: <matplotlib.axes._subplots.AxesSubplot at 0x120d95910>
```



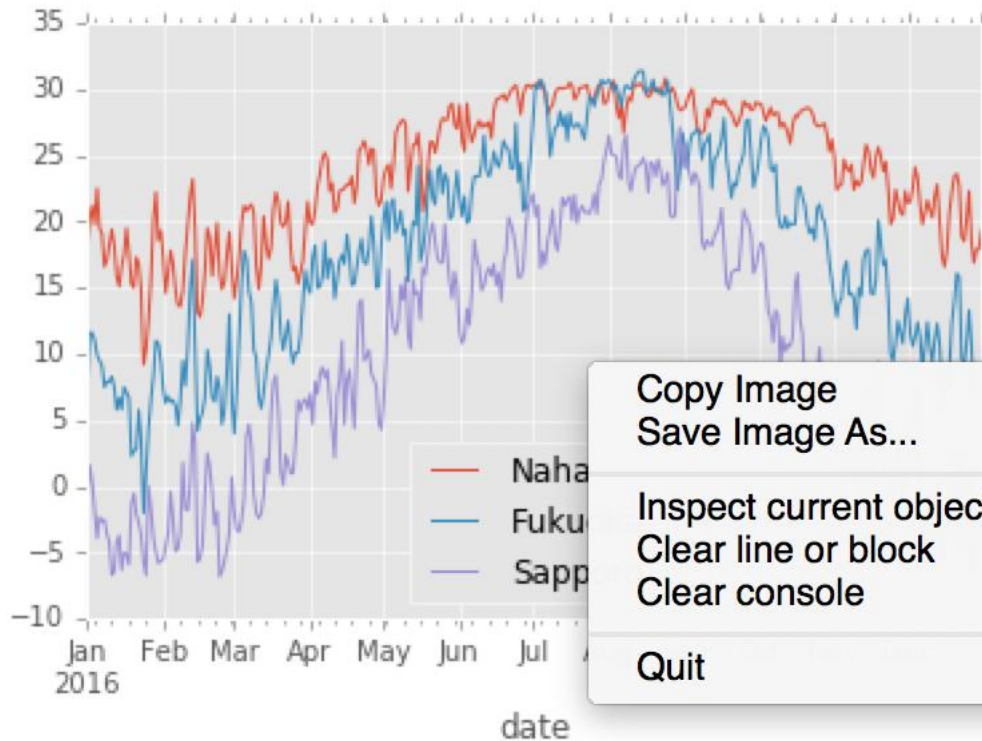
indexを date に変更

折れ線グラフ

●折れ線グラフ

