第8章 | Pandasの基礎

Author: sharo

8.1 Pandasの概観

8.1.1 Pandasとは - 1

PandasもNumPyのようにデータの集合を扱うためのライブラリです。NumPyはデータを数学的な行列として扱うことができ、科学計算に特化しています。一方、Pandasでは**一般的なデータベースにて行われる操作が実行でき、数値以外にも氏名や住所といった文字列データも簡単に扱うことができます。データ分析においてNumPyとPandas**を使い分けることで効率的に分析を行うことができます。

8.1.1 Pandasとは - 2

Pandasには**Series**と**DataFrame**という2種類のデータ構造が存在します。主に使われるデータ構造は**DataFrame**です。横方向のデータを**行**、縦方向のデータを**列**と呼びます。各行、各列に対してそれぞれラベルが付与されており、行ラベルは**インデックス**、列ラベルは**カラム**と言います。**Series**は1次元の配列で、**DataFrame**の行、もしくは列として捉える事ができます。

8.1.1 Pandasとは - 3

	Prefecture	Area	Population	Region
0	Tokyo	2190	13636	Kanto
1	Kanagawa	2415	9145	Kanto
2	Osaka	1904	8837	Kinki
3	Kyoto	4610	2605	Kinki
4	Aichi	5172	7505	Chubu

- DataFrameのラベル情報
  - インデックス: [0, 1, 2, 3, 4]
  - カラム: ["Prefecture", "Area", "Population", "Region"]

8.1.2 SeriesとDataFrameのデータの確認 - 1

Seriesでは辞書型を渡すことで辞書のキーで昇順にソートされます。

```
import pandas as pd
fruits = {"orange": 2, "banana": 3}
print(pd.Series(fruits))
```

```
banana  3
orange  2
dtype: int64
```

8.1.2 SeriesとDataFrameのデータの確認 - 2

```
data = {
    "fruits": ["apple", "orange", "banana", "strawberry", "kiwifruit"],
    "year": [2001, 2002, 2001, 2008, 2006],
    "time": [1, 4, 5, 6, 3],
}
df = pd.DataFrame(data)
print(df)
```

```
       fruits  time  year
0       apple     1  2001
1      orange     4  2002
2      banana     5  2001
3  strawberry     6  2008
4   kiwifruit     3  2006
```

8.2 Series

8.2.1 Seriesを生成する - 1

```
fruits = {"banana": 3, "orange": 2}
print(pd.Series(fruits))
```

```
banana  3
orange  2
dtype: int64
```

8.2.1 Seriesを生成する - 2

```
index = ["apple", "orange", "banana", "strawberry", "kiwifruits"]
data = [10, 5, 8, 12, 3]
series = pd.Series(data, index = index)
print(series)
```

```
apple       10
orange       5
banana       8
strawberry  12
kiwifruit    3
dtype: int64
```

8.2.2 参照

```
fruits = {"banana": 3, "orange": 4, "grape": 1, "peach": 5}
series = pd.Series(fruits)
print(series[0:2])
print(series[["orange", "peach"]])
```

```
banana  3
grape   1
dtype: int64
orange  4
peach   5
dtype: int64
```

8.2.3 データ、インデックスを取り出す

```
index = ["apple", "orange", "banana", "strawberry", "kiwifruit"]
data = [10, 5, 8, 12, 3]
series = pd.Series(data, index = index)
series_values = series.values
series_index = series.index
```

8.2.4 要素を追加する

Seriesに要素を追加する場合、追加する要素もまた**Series**型である必要があります。

```
fruits = {"banana": 3, "orange": 2}
series = pd.Series(fruits)
series = series.append(pd.Series([3], index = ["grape"]))
print(series)
```

```
banana    3
orange    2
grape     3
dtype: int64
```

8.2.5 要素を削除する

Seriesのインデックス参照を用いて、要素を削除することができます。

```
index = ["apple", "orange", "banana", "strawberry", "kiwifruit"]
data = [10, 5, 8, 12, 3]
series = pd.Series(data, index = index)
series = series.drop("strawberry")
print(series)
```

```
apple        10
orange        5
banana        8
kiwifruit     3
dtype: int64
```

