

コンピュータリテラシ発展 ～Pythonを学ぶ～

第10回：表計算やデータ分析をやってみよう

学 部 学 科 メディア専攻

清水 拓也 (shimizu@info.shonan-it.ac.jp)



今回の授業内容

今回の授業内容

- 1回の課題 解
- データ分析を始めるために
- pandasの使い方
- 課題

前回の課題解説

前回の課題解説

- 前回の課題の解法例を示します
- 解法例について質問があればご連絡ください

解答例

<https://colab.research.google.com/drive/1LhxzCzdAQu8Utigrpj2KZTrR1oMnll69?usp=sharing>

データ分析を始める前に

ExcelとPythonの使い分け

- Excelは表計算ソフトウェアの 直感的な使いやすさと普及率を持っており依然として 頻りに使 されます
- Pythonを使ったデータ分析の利 点はデータ収集や 処理の簡便さ、大規模なデータに 対する軽快な 作、機械学習の 意 があります
- JupyterLab (Google Colab)を使うことでデータ分析の過程を記 録し共有が 易です
- データ分析を 業にしている人々はPythonやRというプログラミング言語を多く利 用しています

データ処理の流れ

- 大きく分けて**3ステップ**あります
- Pythonで各ステップを実施する際には,それぞれのステップで異なるライブラリを使用したり, 組み合わせることが必ずになります

1. データ収集

2. 前処理

3. 集計・可視化

データ分析の流れ

1. データ処理

- 必要なデータを集めます
- 集める場所はWebであったり社内^{社内}の各部署^{部署}が管理しているファイル（デジタル、アナログ）です
- デジタルデータであれば、プログラムである利点は解決します
- アナログの場合は、プログラムだけでは解決しない事があり、その場合は、手作業のようなことをする必要がある場合があります

データ分析の流れ

2. 処理

- 集計・分析の前処理に処理がしやすいようにデータ形式などの処理をします
- 例えば、欠損データをどう扱うかもその一つです
- 欠損データを「そのデータごと消してしまう」のか「代となる値を代入する」のかを決める必要があります
- 代入データをするのであれば、そのデータをどのようにして代入するのか、そのロジックはどうするのかということも含まれます

データ分析の流れ

3. 集計・可視化

- 分析の目的に応じ、データを集計し、グラフ化します
- データを の基準で集めます

pandasの使い方

pandasの基本的な使い方

- pandasはPythonでデータ 作や分析を行う で重 なライブラリです
- 以下のような を持っています
 - データ 作
 - データ結合と 作
 - データの欠損 処理
 - 時系列データのサポート
 - データの 出

公式サイト : <https://pandas.pydata.org/>

Qiita : <https://qiita.com/tags/pandas>

pandasの基本的な使い方

- pandasはGoogle Colabにはじめから っているのでインストールの必 はないです
- インポートをする必 があります
- 一般的にpandasは「pd」と省^略して扱われます

```
# pandasモジュールをインポートして「pd」と略する  
import pandas as pd
```

pandasの基本的な使い方

- `Series` オブジェクト
 - 一次元のコンテナ
 - コンテナ：オブジェクトの集まりを表現するデータ構造のこと
- `DataFrame` オブジェクト
 - Excelのような表形式のデータを扱います
 - 行や列として一次元のデータ構造（`Series` オブジェクト）の集まりを、表形式のデータ構造（`DataFrame` オブジェクト）で扱います

pandasの基本的な使い方

- DataFrame と Series オブジェクト
- 名前付きデータを扱う一次元データを作成します

```
#一度インポートしたら毎回書くなくてもよい
import pandas as pd
```

```
# 名前付き のリストを元にpandasのSeriesオブジェクトを作成
# Seriesは一系列のデータを扱うためのデータ構造
name_series = pd.Series(['東京', '札幌'])
```

```
# 作成したSeriesオブジェクトを表示
print(name_series)
```


pandasの基本的な使い方

- DataFrame と Series オブジェクト
- 名前と年齢のデータを 加して 1次元データの集まり (DataFrame) を作成します
- 見出しとなる「名前」「年齢」を 加します

```
import pandas as pd

people = pd.DataFrame({
    'name': name_series,
    'age': [22, 28]})
print(people)
```

pandasの基本的な使い方

- DataFrame と Series オブジェクト
- DataFrameは列単位でSeriesオブジェクトとして取り出すことができます

```
import pandas as pd
```

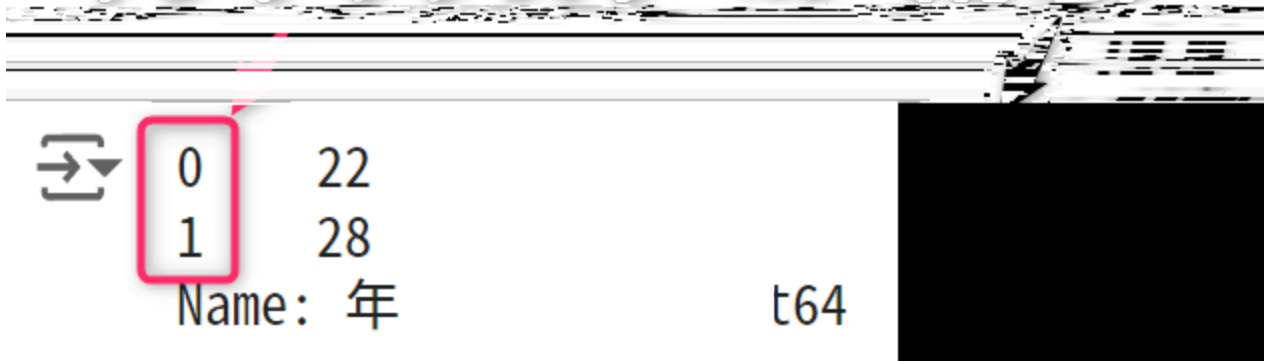
```
# 先に作成されたデータフレーム 'people' の 'name' 列を表示  
print(people['name'])
```