```
In [99]: temper_data.plot()
```

```
Out [99]: <matplotlib.axes._subplots.AxesSubplot at 0x120d95910>
```

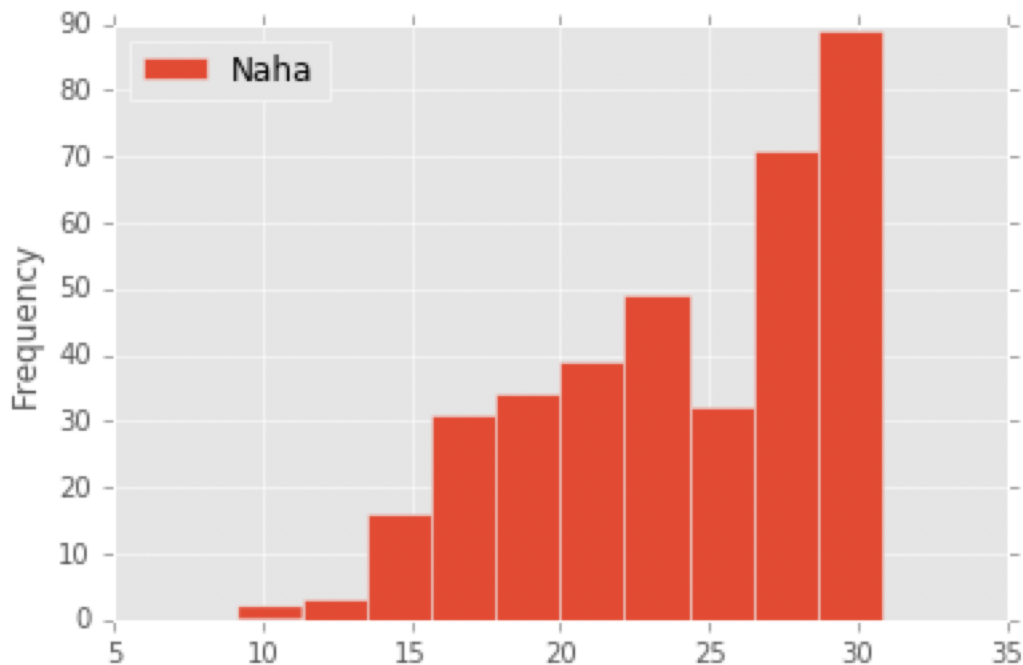


右クリックすれば
保存もできます

ヒストグラム

- ヒストグラム
 - 那覇の気温の分布をしてみる

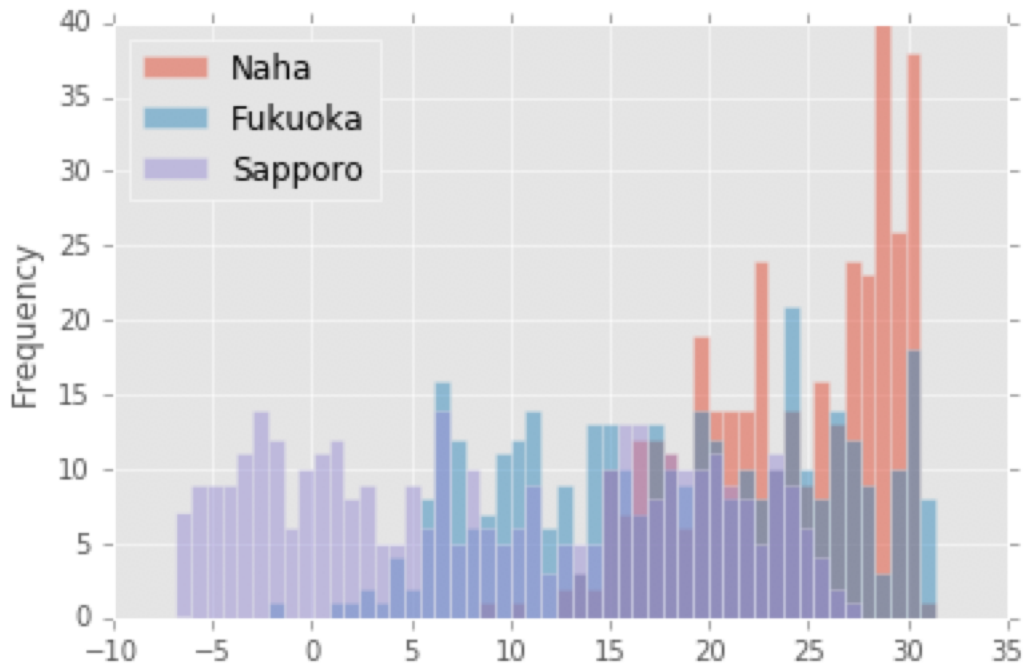
```
In [109]: temper_data.plot(y='Naha',kind='hist')  
Out [109]: <matplotlib.axes._subplots.AxesSubplot at 0x1281584d0>
```



ヒストグラム（発展）

- 那覇，福岡，札幌の分布を比較したい