8.2.6 フィルタリング - 1

```
index = ["apple", "orange", "banana", "strawberry", "kiwifruit"]
data = [10, 5, 8, 12, 3]
series = pd.Series(data, index = index)
conditions = [True, True, False, False, False]
print(series[conditions])
```

```
apple     10
orange     5
dtype: int64
```

8.2.6 フィルタリング - 2

```
index = ["apple", "orange", "banana", "strawberry", "kiwifruit"]
data = [10, 5, 8, 12, 3]
series = pd.Series(data, index = index)
print(series[series >= 5])
print(series[series >= 5][series < 10])
```

```
apple         10
orange         5
banana         8
strawberry    12
dtype: int64
orange    5
banana    8
dtype: int64
```

8.2.7 ソート - 1

Series型の変数 `series` に対して、インデックスについてのソートは `series.sort_index()` 、データについてのソートは `series.sort_values()` で行うことができます。特に指定をしない限りは昇順にソートされますが、引数に `ascending=False` を渡すことで降順にソートされます。

8.2.7 ソート - 2

```
index = ["apple", "orange", "banana", "strawberry", "kiwifruit"]
data = [10, 5, 8, 12, 3]
series = pd.Series(data, index = index)
print(series.sort_index())
print(series.sort_values())
```

```
apple          10
banana          8
kiwifruit       3
orange          5
strawberry     12
dtype: int64
kiwifruit       3
orange          5
banana          8
apple          10
strawberry     12
dtype: int64
```

8.3 DataFrame

8.3.1 DataFrameの生成 - 1

DataFrameは、**Series**を複数束ねたような2次元のデータ構造をしています。`pd.DataFrame()`に**Series**を渡すことで**DataFrame**を生成することができます。行には0から昇順に番号が付きます。

`pd.DataFrame([Series, Series, ...])`

```
data = {
    "fruits": ["apple", "orange", "banana", "strawberry", "kiwifruit"],
    "year": [2001, 2002, 2001, 2008, 2006],
    "time": [1, 4, 5, 6, 3],
}
df = pd.DataFrame(data)
print(df)
```

8.3.1 DataFrameの生成 - 2

out

```
       fruits  year  time
0       apple  2001     1
1      orange  2002     4
2      banana  2001     5
3  strawberry  2008     6
4   kiwifruit  2006     3
```

8.3.2 インデックスとカラムを設定する

```
index = ["apple", "orange", "banana", "strawberry", "kiwifruit"]
data1 = [10, 5, 8, 12, 3]
data2 = [30, 25, 12, 10, 8]
series1 = pd.Series(data1, index = index)
series2 = pd.Series(data2, index = index)
df = pd.DataFrame([series1, series2])
df.index = [1, 2]
print(df)
```

```
   apple  orange  banana  strawberry  kiwifruit
1     10       5       8          12          3
2     30      25      12          10          8
```

8.3.3 行を追加する

```
index = ["apple", "orange", "banana", "strawberry", "kiwifruit"]
data1 = [10, 5, 8, 12, 3]
data2 = [30, 25, 12, 10, 8]
data3 = [30, 12, 10, 8, 25, 3]
series1 = pd.Series(data1, index = index)
series2 = pd.Series(data2, index = index)
index.append("pinapple")
series3 = pd.Series(data3, index = index)
df = pd.DataFrame([series1, series2])
df = df.append(series3, ignore_index = True)
print(df)
```

```
   apple  orange  banana  strawberry  kiwifruit  pinapple
0     10       5       8          12          3       NaN
1     30      25      12          10          8       NaN
2     30      12      10           8         25       3.0
```

8.3.4 列を追加する

```
index = ["apple", "orange", "banana", "strawberry", "kiwifruit"]
data1 = [10, 5, 8, 12, 3]
data2 = [30, 25, 12, 10, 8]
series1 = pd.Series(data1, index = index)
series2 = pd.Series(data2, index = index)
new_column = pd.Series([15, 7], index = [0, 1])
df = pd.DataFrame([series1, series2])
df["mango"] = new_column
print(df)
```

```
   apple  orange  banana  strawberry  kiwifruit  mango
0     10       5       8          12          3     15
1     30      25      12          10          8      7
```

