プログラミングゼミ
python②

2017/ 5/ 2

福田研究室　修士2年　金子法子

今日の内容

✿データ分析の一般的な手順

1. データを読み込む
2. データを分析する
3. 分析したデータを可視化する

今日の目標
これらの手順の基礎的な操作を学ぶ！！

ライブラリ

前回インストールしてもらったPythonだけでは，csvの読み書きや統計的計算など
できないことがたくさんある…

✿ライブラリーのインポート

汎用性の高い複数のプログラムを再利用可能な形でひとまとまりにしたもの

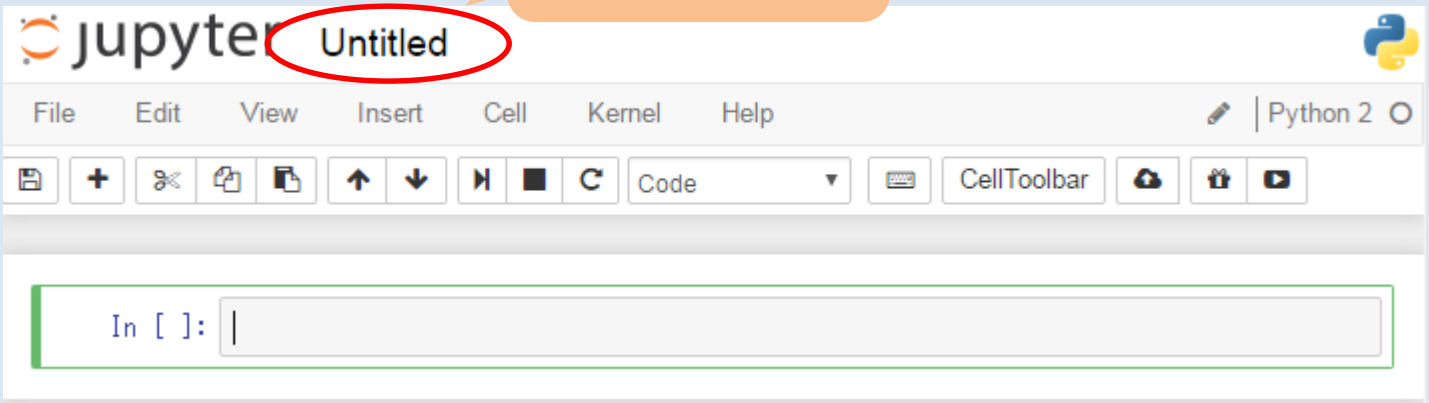
✿本日使うライブラリ

- numpy
  ベクトルや行列計算を使うためのライブラリ
- pandas
  DataFrameを提供するライブラリ
- csv
  csv読み書きのためのライブラリ
- matplotlib
  グラフ描画のためのライブラリ

Csvの読み込み

今回使うデータ:
配布済みの“kanko_matsuyama.csv”, “kanko_castle.csv”

✿ディレクトリの設定



Csvの読み込み

今回使うデータ:
配布済みの“kanko_matsuyama.csv”, “kanko_castle.csv”

✿ライブラリーのインポート

```
In [2]: import numpy
        import pandas
        import csv
        import matplotlib
        import matplotlib.pyplot as plt
```

