

9. Excel/CSV ファイルを読む

Reading Excel and CSV files

プログラミング・データサイエンス I

2024/6/6

1 今日の目的

今日の目的

- データサイエンス
 - データ分析→客観的状況
 - 課題発見、施策立案
- Excel / CSV ファイルを読み込む
- Python の表形式 DataFrame の基本
- Python のデータ形式 Series の基本

データを分析することで、状況を客観的に把握し、課題発見や施策立案に生かそうというのが、データサイエンスです。python では、データ分析に利用できる様々なライブラリを利用することができます。これが、データサイエンスの分野で python が良く利用される理由でしょう。

データの収集や分析で良く使われるアプリケーションの一つが Microsoft Excel です。今回は、Python から Excel の内容を読みましょう。Excel ファイルがあまり大きくなく、かつ一つならば、分析するためにプログラムを書くことは、必要ないかもしれません。Excel の関数やマクロを使用すれば十分です。しかし、ファイルが非常に大きい場合、沢山のファイルをまとめて処理したい場合、柔軟に作図したい場合等には、ちょっとプログラムを書くことができると、処理がすごく楽になります。

Python には、pandas というライブラリがあります。pandas は、表形式のデータを保存する DataFrame、行や列の次元データを保持する Series というデータ形式を持っています。これらのデータ形式について、その基本を学ぶのが今日の目的です。pandas

の公式ホームページには関数の使い方や、利用ガイドがあります。^{*1}

それでは、今日のサンプルプログラムを取得してください。

<https://github.com/first-programming-saga/excelAndCSV>

2 Excel ファイルの読み込み: Reading Excel files

2.1 Excel ファイルの読み込み

Excel ファイルの読み込み: Reading Excel files

- `pandas.read_excel()` を利用して excel を読む
- 内容は `pandas.DataFrame` クラスのインスタンス

表 1 data.xlsx の内容

	English	Math	Science	Social
Tim	80	90	95	70
John	80	60	70	100
Kim	100	60	65	80
Sally	70	80	95	70
Tom	80	70	80	60
Bob	70	100	90	80

ソースコード 2.1 Excel ファイルの読み込み

```
1 filename = 'data.xlsx'  
2 #Excel ファイルを読み、pandas.DataFrame とする  
3 with pandas.ExcelFile(filename) as f:  
4     data = pandas.read_excel(f, index_col = 0, header = 0)
```

最初の例は、表 1 に示す内容の data.xlsx を使うプログラムです。python を使って、表の中の特定の行、列、セルを指定する方法を学んでいきます。それでは、`howToUseDataFrame.ipynb` を開けてください。

^{*1} <https://pandas.pydata.org/>

Excel ファイルを python から開くのは簡単です (ソースコード 2.1)。3 行目では、filename でファイル名を指定した Excel ファイルを開いています。with で始まるプログラムブロックで、ファイルを開き、その読み出し口であるファイルオブジェクトに f という名前を付けます。実際のファイル読み出しは、4 行目で読みだした結果を data に保存します。with で始まるプログラムブロックが終わると、ファイルを正しく閉じてくれます。

4 行目の、f からファイルを読み込み部分を見ましょう。二番目の引数 index_col = 0 は、一番左の列を行の名前として使用することを指定しています。これは、デフォルト値ですから、指定しなくても大丈夫です。最後の引数 header = 0 は、一行目を列の名前として使用することを指定しています。こちらもデフォルト値です。

簡単のために、Excel ファイルには、一つのシートしかないとします。この場合、4 行目で読み込んだものは、pandas の DataFrame というデータ構造になります。

2.2 行と列

行と列:Columns and rows

- 列の名前一覧: columns
- 行の名前一覧: index
- 列データ取り出し
- 行データ取り出し

ソースコード 2.2 行と列の名前の認識

```
1 print('dataFrame の内容')
2 print(data)
3 print(' 認識された列名')
4 print(data.columns)
5 print(' 認識された行名')
6 print(data.index)
7 print()
```

