04 CSV・Excel ファイルの読み込み・書き出し

YouTube はこちら

「ここでは、CSV や Excel ファイルを読み込む（取り入れる）方法を学びます。」

「データを読み込むと何かいいことがあるのですか？」

「CSV や Excel ファイルを読み込むと、データフレームやシリーズをリストなどを使って作成する必要がなくなります。行数・列数の多いデータの集計や加工処理をしたい時にとても活躍してくれるんです。」

「それはすごい！　なんだかわくわくしてきました。」

●使用データ

ここでは data.csv,data01.csv,data.xlsx,data01.xlsx,data02.xlsx,data03.xlsx を使用します。それぞれのファイルには、政府発表の「1920 年から 2015 年までの全国の人口推移のデータ」が格納されています。

まず、最初にファイルパスの設定を行います。変数に代入しておくことで、別のファイルを読み込んで使用したい時、ここだけ編集すればよいので便利です。

```
data_csv_path = './Data/MyPandas/data.csv'
data_csv01_path = './Data/MyPandas/data01.csv'
data_xl_path = './Data/MyPandas/data.xlsx'
data01_path = './Data/MyPandas/data01.xlsx'
data02_path = './Data/MyPandas/data02.xlsx'
data03_path = './Data/MyPandas/data03.xlsx'
```

```
import pandas as pd
```

●CSVデータを読み込み

set_option メソッドで、読み込んだデータを表示させる行数や列数を指定できます。
ここでは 10 行に制限して表示させてみましょう。
全ての行を表示させたい場合は、行数を None に設定します。
また、head メソッドや tail メソッドでも行数を指定できますが、その都度記述するのが手間な場合は set_opition を使うと良いでしょう。

```
pd.set_option('display.max_rows', 10)
```

read_csv メソッドを使って、CSV ファイルを読み込みます。
引数 encoding で、文字コードを指定することができます。
今回は shift-jis を指定しますが、他にも utf-8 等も指定できます。

```
df_csv = pd.read_csv(data_csv_path, encoding='shift-jis')
df_csv
```

	都道府県コード	都道府県名	元号	和暦（年）	西暦（年）	人口（総数）	人口（男）	人口（女）
0	1	北海道	大正	9.0	1920.0	2359183	1244322	1114861
1	2	青森県	大正	9.0	1920.0	756454	381293	375161
2	3	岩手県	大正	9.0	1920.0	845540	421069	424471
3	4	宮城県	大正	9.0	1920.0	961768	485309	476459
4	5	秋田県	大正	9.0	1920.0	898537	453682	444855
...	...	...	...	...	...	...	...	...

出力結果

●カラム名を指定して読み込み

カラム名の無い data01.csv ファイルを読み込み、カラム名を追加します。
同様に read_csv メソッドを使ってファイルを読み込みます。
引数 names にカラム名のリストを代入します。

```
df_csv = pd.read_csv(data_csv01_path, encoding='shift-jis',
        names = ['area_code','area', 'GG', 'gg',  'yyyy',
'population', 'man', 'woman'])
df_csv
```

	area_code	area	GG	gg	yyyy	population	man	woman
0	1	北海道	大正	9	1920	2359183	1244322	1114861
1	2	青森県	大正	9	1920	756454	381293	375161
2	3	岩手県	大正	9	1920	845540	421069	424471
3	4	宮城県	大正	9	1920	961768	485309	476459
4	5	秋田県	大正	9	1920	898537	453682	444855
...	...	...	...	...	...	...	...	...

出力結果

●インデックスを指定して読み込み

指定した列をインデックスとして、CSV を読み込むこともできます。

ここでは、都道府県名をインデックスに指定します。

read_csv メソッドの引数 index_col を追加し、指定したい列番号や列名を代入します。

列番号は、リストと同じように 0 から始まります。

したがって、都道府県の列番号は 1 です。

```
df_csv = pd.read_csv(data_csv_path, encoding='shift-jis', index_
col=1)
df_csv
```

都道府県名	都道府県コード	元号	和暦（年）	西暦（年）	人口（総数）	人口（男）	人口（女）	人口（女）
北海道	1	大正	9.0	1920.0	2359183	1244322	1114861	1114861
青森県	2	大正	9.0	1920.0	756454	381293	375161	375161
岩手県	3	大正	9.0	1920.0	845540	421069	424471	424471
宮城県	4	大正	9.0	1920.0	961768	485309	476459	476459
秋田県	5	大正	9.0	1920.0	898537	453682	444855	444855
...	...	...	...	...	...	...	...	...