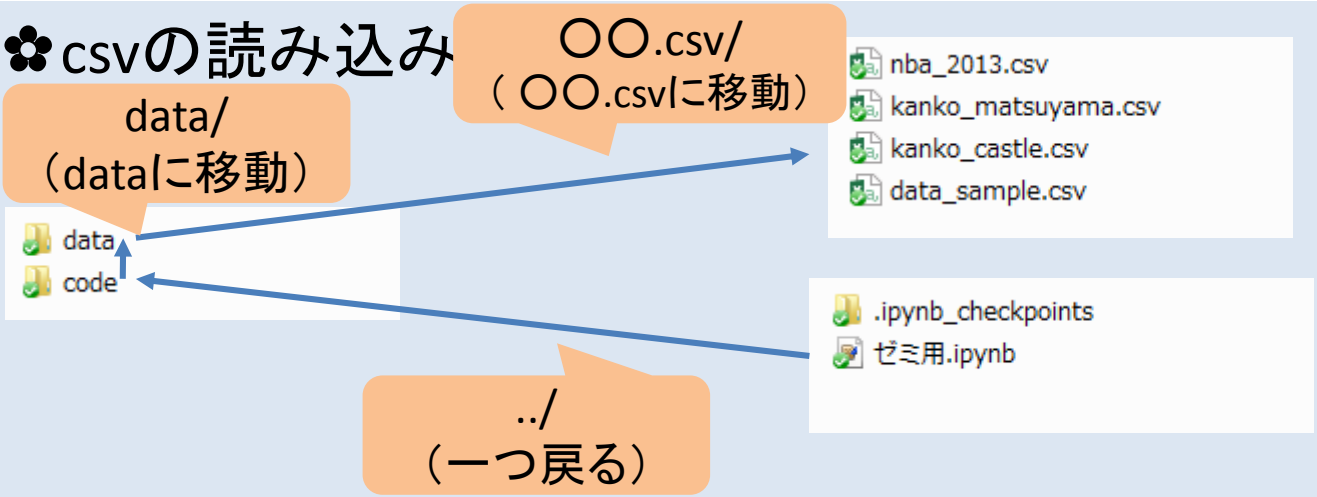
長くて面倒なときは省略名定義

Code
import [ライブラリー名]

Csvの読み込み

今回使うデータ:
配布済みの“kanko_matsuyama.csv”, “kanko_castle.csv”

✿csvの読み込み




```
In [63]: matsuyama = pandas.read_csv("../data/kanko_matsuyama.csv")
         castle    = pandas.read_csv("../data/kanko_castle.csv")
```

Code
名前　＝　pandas.read_csv(“××/〇〇.csv”)

numpy配列とpandas データフレーム

- Numpy のndarray

多次元配列を扱うクラスのこと
メリット：ndarrayは要素の型や要素数を揃えて固定してしまうことで行列演算用の関数が使えて、計算を楽にすばやくできる

0行目 1行目

```
In [73]: #numpy の作り方
         numpy1 = np.array([[1,5,4],[2,5,6]])

In [74]: print(numpy1)
         [[1 5 4]
          [2 5 6]]
```

$$\begin{array}{c} \\ \text{0行目} \\ \text{1行目} \end{array} \begin{array}{c} \begin{array}{ccc} \text{0列目} & \text{1列目} & \text{2列目} \end{array} \\ \begin{pmatrix} 1 & 5 & 4 \\ 2 & 5 & 6 \end{pmatrix} \end{array}$$

```
In [75]: numpy1[0]
Out[75]: array([1, 5, 4])
```

0行目

```
In [77]: numpy1[:,1]
Out[77]: array([5, 5])
```

1列目

```
In [79]: numpy1[0,2]
Out[79]: 4
```

0行目2列目の要素

数え方が特殊なので注意！！

numpy配列とpandas データフレーム

- Pandas の DataFrame

  DataFrameはカラムごとに異なる型の値を持つことができる2次元のラベル付けされたデータ配列のこと

  メリット：csvをそのまま読み取ることができ，簡単な操作がしやすい

```
In [92]: df1 = pd.DataFrame(
             {'名前': ['朝倉先生', '屋井先生', '室町先生', '花岡先生', '福田先生' ],
             '研究室': ['緑ヶ丘','すずかけ台', 'すずかけ台','石川台', '緑ヶ丘'],
              '性別': ['m', 'm', 'm', 'm', 'm']})

In [93]: df1
```

Out[93]:

