PYTHON ACADEMY

MANIPULAÇÃO DE DADOS COM DATAFRAMES DO PANDAS

Complemento para o ebook de pandas básico, vamos aprender mais sobre Pandas Dataframes! A estrutura tabular que todo cientista de dados vai ficar careca de saber (eu fiquei, literalmente :O). Cola na Python Academy que você vai sair um cientista de dados!

PYTHONACADEMY.COM.BR

Pandas, Pandas mais uma vez!

Agora que você é craque em Séries do Pandas, vamos ao complemento dessa dupla inseparável: Séries e Dataframes Pandas!

Definição de DataFrame

Depois de aprender tudo sobre Séries, chegou a hora de aprender tudo sobre **Dataframes**.

Como assim? Não sabe de Séries?? Pode parar por aqui então e ler nosso último post sobre as Séries do Pandas.

Como a gente é bonzinho, ele tá aqui ó.

Um jeito fácil de entender: Séries são colunas, Dataframes são tabelas!

Na verdade, um Dataframe é formado por um conjunto de séries, cada uma delas sendo uma coluna da 'tabela'.

Para criar um Dataframe, é simples! Só precisamos instanciar um objeto a partir da classe `DataFrame` do Pandas:

```
>>> import pandas as pd
>>> df = pd.DataFrame([200, 350, 550])
>>> df
     0
0  200
1  350
2  550
```

Exemplo simples, mas já mostra que alguns componentes de um Dataframe.

Assim como as séries, Dataframes possuem índices, mas também possui colunas, cada uma delas com um nome.

Como não passamos explicitamente o nome das colunas, o pandas cria... adivinhem: um `RangeIndex` de 0 até o número de colunas menos um.

Mas, vamos devagar. No exemplo anterior criamos um Dataframe a partir de uma lista.

Outra forma é criar a partir de um dicionário, onde a chave é o nome da coluna e o valor, uma lista de elementos:

```
>>> import pandas as pd
>>> df = pd.DataFrame({'calorias':[200, 350, 550], 'gordura (%)':[0, 6,
        15]})
>>> df
   calorias  gordura (%)
0       200            0
1       350            6
2       550           15
```

Agora, podemos alterar os índices para algo mais mnemônico, que facilite nossa vida:

```
>>> import pandas as pd
>>> df = pd.DataFrame({'calorias':[200, 350, 550], 'gordura (%)':[0, 6,
        15]}, index=['banana', 'prato feito', 'big mac'])
>>> df
             calorias  gordura (%)
banana            200            0
prato feito       350            6
big mac           550           15
```

Vale lembrar que tudo que funcionava com o índice das séries também vale para os Dataframes!

Agora vamos aprender a acessar os dados de um Dataframe.

Acesso aos Dados

Essa parte é um capítulo à parte de teremos um post detalhando várias formas de acessar os dados de um Dataframe.

Alguns dos métodos, como `loc`, `iloc`, `at` e `iat` merecem bastante detalhe e exemplos, mas traremos isso pra vocês em breve!

A verdade é que existem MUITAS formas de acessar os dados de um Dataframe, mas não vamos deixar os padawans na mão: a Python Academy sempre ao serviços de vocês ❤️

Uma forma simples de acessar todos os valores de uma coluna de um Dataframe é utilizar o `df[coluna]`, dessa forma:

```
>>> df['gordura (%)']
banana          0
prato feito     6
big mac        15
Name: gordura (%), dtype: int64
```

Que estranho?! Parece uma série!

Certíssimo!

Lembra que dissemos que uma coluna de um Dataframe é uma série? Vamos testar:

```
>>> type(df['gordura (%)'])
<class 'pandas.core.series.Series'>
```

E para acessar os valores de uma linha?

Vamos recorrer aos índices!

```
>>> df = pd.DataFrame({'calorias':[200, 350, 550], 'gordura (%)':[0, 6,
        15]})
>>> df
banana          0
prato feito     6
big mac        15
Name: gordura (%), dtype: int64
```

Nesse caso, como não criamos o índice, ele é numérico e corresponde à posição de cada linha.

Uma forma é acessar como acessamos elementos de uma lista, utilizando *slice*:

```
>>> df[1:2]
   calorias  gordura (%)
1       350            6
```

> 💡 *Estou desenvolvendo o **DevBook**, uma plataforma que usa IA para gerar ebooks técnicos profissionais. Não deixe de conferir clicando no botão abaixo!*

Outros tipos de acesso (`loc`)

Adiantando um assunto importante, podemos também utilizar o método `loc`, que acessa por meio do índice:

```
>>> df.loc[0]
calorias       200
gordura (%)      0
Name: 0, dtype: int64
```

Percebam que a apresentação dos dois resultados é bem diferente!