Excel の表を pandas で読み込むと、**DataFrame 型**のデータとなります。DataFrame は、行と列のラベルを持つ表形式のデータで、様々な操作のメソッドを持っています。DataFrame では、指定された行を columns、つまり**各列のラベル**として認識し、指定された列を index、つまり**各行のラベル**として認識します。data.columns という指定は、

ソースコード 2.3 data の出力

	English	Math	Science	Social
Tim	80	90	95	70
John	80	60	70	100
Kim	100	60	65	80
Sally	70	80	95	70
Tom	80	70	80	60
Bob	70	100	90	80

`data` という `DataFrame` オブジェクトの `columns` という属性という意味です。オブジェクトに属性がある場合には、この例のようにピリオドの後に属性名を指定します。正しく認識していることを確認しましょう (ソースコード 2.2)。

ソースコード 2.2 の 2 行目は、`DataFrame` 全体を表示します。その結果は、ソースコード 2.3 のようになります。また、4 行目と 6 行目は、それぞれ列のラベルと行のラベルを表示します。列のラベル及び行のラベルを取り出して利用できることがわかります。

次に、読み込んだ `DataFrame` 形式のデータ `data` から、行や列のデータを取り出しましょう。行や列のデータは、**Series 形式**になります。**列の取り出し**は、列の名前を指定して行います。ソースコード 2.4 の 2 行目では、`Math` の成績を取り出しています。

ソースコード 2.4 行と列の取り出し

```
1 print(' 列の取り出し例')
2 print(data['Math'])
3 print()
4 print(' 行の取り出し例')
5 print(data.loc['Kim'])
6 print()
```

取り出しの結果は、ソースコード 2.5 のようになります。単に列の値を取り出すだけでなく、対応する行をラベルとして持った形になっていることがわかります。つまり、`Series` は、単なるリストではありません。そのため、`Series` になったものに対して、直接的に列の名前を指定して、値を取り出すことができます。

また、**行の取り出し**も、形式が異なりますが、行の名前を指定します。`DataFrame` 形式のデータが持つ、`loc` という属性を使用します。5 行目では、`Kim` を指定して、行データを取り出しています。同様に、列の名前がラベルとなっていることを確認しましょう。

ソースコード 2.5 列の出力: Math の列を取り出している

```
Tim      90
John     60
Kim      60
Sally    80
Tom      70
Bob     100
Name: Math, dtype: int64
```

ソースコード 2.6 行と列の取り出し

```
1 print('セルの取り出し例: Kim の Math の成績')
2 print(data['Math']['Kim'])
3 print(data.loc['Kim', 'Math'])
4 print(data.at['Kim', 'Math'])
5 print(data.iloc[2,1])
6 print()
7 print('全員の Math と Science の成績を取り出し')
8 df2 = data.iloc[:, 1:3]
9 print(df2)
```

特定のセルの値は、行と列の名前をそれぞれ指定します。ソースコード 2.6 の 2 行目から 4 行目です。data['Math'] で列のデータを取り出した結果が Series 形式のデータです。Series 形式のデータには、元の行に相当する名前がついています。そこで、data['Math']['Kim'] で、Kim の Math の成績を取り出しています。

一方、data.loc['Kim'] で一行のデータを取り出したものも Series 形式のデータです。こんどは、元の列に相当する名前が付いています。そこで、data.loc['Kim', 'Math'] で、Kim の Math の成績を取り出しています。

data.at['Kim', 'Math'] では、行と列のラベルを使って、data.iloc[2,1] では行と列の位置を使って、セルを指定しています。

セルの位置を数字の範囲で指定すると、部分的な DataFrame を取り出すことが出来ます。14 行目の例では、最初の部分が: となっています。つまり、全ての行が対象です。二番目に 1:3 を指定し、Math と Science の成績を取り出しています。結果は、新しい DataFrame 形式のデータ df2 に保存しています。