	名前	性別	研究室
0	朝倉先生	m	緑ヶ丘
1	屋井先生	m	すずかけ台
2	室町先生	m	すずかけ台
3	花岡先生	m	石川台
4	福田先生	m	緑ヶ丘

column

index

これも0から数えるので注意！！

Code

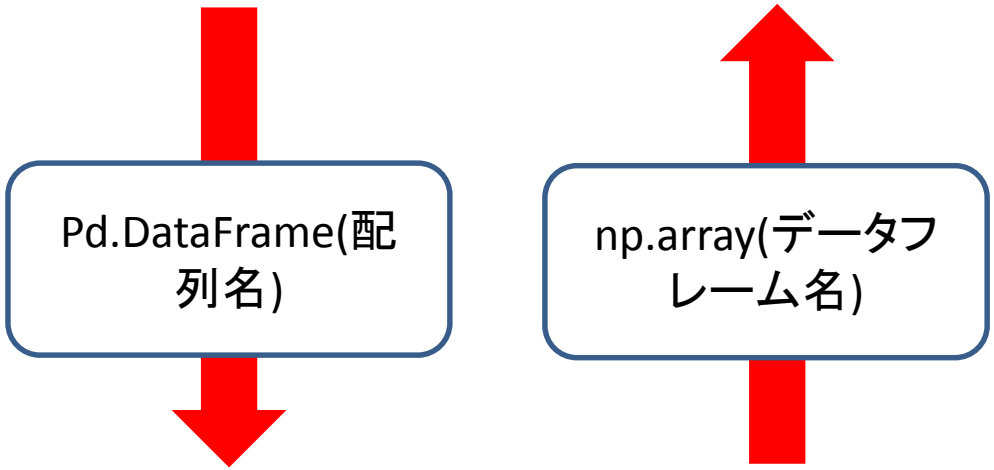
名前 ＝ pandas.DataFrame({‘列名’:[data,data,data…],
‘列名’:[data,data,data…],})

numpy配列とpandas データフレーム

- Numpy のndarray

それぞれのメリットを生かして
型を変換しながら使おう！！！

多次元配列を扱うクラスのこと
メリット：ndarrayは要素の型や要素数を揃えて固定してしまうことで行列演算用の関数が使えて、計算を楽にすばやくできる



- Pandas の DataFrame

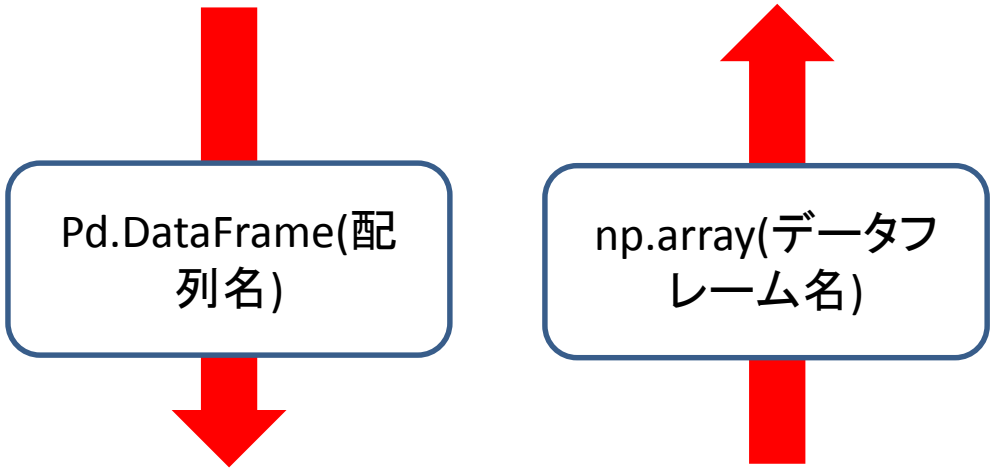
DataFrameはカラムごとに異なる型の値を持つことができる2次元のラベル付けされたデータ配列のこと
メリット：csvをそのまま読み取ることができ，簡単な操作がしやすい