```
In [115]: temper_data.plot.hist(bins=50, alpha=0.5)
Out[115]: <matplotlib.axes._subplots.AxesSubplot at 0x128cdec0>
```



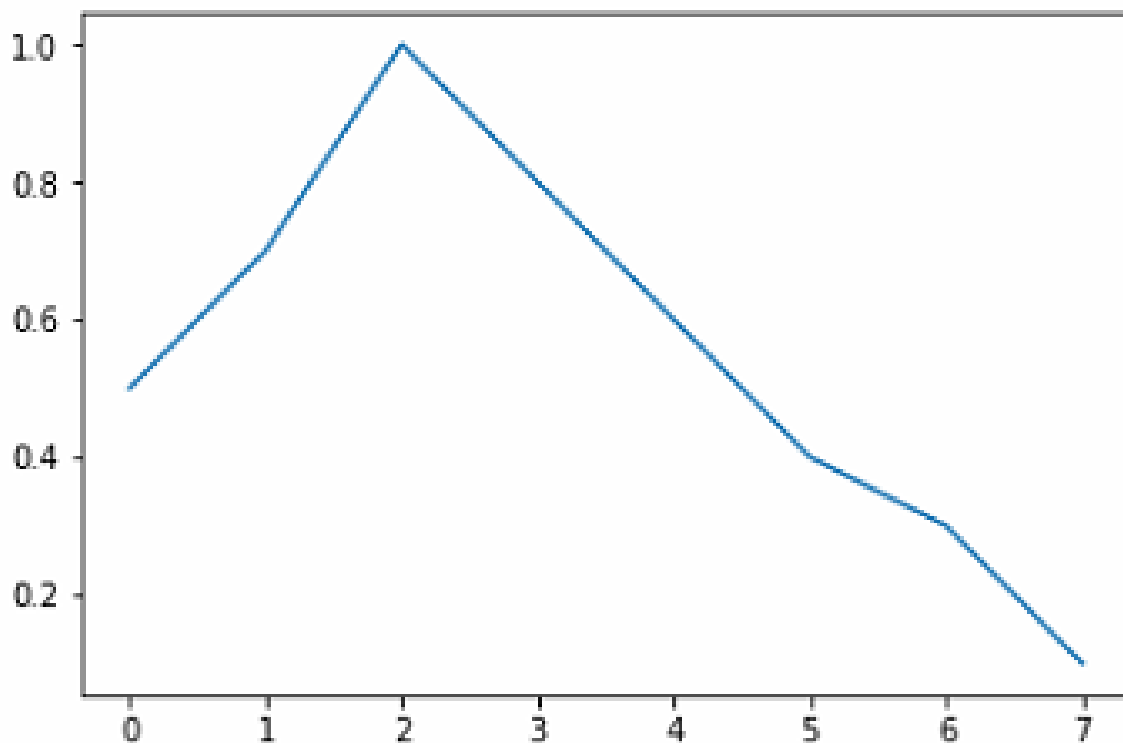
`data.plot(kind='hist')`
`data.plot.hist()`
は同じ処理

binsは棒の数， alpha は透明度

Matplotlibを使った可視化

- `import matplotlib.pyplot as plt`

