



Pandas – Outros conceitos importantes

Luiz Celso Gomes-Jr

Inicializando DataFrames a partir de dicionários

Uma das formas mais fáceis de se inicializar um DataFrame é usando dicionários. Uma lista de dicionários pode ser usada para compor as linhas do DataFrame como no exemplo abaixo:

```
import pandas as pd

ap1 = {'endereco': 'Av V. Guarapuava, 1000',
       'area': 45, 'aluguel': 800}
ap2 = {'endereco': 'Av Sete de Setembro, 170',
       'area': 53, 'aluguel': 950}

apartamentos = [ap1, ap2]
df = pd.DataFrame(apartamentos)
df
```

	endereco	area	aluguel
0	Av V. Guarapuava, 1000	45	800
1	Av Sete de Setembro, 170	53	950

Inicializando DataFrames a partir de dicionários

Podemos usar um dicionário para adicionar uma nova linha a um DataFrame:

```
ap3 = {'endereco': 'Av Sete de Setembro, 830',  
       'area': 35, 'aluguel': 850}  
  
df = df.append([ap3])  
df
```

	endereco	area	aluguel
0	Av V. Guarapuava, 1000	45	800
1	Av Sete de Setembro, 170	53	950
0	Av Sete de Setembro, 830	35	850

Iterando sobre um DataFrame

Podemos usar o comando `for` para analisar um DataFrame linha por linha. Para isto usamos o método **`iterrows`**. O exemplo abaixo calcula a média dos valores dos aluguéis:

	endereco	area	aluguel
0	Av V. Guarapuava, 1000	45	800
1	Av Sete de Setembro, 170	53	950
0	Av Sete de Setembro, 830	35	850

```
total = 0
for indice, apartamento in df.iterrows():
    print("Aluguel {}: {}".format(indice, apartamento['aluguel']))
    total = total + apartamento['aluguel']

media = total/len(df)
print("Média:", media)
```

```
Aluguel Av V. Guarapuava, 1000: 800
Aluguel Av Sete de Setembro, 170: 950
Aluguel Av Sete de Setembro, 830: 850
Média: 866.6666666666666
```

Séries

Um bloco básico na construção de um DataFrame é a Série (**Series**). Cada coluna de um DataFrame é uma Series. Uma Series contém:

- um nome
- um índice
- uma lista (array) de valores

Método apply()

O método `apply()` aplica uma função a colunas ou linhas de um `DataFrame`. No exemplo abaixo definimos uma função que conta o número de valores nulos em uma coluna. Esta função é então aplicada no `DataFrame` exibido na célula anterior.

	endereco	area	aluguel
0	Av V. Guarapuava, 1000	45.0	800
1	Av Sete de Setembro, 170	53.0	950
2	Av Sete de Setembro, 830	35.0	850
3	Av Sete de Setembro, 730	NaN	775

```
def conta_nan(coluna):  
    return coluna.isna().sum()  
  
df.apply(conta_nan)
```

```
endereco      0  
area          1  
aluguel       0  
dtype: int64
```

Método apply() - Linhas

Uma função também pode ser aplicada sobre valores das linhas de um DataFrame. No caso abaixo, definimos a função `aumenta_aluguel()` e a aplicamos às linhas do DataFrame. Para aplicar a função às linhas, definimos o parâmetro **axis=1**.

```
def aumenta_aluguel(linha):  
    return linha['aluguel'] + linha['aluguel']*0.1  
  
df['novo aluguel'] = df.apply(aumenta_aluguel, axis=1)  
df
```

	endereco	area	aluguel	novo aluguel
0	Av V. Guarapuava, 1000	45.0	800	880.0
1	Av Sete de Setembro, 170	53.0	950	1045.0
2	Av Sete de Setembro, 830	35.0	850	935.0
3	Av Sete de Setembro, 730	NaN	775	852.5

Método apply() - Função lambda

O Python possui o conceito de "**função anônima**", ou "**função lambda**". Este recurso é útil para especificar uma função simples sem precisar defini-la. Abaixo construímos uma função lambda que retorna apenas a parte do número do endereço de um apartamento.

```
df['num ap'] = df.apply(lambda x: x['endereco'].split(',')[1],  
                        axis=1)  
df
```

	endereco	area	aluguel	novo aluguel	num ap
0	Av V. Guarapuava, 1000	45.0	800	880.0	1000
1	Av Sete de Setembro, 170	53.0	950	1045.0	170
2	Av Sete de Setembro, 830	35.0	850	935.0	830
3	Av Sete de Setembro, 730	NaN	775	852.5	730

Método apply() - Função lambda

As células de código abaixo produzem o mesmo resultado:

```
df['num ap'] = df.apply(lambda x: x['endereco'].split(',')[1],  
                        axis=1)  
df
```

```
def obtem_numero(linha):  
    numero = linha['endereco'].split(',')[1]  
    return numero  
  
df['numero ap.'] = df.apply(obtem_numero, axis=1)  
df
```

Tratando DataFrames grandes

Muitas vezes precisamos processar dados que não cabem na memória do computador ou que demandam procedimentos complexos que deixam o processamento lento. Estratégias possíveis:

- Amostragem do DataFrame
- Amostragem diretamente no CSV
- Tratamento de pedaços (chunks)

Vejam exemplos nos tutoriais.

Exercícios!

- Revise o conteúdo e faça os exercícios do notebook:
04b-Outros Conceitos - Pandas.ipynb