numpy配列とpandas データフレーム

- Numpy のndarray

それぞれのメリットを生かして
型を変換しながら使おう！！！

多次元配列を扱うクラスのこと
メリット：ndarrayは要素の型や要素数を揃えて固定してしまうことで行列演算用の関数が使えて、計算を楽にすばやくできる



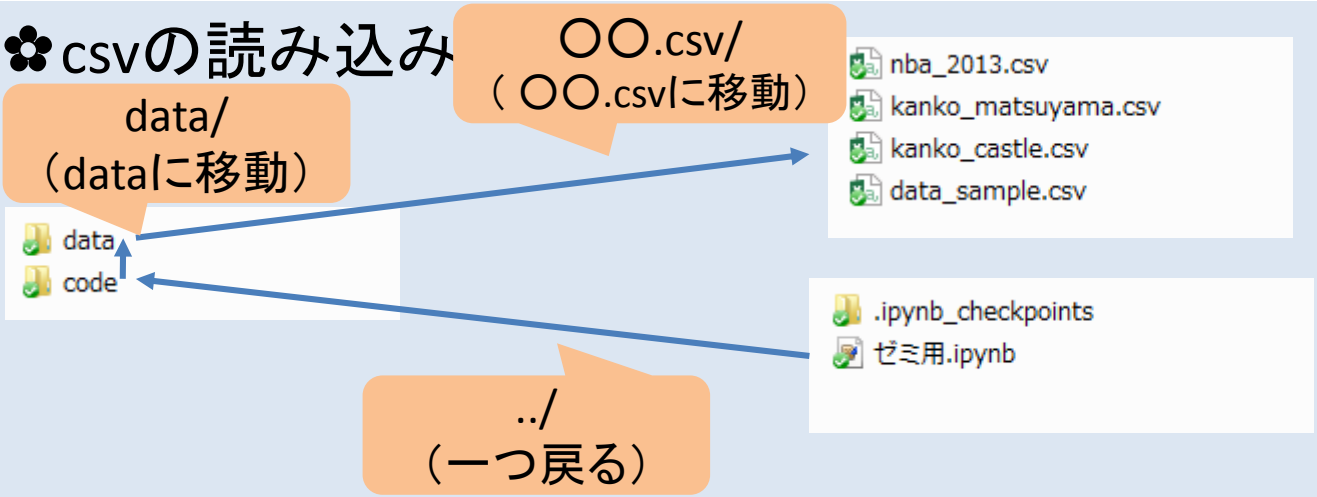
- Pandas の DataFrame

DataFrameはカラムごとに異なる型の値を持つことができる2次元のラベル付けされたデータ配列のこと
メリット：csvをそのまま読み取ることができ，簡単な操作がしやすい

Csvの読み込み

今回使うデータ:
配布済みの“kanko_matsuyama.csv”, “kanko_castle.csv”

✿csvの読み込み




```
In [63]: matsuyama = pandas.read_csv("../data/kanko_matsuyama.csv")
         castle    = pandas.read_csv("../data/kanko_castle.csv")
```

Code
名前　＝　pandas.read_csv(“××/〇〇.csv”)

データの可視化

- データを見てみよう！

```
In [5]: matsuyama
```

Out[5]:

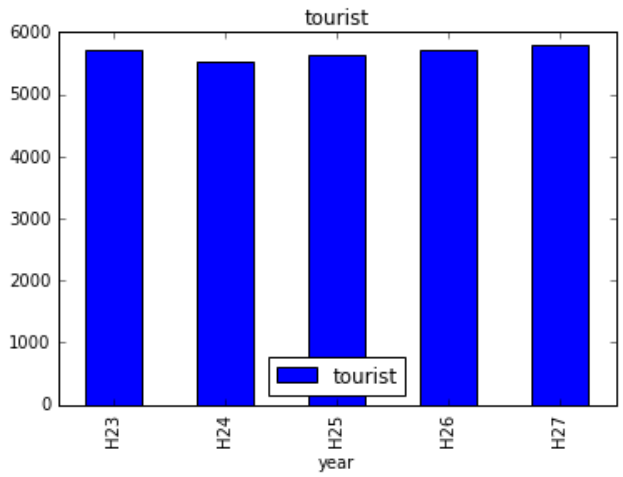
	year	tourist	tourist(kennai)	tourist(kengai)	zennenhi
0	H23	5711	1713	3998	97.100000
1	H24	5523	1657	3866	96.700000
2	H25	5642	1692	3950	102.172732
3	H26	5707	1712	3995	101.200000
4	H27	5804	1741	4063	101.700000