```
In [12]: a = [0.5,0.7,1,0.8,0.6,0.4,0.3,0.1]
...: plt.plot(a)
Out[12]: [<matplotlib.lines.Line2D at 0x1d8b0bc9ba8>]
```



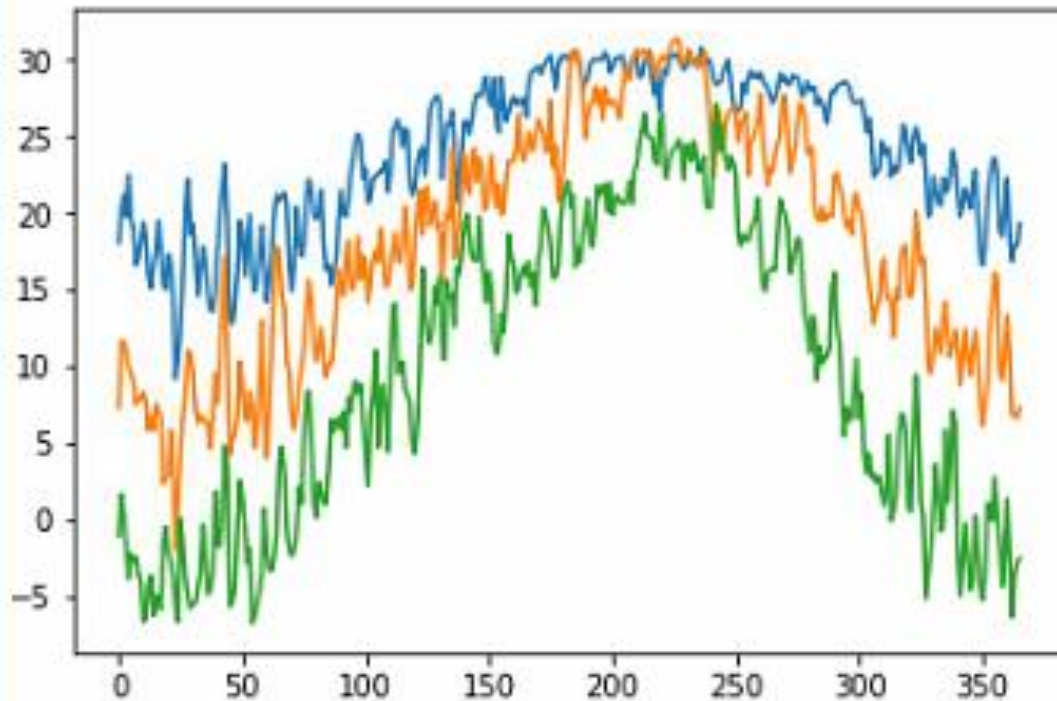
リストもしくは
numpyのarray型
を入力

pandas以外でも
使えます。
csvデータは基本的に
pandas の plotでも十分
ですが、より高度な
可視化（自由な可視化）
をしたいときには便利
（後ほど紹介）

Numpy array への変換

```
In [34]: Naha = temper_data.values[:,1]
...: Fukuoka = temper_data.values[:,2]
...: Sapporo = temper_data.values[:,3]
...:
...: plt.plot(Naha)
...: plt.plot(Fukuoka)
...: plt.plot(Sapporo)
Out[34]: [<matplotlib.lines.Line2D at 0x1d8b0b14d30>]
```

Pandas の values を使うと
numpy の array型として
抽出できる



練習 1 : データの可視化

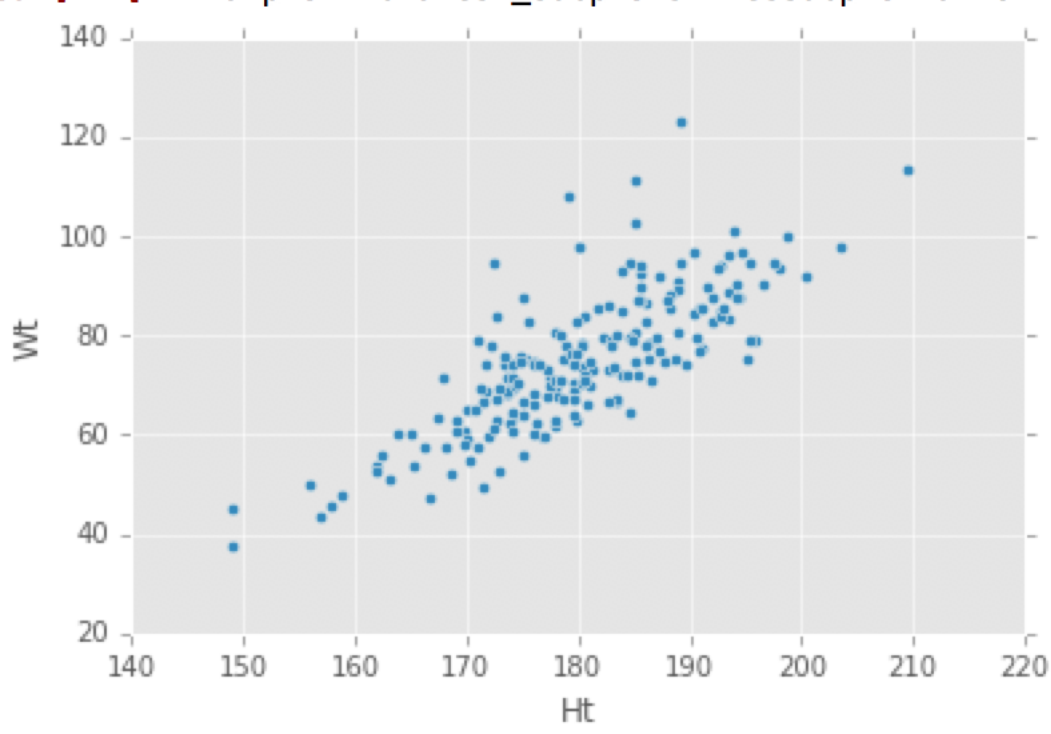
- 他の地方の分布も見てみよう
 - 福岡、札幌、那覇それぞれのヒストグラムを作成
- bins と alpha を変えて 3 地点の分布を比較してみよう
- 身長体重データでの統計値も見てみよう

散布図

- 身長と体重はどんな関係がある？