セルの指定方法として、loc と at は似ていますが、使い方に異なるところがあります。単一のセルを指定する場合には、at を使います。loc は、iloc と同様に、行や列の

範囲を指定することができます。ソースコード 2.7 では、`loc` を使って、Kim と Bob という二人について、Math と Science という二科目の成績を取り出しています。結果は、`DataFrame` 形式のデータになります。

```
1 df3 = data.loc[['Kim', 'Bob'], ['Math', 'Science']]
```

ソースコード 2.7 複数の行と列を指定する

課題 2.1 セルを指定することができたら、その値を変更することができます。実際に行ってみましょう。`howToUseDataFrame.ipynb` の最後の課題を実施してください。

3 DataFrame の操作: Manipulating DataFrame

— DataFrame の操作: Manipulating DataFrame —

- `Series` の操作
- `index` のある列に名前を付ける

次に、`howToUseDataFrame2.ipynb` を開けてください。同じ `data.xlsx` を使う例です。一行や一列のデータは、`Series` という形式のデータとなります。ソースコード 3.1 では、一行のデータを取り出しています。`Series` のデータには、インデクスが付いています。行のデータを取り出すと、表の列のラベルが `Series` のデータのインデクスになります。3 行目からの `for` ループでは、インデクスを使って一つ一つデータを取り出して印刷します。

ソースコード 3.1 `Series` の操作

```
1 print('Series の操作')
2 ser = data.loc['Tim']
3 for k in ser.index:
4     v = ser[k]
5     print(f'ser[{k}]:{v}')
```

使用している `data.xlsx` 一番左の列には、名前がついていませんでした。ファイルを読み込んだ後から、列に名前を付けることができます。ソースコード 3.2 の 3 行目で、行

に名前 `name` を付けています。ソースコード 3.3 がその結果です。

ソースコード 3.2 行に名前を付ける

```
1 print("操作前")
2 print(data.index.name)
3 data.index.name = "name"
4 print("操作後")
5 print(data.index.name)
6 print(data)
```

ソースコード 3.3 行に名前を付けた結果

	English	Math	Science	Social
name				
Tim	80	90	95	70
John	80	60	70	100
Kim	100	60	65	80
Sally	70	80	95	70
Tom	80	70	80	60
Bob	70	100	90	80

3.1 Series に対する関数

Excel では、セルに対して関数を適用することができます。例えば、行や列の合計、平均、最大値、最小値などを求めることができます。同様に、Series に対して用いることができる、様々な関数が用意されています。ここでは、幾つか例を見ることにします。使用例をソースコード 3.4 に示します。

- `mean()`: 平均
- `min()`: 最小値
- `max()`: 最大値
- `sum()`: 和
- `median()`: 中央値

ソースコード 3.4 Series に対する関数の例

```
1 ser :pandas.Series = data['Math']
2 print(' 平均:',ser.mean())
3 print(' 最高点:',ser.max())
4 print(' 最低点:',ser.min())
```

4 CSV ファイルを読む: Reading CSV files

CSV ファイルを読む: Reading CSV files

- CSV ファイルを読む
- Python で、一列目を行のインデクスに設定する

最後に、`howToUseDataFrame3.ipynb` を開けてください。CSV ファイルを読む例題です。CSV ファイルを読み込むと、Excel の場合と同様に、`DataFrame` 形式のデータとなります。ソースコード 4.1 は、引数で指定した CSV ファイルを開けて、`DataFrame` 形式のデータを返す関数です。

ソースコード 4.1 CSV ファイルを読み、`DataFrame` を返す関数

```
1 def readFromCSV(filename:str)->pandas.DataFrame:
2     """
3     Excel ファイルを読み、pandas.DataFrame とする
4     """
5     with open(filename) as f:
6         data = pandas.read_csv(f)
7     return data
```