各項目ごとの年次ごとの推移を見たい！

Pythonのoutに出力

```
In [120]: %matplotlib inline
          matsuyama.plot.bar(x=['year'], y=['tourist'] )
          plt.title('tourist')
```

```
Out[120]: <matplotlib.text.Text at 0xc279d68>
```



Code

データフレーム名.plot.bar(x=〇, y=×)
plt.title(‘***’)

データの可視化

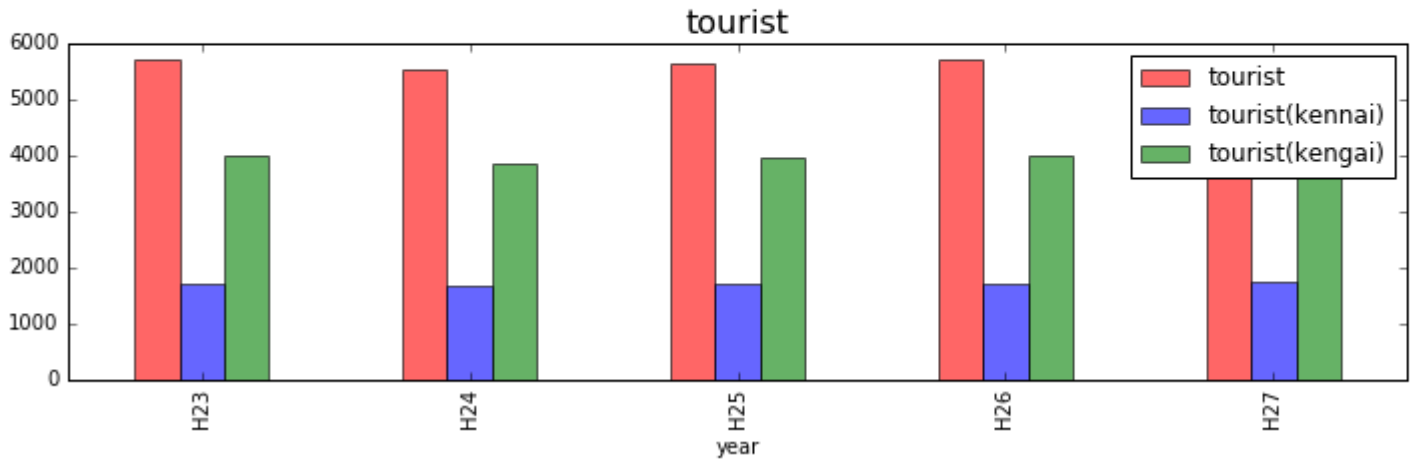
- ちょっと応用編

3つの棒グラフを
重ね合わせる　　淡い色にする　　色の指定

```
%matplotlib inline
matsuyama.plot.bar(x=['year'], y=['tourist','tourist(kennai)','tourist(kengai)'], alpha=0.6, figsize=(12,3),color = ["R","B","G"],)
plt.title('tourist', size=16)
```

```
<matplotlib.text.Text at 0xef23470>
```