出力結果

●複数のインデックスを指定して読み込み

5 列目までをインデックスとし、データ部分を人口データのみとします。

インデックスの列を複数指定する場合は、引数 index_col に列番号のリストを代入渡します。

```
df_csv = pd.read_csv(data_csv_path, encoding='shift-jis', index_
col=[0, 1, 2, 3, 4])
df_csv.head()
```

					人口（総数）	人口（男）	人口（女）
都道府県コード	都道府県名	元号	和暦（年）	西暦（年）			
1	北海道	大正	9.0	1920.0	2359183	1244322	1114861
2	青森県	大正	9.0	1920.0	756454	381293	375161
3	岩手県	大正	9.0	1920.0	845540	421069	424471
4	宮城県	大正	9.0	1920.0	961768	485309	476459
5	秋田県	大正	9.0	1920.0	898537	453682	444855
4	宮城県	大正	9.0	1920.0	961768	485309	476459
5	秋田県	大正	9.0	1920.0	898537	453682	444855

出力結果

このように、複数の列がインデックスである状態を「マルチインデックス」といいます。

●CSV ファイルに書き出し

```
df_csv.to_csv('data_csv.csv', encoding='shift-jis')
```

to_csv メソッドを使って、CSV ファイルを書き出します。

引数 encoding で、shift-jis を指定します。

これによって、CSV ファイルに書き込んだときに文字化けを防ぐことができます。

●Excel ファイルの読み込み

read_excel メソッドで Excel のデータを読み込みます。

read_csv と read_excel は、使い方は非常に似ています。

```
pd.read_excel(data_xl_path)
```

	都道府県コード	都道府県名	元号	和暦（年）	西暦（年）	人口（総数）	人口（男）	人口（女）
0	1	北海道	大正	9	1920	2359183	1244322	1114861
1	2	青森県	大正	9	1920	756454	381293	375161
2	3	岩手県	大正	9	1920	845540	421069	424471
3	4	宮城県	大正	9	1920	961768	485309	476459
4	5	秋田県	大正	9	1920	898537	453682	444855
…	…	…	…	…	…	…	…	…

出力結果

data01.xlsx は、最初の 2 行が空白です。

カラム名が Unnamed や、1 行目が NaN になっていますね。

```
pd.read_excel(data01_path)
```

	Unnamed: 0	Unnamed: 1	Unnamed: 2	Unnamed: 3	Unnamed: 4	Unnamed: 5	man	woman
0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1	都道府県コード	都道府県名	元号	和暦（年）	西暦（年）	人口（総数）	人口（男）	人口（女）
2	1	北海道	大正	9	1920	2359183	1244322	1114861
3	2	青森県	大正	9	1920	756454	381293	375161
4	3	岩手県	大正	9	1920	845540	421069	424471
…	…	…	…	…	…	…	…	…

出力結果

●カラム名を指定してExcelファイルを読み込み

最初の 2 行をスキップして Excel ファイルを読み込む記述をします。

引数 skiprows に、スキップする行数を記述します。

```
pd.read_excel(data01_path, skiprows=2)
```

	都道府県コード	都道府県名	元号	和暦（年）	西暦（年）	人口（総数）	人口（男）	人口（女）
0	1	北海道	大正	9	1920	2359183	1244322	1114861
1	2	青森県	大正	9	1920	756454	381293	375161
2	3	岩手県	大正	9	1920	845540	421069	424471
3	4	宮城県	大正	9	1920	961768	485309	476459
4	5	秋田県	大正	9	1920	898537	453682	444855
…	…	…	…	…	…	…	…	…

出力結果

Excel や CSV ファイルにカラム名がある場合は、header を明示的に指定する方法もあります。

1 行目をカラム名にしている場合は、引数 header に 0 を指定します。

デフォルトでは 1 行目がヘッダーに設定されるようになっているので変化はありません。

```
pd.read_excel(data01_path, skiprows = 2, header=[0])
```

	都道府県コード	都道府県名	元号	和暦（年）	西暦（年）	人口（総数）	人口（男）	人口（女）
0	1	北海道	大正	9	1920	2359183	1244322	1114861
1	2	青森県	大正	9	1920	756454	381293	375161
2	3	岩手県	大正	9	1920	845540	421069	424471
3	4	宮城県	大正	9	1920	961768	485309	476459
4	5	秋田県	大正	9	1920	898537	453682	444855
…	…	…	…	…	…	…	…	…

出力結果

引数 header を 1 とすると、このように 2 行目がヘッダーとして表示されます。

```
pd.read_excel(data01_path, skiprows = 2, header=[1])
```

	1	北海道	大正	9	1920	2359183	1244322	1114861
0	2	青森県	大正	9	1920	756454	381293	375161
1	3	岩手県	大正	9	1920	845540	421069	424471
2	4	宮城県	大正	9	1920	961768	485309	476459
3	5	秋田県	大正	9	1920	898537	453682	444855
4	6	山形県	大正	9	1920	968925	478328	490597
...	...	...	...	...	...	...	...	...