pandasの基本的な使い方

インデックスラベル（行名）

- これまでの結果からわかるように1番左の列に、「0」や「1」の数字が表示されます
- これを「インデックスラベル（行名）」と呼びます
- デフォルトでは「0」からはじまるので注意してください

インデックスラベル（行名）



The screenshot shows a pandas DataFrame with two rows. The index labels '0' and '1' are highlighted with a red box. The first column is labeled 'Name: 年' and the second column is labeled 't64'. The values in the first column are 22 and 28. The values in the second column are obscured by a black box.

0	22	
1	28	

pandasの基本的な使い方

- インデックスラベル（行名）
- `Series` オブジェクトや `DataFrame` オブジェクトの引数「`index`」を指定します
- これによりデフォルトでないインデックスを使用することができます
- `DataFrame`オブジェクトでもインデックスに指定できます
- インデックスは数字だけでなく、文字列を指定することもできます

```
# 前 のリストを元にpandasのSeriesオブジェクトを作成
# ' 前 ' と ' 後 ' という前 のリスト
# インデックスを [10, 20] に 定（デフォルトのインデックスを使わずに指定したインデックスを使用）
name_series = pd.Series([' 前 ', ' 後 '], index = [10, 20])

print(name_series)
```

pandasの基本的な使い方

行インデックス（行番号）

- インデックスラベルとは別に、「行インデックス(行番号)」も指定に指 することができ
ます
- 行インデックスは、インデックスラベルのように別 をつけることができません
- 1行目が「0」となるので 意が必 です

pandasの基本的な使い方

- `loc` と `iloc` によるデータの選択
 - `loc`属性
 - ラベルを 用いて行や列にアクセスするための
 - 行や列のラベルを指定して のデータにアクセスする際に使
 - `iloc`属性
 - インデックスを 用いて行や列にアクセスするための
 - 行や列の位置（インデックス）を指定して のデータにアクセスする際に使

pandasの基本的な使い方

- **loc**と**iloc** によるデータの選択
 - 標準としてDataFrameオブジェクトを作成する

```
# 身長、体重データを含むデータフレームを作成
# 身長、体重という列を持つデータフレーム
# インデックスは ['山田', '佐藤', '高橋', '岡田', '谷川'] に設定
people = pd.DataFrame({
    '身長': [21, 34, 23, 44, 19],
    '体重': [180, 168, 174, 181, 169],
},
    index=['山田', '佐藤', '高橋', '岡田', '谷川'])

print(people)
```

pandasの基本的な使い方

- `loc` と `iloc` によるデータの選択
 - インデックスラベルが「`姓`」のデータを取り出す方法 (`loc` を利用)
 - 行インデックスが「`1`」のデータを取り出す方法 (`iloc` を利用)
 - 同じ結果になります

```
# '姓' というラベルのインデックスに対応する行をデータフレーム 'people' から取り出し、表示  
print(people.loc['姓'])
```

```
# インデックス位置が1 (2番目の行) に対応する行をデータフレーム 'people' から取り出し、表示  
print(people.iloc[1])
```


pandasの基本的な使い方

- `loc` と `iloc` によるデータの選択
 - 必要な列に絞って取り出すことも可能
 - 結果はDataFrameになります

```
# '高橋'という名のインデックスに対応する行から、  
# 特定の列（'姓'と'体高'）を取り出し、表示  
print(people.loc['高橋', ['姓', '体高']])
```

```
# インデックス位置が2（3番目の行）に対応する行から、  
# 特定の列（0番目と2番目の列：'姓'と'体高'）を取り出し、表示  
print(people.iloc[2, [0, 2]])
```

pandasの基本的な使い方

- `loc` と `iloc` によるデータの選択
 - `[start:stop]` 形式で取り出したい部分を指定できます
 - これをスライス記法と呼ぶ
 - 結果は同じになります

```
# インデックスが '高橋' から '三川' までの行を  
# データフレーム 'people' から取り出し、locプロパティを使用して、  
# インデックスラベルで範囲を指定  
print(people.loc['高橋':'三川'])
```

```
# インデックス位置が2から4 (3番目から5番目の行) までの行を  
# データフレーム 'people' から取り出し、ilocプロパティを使用して、  
# 整数位置で範囲を指定  
print(people.iloc[2:5])
```

課題

課題

- Moodleにある「SCfCL-10th-prac.ipynb」ファイルをダウンロードしてColabにアップロードしてください
- 課題が完了したら「File」>「Download」>「Download.ipynb」で「.ipynb」形式でダウンロードしてください
- ダウンロードした**.ipynb**ファイルと作成した「Prac09」フォルダを圧縮してMoodleに出してください
- 出期限は **6月27日(木) 20時まで** です