文字サイズ

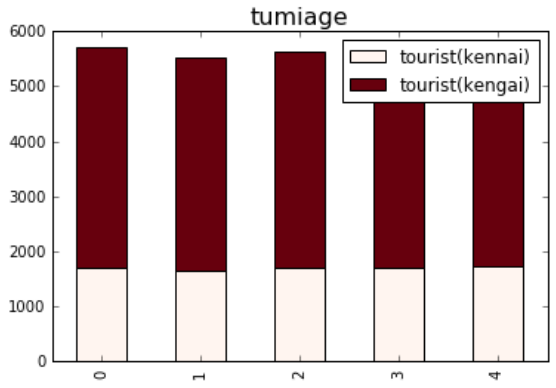


データの可視化

- ちょっと応用編

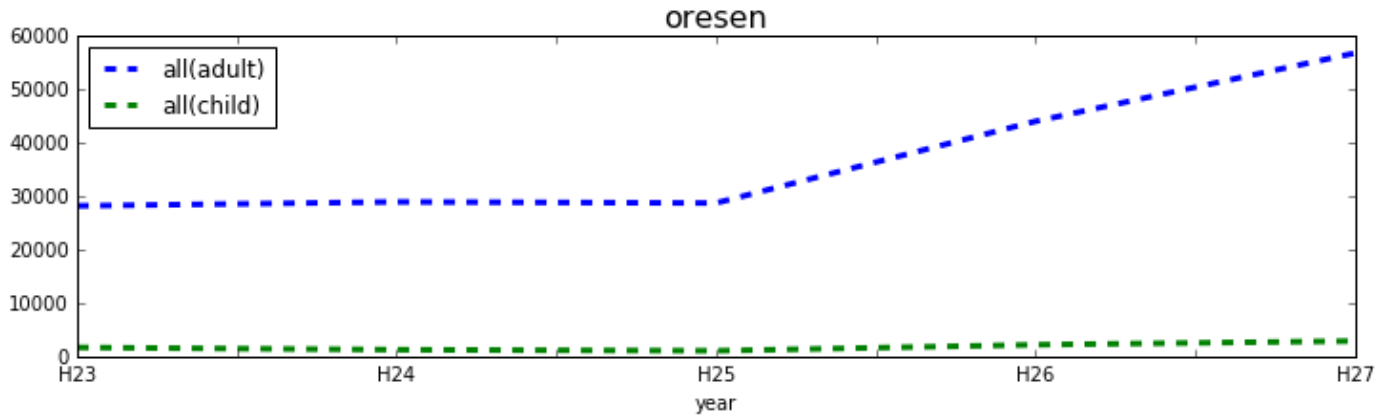
積み上げ棒グラフ 色の指定

```
In [130]: matsuyama.plot.bar(y=['tourist(kennai)', 'tourist(kengai)'], stacked=True, cmap='Reds')
          plt.title('tumiage', size=16)
Out[130]: <matplotlib.text.Text at 0x16d0afd0>
```



線の太さ　破線

```
In [150]: %matplotlib inline
          castle.plot(x=['year'], y=['all(adult)','all(child)'], figsize=(12,3),linewidth=3.0,linestyle='dashed')
          plt.title('oresen', size=16)
Out[150]: <matplotlib.text.Text at 0xf65edd8>
```



データの可視化

- ちょっと応用編

```
In [71]: %matplotlib inline
         x=castleH27_1
         plt.pie(x,labels = castle_label,radius=1,labeldistance=1.5, explode=[0, 0, 0, 0, 0,0.5])
```

半径

強調

```
Out[71]: ([<matplotlib.patches.Wedge at 0xc5ca198>,
           <matplotlib.patches.Wedge at 0xc5cae10>,
           <matplotlib.patches.Wedge at 0xc5d1a90>,
           <matplotlib.patches.Wedge at 0xc5d6710>,
           <matplotlib.patches.Wedge at 0xc5dc390>,
           <matplotlib.patches.Wedge at 0xc5dcfd0>],
          [<matplotlib.text.Text at 0xc5ca9b0>,
           <matplotlib.text.Text at 0xc5d1630>,
           <matplotlib.text.Text at 0xc5d62b0>,
           <matplotlib.text.Text at 0xc5d6ef0>,
           <matplotlib.text.Text at 0xc5dcb70>,
           <matplotlib.text.Text at 0xc5e17f0>])
```