CSV ファイルでも、`index_col` や `header` を指定することが出来ますが、今回は指定していません。その結果がソースコード 4.2 です。行の名前の代わりに、行の番号が左端に出ています。なお、読み込んだ CSV ファイルでは、左上端のセルに `name` という文字列を入れていました。

`howToUseDataFrame3.ipynb` の `main` 部分がソースコード 4.3 です。読み込んだ `DataFrame` 形式のデータは `df` という名前で保存しています。5 行目の `set_index()` メソッドによって、`name` というラベルのある列を行のインデクスとして設定しています。

ソースコード 4.2 index_col を指定しない場合

	name	English	Math	Science	Social
0	Tim	80	90	95	70
1	John	80	60	70	100
2	Kim	100	60	65	80
3	Sally	70	80	95	70
4	Tom	80	70	80	60
5	Bob	70	100	90	80

inplace=True を指定することで、df そのもので、name 列をインデクスとして指定することができます。正しく、行のインデクスが設定できていることを確認してください。

ソースコード 4.3 main 部分

```
1 filename = 'data2.csv'
2 df=readFromCSV(filename)
3 showColumnsAndIndex(df)
4 print('name カラムを row index として指定')
5 df.set_index('name',inplace=True)
6 print(df)
7 print(df.iloc[0])
```

5 実際のデータをエクセルを読む

インターネット上には、様々なデータが公開されています。公開されているデータの中には、エクセル形式のものや CSV のように、表形式のものが多数あります。例として、佐賀県の人口に関するデータを読み込んでみましょう。

佐賀県では、情報公開として Excel ファイルなどを公開しています。今日は以下の URL にあるファイルを使います。

<http://data.bodik.jp/dataset/77e0cc66-c15d-4473-b3df-2664fe8e2e63/resource/8dc71515-526a-4168-866c-05d2cc8dad7b/download/jinkou.xlsx>

この URL に Python から直接接続してデータを読むには、ちょっと工夫が必要です。そこで、同じファイルを配布しています。見てください。

このエクセルを見ると、データを分析しようとする際に、問題になりそうな点がいくつかあることに気が付きます。例えば、以下のような点があります。

- 列のタイトルが 4 行目にある。また、和暦であって、元号が省略されている
- 5 行目、19 行目、28 行目のようにデータが入っていない行がある
- 行の名前として A 列と B 列が連結されている。V 列にも行の名前がある。
- 31 行目以降に、説明がついている
- 数値でなく、“-”が入っているセルがある

ソースコード 5.1 必要な部分だけの読み込み

```

1 data = pandas.read_excel(url, header = 3, usecols = 'C:V',
2   index_col = 19, skiprows = [4, 18, 27], skipfooter = 8)

```

Pandas は、これらの問題の幾つかを、エクセル読み込み時に解決できる機能を持っています。例えば、ソースコード 5.1 では、

- `header = 3`: 4 行目が列の名前
- `usecols = 'C:V'`: C 列から V 列を利用
- `index_col = 19`: V 列が行名
- `skiprows = [4, 18, 27]`: 指定した行を読まない
- `skipfooter = 8`: 最後の 8 行は無視

を指定しています。このような処理で、概ね必要なデータを `DataFrame` に保存することができます。

更に、処理が行いやすいようにするには、列名や行名を変更することです。配布した `sagaPopulation0.ipynb` では、`DataFrame.rename()` を使って、以下の変更をしています。詳しくは、コードを参照してください。

- 列名を和暦から西暦に変更
- 行の名前を A 列と B 列を活用して変更

課題 5.1 整形した `DataFrame` を確認しなさい。

6 課題

quiz.ipynb 末尾にあるように、data.xlsx について、各科目の平均値を計算し、結果を印刷するプログラムを作成し、実行しなさい。

7 次回

次回は、プログラム内で表を作り、エクセルファイルに出力します。