```
In [111]: HW_data.plot(x='Ht', y='Wt', kind='scatter')
```

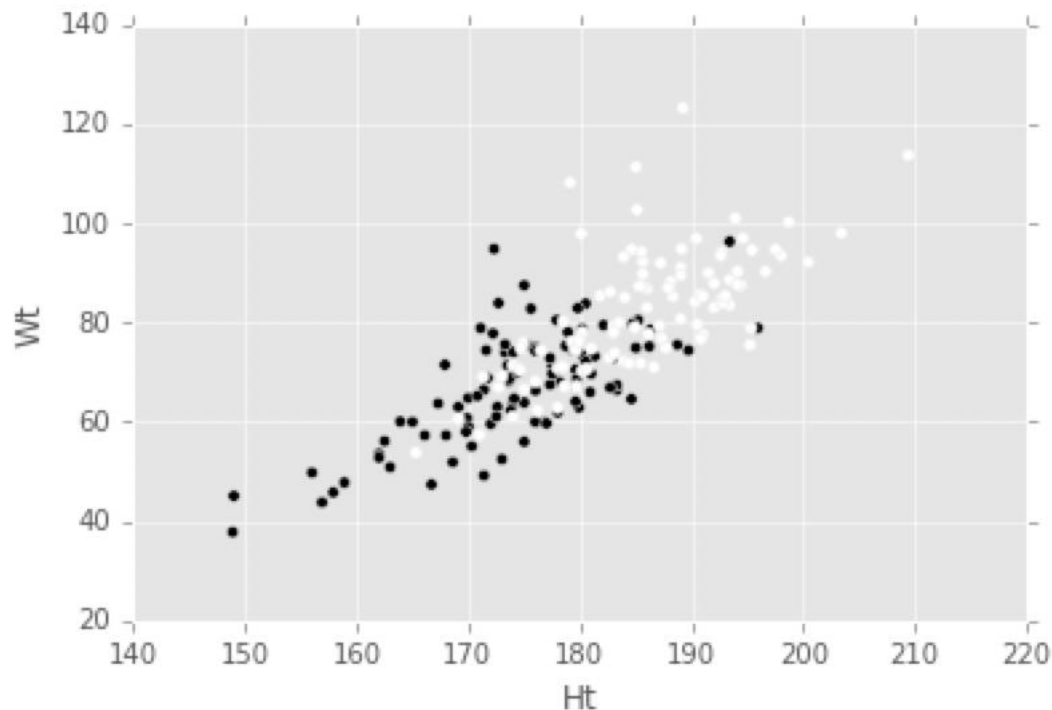
```
Out[111]: <matplotlib.axes._subplots.AxesSubplot at 0x12855f310>
```



散布図

- 身長と体重はどんな関係があるか？
- 性別でどんな違いがあるか？

```
In [122]: HW_data.plot(kind='scatter',x='Ht',y='Wt',c=HW_data.Sex)  
Out[122]: <matplotlib.axes._subplots.AxesSubplot at 0x12be06d10>
```



項目同士の関係性を見るのに便利なプロット

- 項目がたくさんあるとき、まとめて関係をプロット