8.3.5 データの参照

DataFrameのデータは行と列を指定することで参照ができます。行、列の指定の仕方により参照の仕方が変わります。参照の方法はいくつかありますが**loc**と**iloc**を扱います。**loc**は名前による参照を行い、**iloc**は番号による参照を行います。

8.3.6 名前による参照 - 1

```
data = {
    "fruit": ["apple", "orange", "banana", "strawberry", "kiwifruit"],
    "year": [2001, 2002, 2001, 2008, 2006],
    "time": [1, 4, 5, 6, 3],
}
df = pd.DataFrame(data)
df = df.loc[[1, 2], ["time", "year"]]
print(df)
```

```
   time  year
1     4  2002
2     5  2001
```

8.3.6 名前による参照 - 2

```
import numpy as np
import pandas as pd
np.random.seed(0)
columns = ["apple", "orange", "banana", "strawberry", "kiwifruit"]
df = pd.DataFrame()
for column in columns:
    df[column] = np.random.choice(range(1, 11), 10)
df.index = range(1, 11)
print(df.loc[range(2, 6), ["banana", "kiwifruit"]])
```

```
   banana  kiwifruit
2      10         10
3       9          1
4      10          5
5       5          8
```

8.3.7 番号による参照

```
data = {
    "fruits": ["apple", "orange", "banana", "strawberry", "kiwifruit"],
    "year": [2001, 2002, 2001, 2008, 2006],
    "time": [1, 4, 5, 6, 3],
}
df = pd.DataFrame(data)
# print(df.iloc[[1, 3], [0, 2]])
print(df)
```

```
        fruits  time
1       orange     4
3  strawberry     6
```

教科書と実行結果が異なるので注意

8.3.8 行または列の削除

```
data = {
    "fruits": ["apple", "orange", "banana", "strawberry", "kiwifruit"],
    "time": [1, 4, 5, 6, 3],
    "year": [2001, 2002, 2001, 2008, 2006],
}
df = pd.DataFrame(data)
print(df.drop(range(0, 2)))
print(df.drop("year", axis = 1))
```

```
       fruits  time  year
2      banana     5  2001
3  strawberry     6  2008
4   kiwifruit     3  2006
       fruits  time
0       apple     1
1      orange     4
2      banana     5
3  strawberry     6
4   kiwifruit     3
```

8.3.9 ソート

```
data = {
    "fruits": ["apple", "orange", "banana", "strawberry", "kiwifruit"],
    "time": [1, 4, 5, 6, 3],
    "year": [2001, 2002, 2001, 2008, 2006],
}
df = pd.DataFrame(data)
print(df.sort_values(by = "year", ascending = True))
print(df.sort_values(by = ["time", "year"], ascending = True))
```

```
       fruits  time  year
0       apple     1  2001
2      banana     5  2001
1      orange     4  2002
4   kiwifruit     3  2006
3  strawberry     6  2008
       fruits  time  year
0       apple     1  2001
4   kiwifruit     3  2006
1      orange     4  2002
2      banana     5  2001
3  strawberry     6  2008
```

8.3.10 フィルタリング - 1

```
data = {
    "fruits": ["apple", "orange", "banana", "strawberry", "kiwifruit"],
    "time": [1, 4, 5, 6, 3],
    "year": [2001, 2002, 2001, 2008, 2006],
}
df = pd.DataFrame(data)
print(df.index % 2 == 0)
print(df[df.index % 2 == 0])
```

```
      fruits  time  year
0      apple     1  2001
2     banana     5  2001
4  kiwifruit     3  2006
```

8.3.10 フィルタリング - 2

```
import numpy as np
import pandas as pd
np.random.seed(0)
columns = ["apple", "orange", "banana", "strawberry", "kiwifruit"]
df = pd.DataFrame()
for column in columns:
    df[column] = np.random.choice(range(1, 11), 10)
df.index = range(1, 11)
print(df.loc[(df["apple"] >= 5) & (df["kiwifruit"] >= 5)])
# df.loc[df["apple"] >= 5][df["kiwifruit"] >= 5)]ではwarningが出る
```

```
   apple  orange  banana  strawberry  kiwifruit
1      6       8       6           3         10
5      8       2       5           4          8
8      6       8       4           8          8
```