出力結果

列名の無い data02.xlsx ファイルを読み込みます。

ヘッダーのないファイルの場合は、引数 header に None を渡すと、カラム名に自動的に連番を振ります。

```
pd.read_excel(data02_path, header=None)
```

	0	1	2	3	4	5	6	7
0	1	北海道	大正	9	1920	2359183	1244322	1114861
1	2	青森県	大正	9	1920	756454	381293	375161
2	3	岩手県	大正	9	1920	845540	421069	424471
3	4	宮城県	大正	9	1920	961768	485309	476459
4	5	秋田県	大正	9	1920	898537	453682	444855
...	...	...	...	...	...	...	...	...

出力結果

data03.xlsx というファイルを開き、1 行目と 2 行目をカラムに設定します。

1 行目は地域、2 行目には各列の名前が入っています。

```
pd.read_excel(data03_path, header=[0, 1])
```

地域		羽田		成田		千歳	
	年月日	平均気温 (℃)	降水量の合計 (mm)	平均気温 (℃)	降水量の合計 (mm)	平均気温 (℃)	降水量の合計 (mm)
0	2019/1/1	6.8	0.0	2.0	0.0	-4.7	0.0
1	2019/1/2	7.3	0.0	3.9	0.0	-6.9	0.0
2	2019/1/3	6.1	0.0	2.3	0.0	-7.4	0.0
3	2019/1/4	6.5	0.0	2.2	0.0	-2.6	0.0
4	2019/1/5	8.3	0.0	6.2	0.0	-3.3	4.5
...	...	...	...	...	...	...	...

出力結果

●インデックスを指定してExcelファイルを読み込み

```
pd.read_excel(data_xl_path, index_col = '都道府県コード')
```

都道府県コード	都道府県名	元号	和暦(年)	西暦(年)	人口(総数)	人口(男)	人口(女	人口(女)
1	北海道	大正	9	1920	2359183	1244322	1114861	1114861
2	青森県	大正	9	1920	756454	381293	375161	375161
3	岩手県	大正	9	1920	845540	421069	424471	424471
4	宮城県	大正	9	1920	961768	485309	476459	476459
5	秋田県	大正	9	1920	898537	453682	444855	444855
...	...	...	...	...	...	...	...	...

出力結果

再度、data.xlsx のファイルを開きます。

引数 index_col で、列名の都道府県コードをインデックスに設定します。

●インデックスを日付型として読み込む

data03.xlsx を再度読み込み、インデックスを日付型として表示させてみましょう。

地域が記載された 1 行目はスキップさせ、2 行目から読み込みます。

```
df_excel = pd.read_excel(data03_path, skiprows=1, index_col='年月日')
df_excel
```

	平均気温(℃)	降水量の合計(mm)	平均気温(℃).1	降水量の合計(mm).1	平均気温(℃).2	降水量の合計(mm).2
年月日						
2019/1/1	6.8	0.0	2.0	0.0	-4.7	0.0
2019/1/2	7.3	0.0	3.9	0.0	-6.9	0.0
2019/1/3	6.1	0.0	2.3	0.0	-7.4	0.0
2019/1/4	6.5	0.0	2.2	0.0	-2.6	0.0
2019/1/5	8.3	0.0	6.2	0.0	-3.3	4.5
...	...	...	...	...	...	...

インデックスのデータ型を type メソッドで確認します。

通常のインデックスになっています。

```
type(df_excel.index)
```

```
pandas.core.indexes.base.Index
```

出力結果

さらに、読み込む際に引数 parse_dates に True を渡します。
こうすることで、インデックスで指定された列が日付型として読み込まれます。
日付部分がハイフン表示になります。

```
df_excel = pd.read_excel(data03_path, skiprows=1, index_col='年月日
', parse_dates=True)
df_excel
```

年月日	平均気温（℃）	降水量の合計（mm）	平均気温（℃）.1	降水量の合計（mm）.1	平均気温（℃）.2	降水量の合計（mm）.2
2019-01-01	6.8	0.0	2.0	0.0	-4.7	0.0
2019-01-02	7.3	0.0	3.9	0.0	-6.9	0.0
2019-01-03	6.1	0.0	2.3	0.0	-7.4	0.0
2019-01-04	6.5	0.0	2.2	0.0	-2.6	0.0
2019-01-05	8.3	0.0	6.2	0.0	-3.3	4.5
...	...	...	...	...	...	...

出力結果

データ型が DatetimeIndex となり、日付型が確認できます。
インデックスが日付型になると、特殊な集計やメソッドを使うことができます。

```
type(df_excel.index)
```

```
pandas.core.indexes.datetimes.DatetimeIndex
```

出力結果

●Excelファイルに書き出し

to_excel メソッドを使って、データフレームを Excel データへ書き出すことができます。

```
df_excel.to_excel('df_excel.xlsx')
```