```
import pandas as pd
wine_data = pd.read_csv("../winequality-red.csv", sep=";")
import seaborn as sns
sns.pairplot(wine_data)
```

- 実際に実行してみましよう
 - 図は右クリックで保存できます



欠損値について

- csv の中に空白（欠損）があると、NaN（Not a Number）というものが入る
- サンプルファイルを読み込んでみてみよう

```
In [92]: temper_data2=pd.read_csv("../DS_enshu1/temperature_data_NaN.csv")
```

```
In [93]: temper_data2
```

欠損値について

- csv の中に空白（欠損）があると、NaN（Not a Number）というものが入る
- 欠損値のあるデータ（行）を取り除く

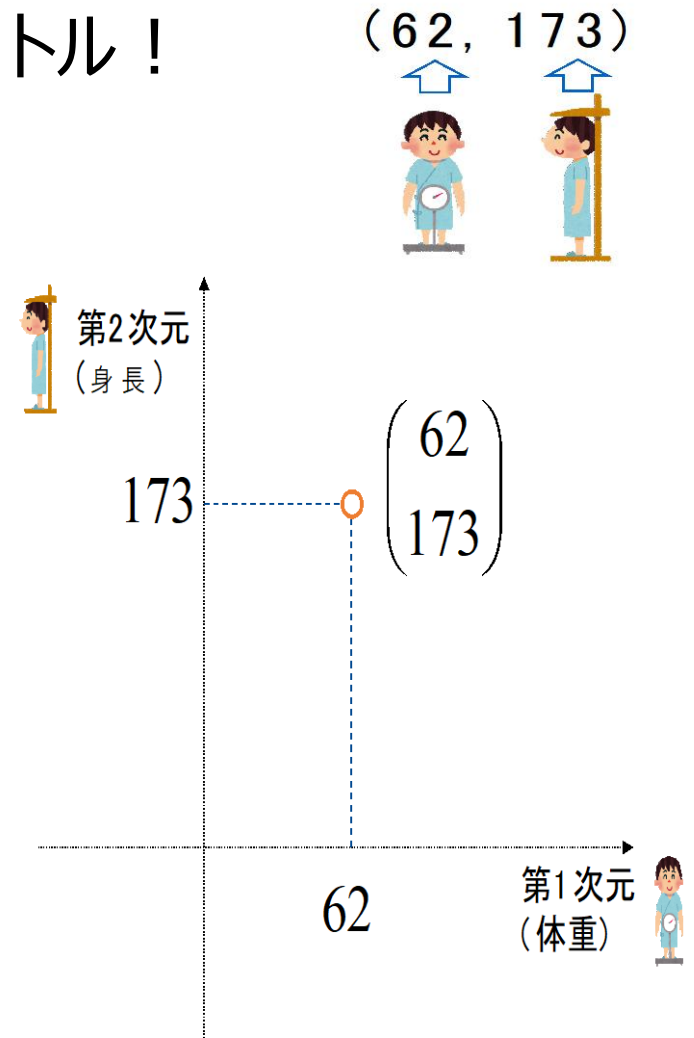
```
In [130]: temper_data2.dropna(0)
Out[130]:
```

	date	Naha	Fukuoka	Sapporo
0	2016/1/1	18.1	7.3	-1.1
1	2016/1/2	20.2	11.6	1.6
2	2016/1/3	21.2	11.6	0.3
3	2016/1/4	19.8	11.0	-1.7
4	2016/1/5	22.5	9.9	-3.9
5	2016/1/6	19.5	9.5	-2.3
6	2016/1/7	19.1	9.1	-2.7
7	2016/1/8	16.6	7.5	-2.5
8	2016/1/9	17.1	7.9	-3.6
9	2016/1/10	18.3	7.9	-4.2
10	2016/1/11	19.4	8.3	-6.7
11	2016/1/12	18.2	7.5	-6.4
12	2016/1/13	15.9	5.8	-4.2
13	2016/1/14	15.1	6.7	-3.7
15	2016/1/16	17.5	7.5	-5.0

ベクトル表現と ベクトル演算の関数化

ベクトル

- データの多くは数字の組, つまりベクトル!
 - 例: $x = [62, 173]$
- ベクトルで表現すると,
 - データ間の「距離」
 - 類似度
 - などが計算できました



ベクトルと python

- np.array を使うと色々な演算が楽
- まず「import numpy as np」を実行
- 例 :