Isso porque utilizando *slice*, o resultado é um novo Dataframe com as linhas selecionadas.

No segundo caso, o `loc` retorna uma série com os valores da linha selecionada.

Outro assunto que traremos pra vocês em mais detalhes: quando temos Dataframes, Séries e Cópias como resultados.

Séries são a base de tudo

Peraí, peraí: a Série não era uma coluna, agora é uma linha?

Ótima pergunta, Pythonista!

Olha que legal, o Pandas retorna tanto uma coluna como uma linha completa (com o respectivo índice) como uma Série!

Uma série "coluna" possui um nome (o título da caluna), os valores (elementos da coluna) bem como o tipo.

De forma similiar, uma série “linha” possui um nome (o índice que referencia a linha), os valores (elementos de uma linha) bem como tipo.

As séries são tão flexíveis e são uma mão na roda para representar os dados, não é à toa que os Cientistas de Dados utilizam Dataframes sempre que possível.

Conclusão

É isso, galera! Esse post + post de Séries dão uma boa base para conhecer a base de Pandas!

Daqui pra frente deixaremos de ser Padawans para sermos Mestres Jedi da Ciência de Dados.

Teremos outros tantos posts, com detalhamento do acesso aos dados com pandas, índices hierárquicos, tratando grandes volumes de dados e muito mais!

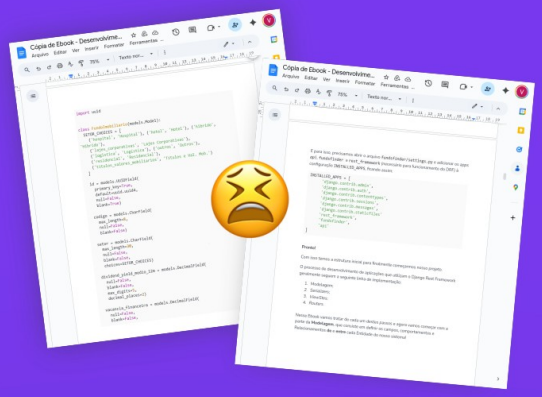
Até a próxima!

Não se esqueça de conferir!

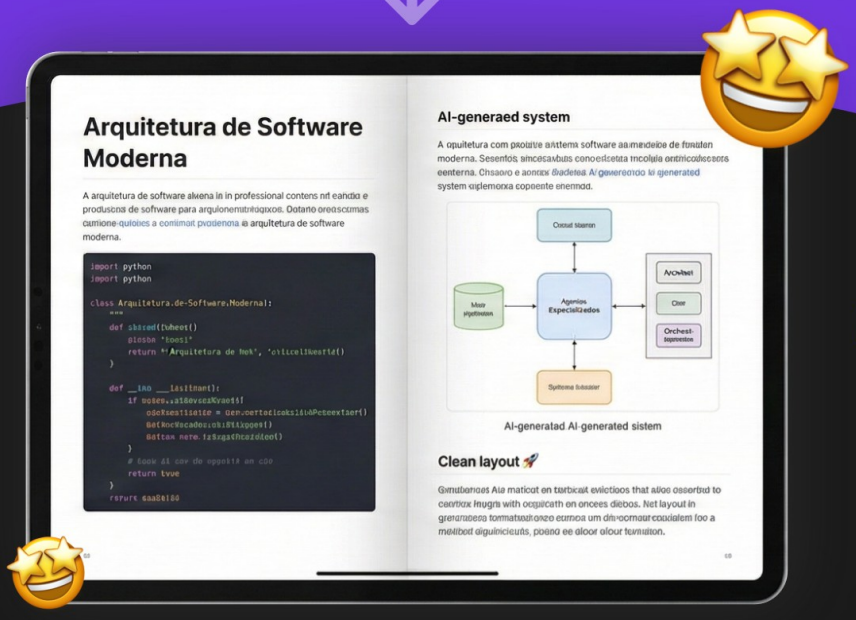
DevBook

Crie Ebooks técnicos em minutos com IA

Conheça a 1ª IA Especializada na criação de Ebooks **com código**!



Chega de formatar código no Google Docs



Syntax Highlight

Infográficos feitos por IA

Adicione Banners Promocionais

Deixe que nossa IA faça o trabalho pesado

Edite em Markdown em Tempo Real

TESTE AGORA

PRIMEIRO CAPÍTULO 100% GRÁTIS