```
a=np.array([0,1,2])
```

```
print(a)
```

```
[0 1 2]
```

ベクトルの演算:和

- ベクトルの和 (要素同士の和を取る)

$$\begin{array}{c} \mathbf{x} \\ \downarrow \\ \begin{array}{|c|} \hline x_1 \\ \hline \end{array} \\ \begin{array}{|c|} \hline x_2 \\ \hline \end{array} \end{array} + \begin{array}{c} \mathbf{y} \\ \downarrow \\ \begin{array}{|c|} \hline y_1 \\ \hline \end{array} \\ \begin{array}{|c|} \hline y_2 \\ \hline \end{array} \end{array} = \begin{array}{c} (x_1 + y_1) \\ (x_2 + y_2) \end{array}$$

```
a=np.array([1,2,3,4,5])  
b=np.array([2,2,3,3,4])  
print(a+b)
```

ベクトルの演算: 差

- ベクトルの差 (要素同士の差を取る)

$$\begin{array}{c} \mathbf{x} \\ \downarrow \\ \begin{array}{|c|} \hline x_1 \\ \hline \end{array} \\ \begin{array}{|c|} \hline x_2 \\ \hline \end{array} \end{array} - \begin{array}{c} \mathbf{y} \\ \downarrow \\ \begin{array}{|c|} \hline y_1 \\ \hline \end{array} \\ \begin{array}{|c|} \hline y_2 \\ \hline \end{array} \end{array} = \begin{array}{c} x_1 - y_1 \\ x_2 - y_2 \end{array}$$

```
a=np.array([1,2,3,4,5])  
b=np.array([2,2,3,3,4])  
print(a-b)
```

ベクトルと数値の積

- ベクトルと数値の積

$$s \times \begin{pmatrix} x \\ x_1 \\ x_2 \end{pmatrix} \equiv \begin{pmatrix} s \times x \\ s \times x_1 \\ s \times x_2 \end{pmatrix}$$

```
a=np.array([1,2,3,4,5])
```

```
print(2*a)
```

```
[ 2  4  6  8 10]
```

練習 2 : 2つのベクトル間の和と差を 「プログラムで」計算させよう

$$\mathbf{x} = \begin{pmatrix} 3 \\ 5 \end{pmatrix}, \mathbf{y} = \begin{pmatrix} 6 \\ 1 \end{pmatrix} \text{のとき } \mathbf{x} + \mathbf{y}, \mathbf{x} - \mathbf{y} \text{は?}$$

$$\mathbf{x} = \begin{pmatrix} 3 \\ 5 \\ 2 \end{pmatrix}, \mathbf{y} = \begin{pmatrix} 6 \\ 1 \\ 2 \end{pmatrix} \text{のとき } \mathbf{x} + \mathbf{y}, \mathbf{x} - \mathbf{y} \text{は?}$$

データ「セット」

- 複数のベクトルを行列で表現

```
a=np.array([1,2,3,4,5])  
b=np.array([2,2,3,3,4])  
c=np.array([5,4,2,3,3])
```

```
d=np.array([a,b,c])
```

```
In [474]: d
```

```
Out[474]:
```

```
array([[1, 2, 3, 4, 5],  
       [2, 2, 3, 3, 4],  
       [5, 4, 2, 3, 3]])
```

演習 1

- 那覇, 福岡, 札幌の気温データを読み込んで
→ `temperature_data.csv`
- 各都市の365日分の気温を一つのベクトル (`numpy array`型) として定義し、ベクトル間(那覇-福岡、福岡-札幌、札幌-那覇)の和と差をそれぞれ求めよう (引き算の順序は問わない) 。
- 求めた差のベクトルを折れ線グラフやヒストグラムで可視化してみよう。
- ※) `matplotlib.pyplot` を利用する