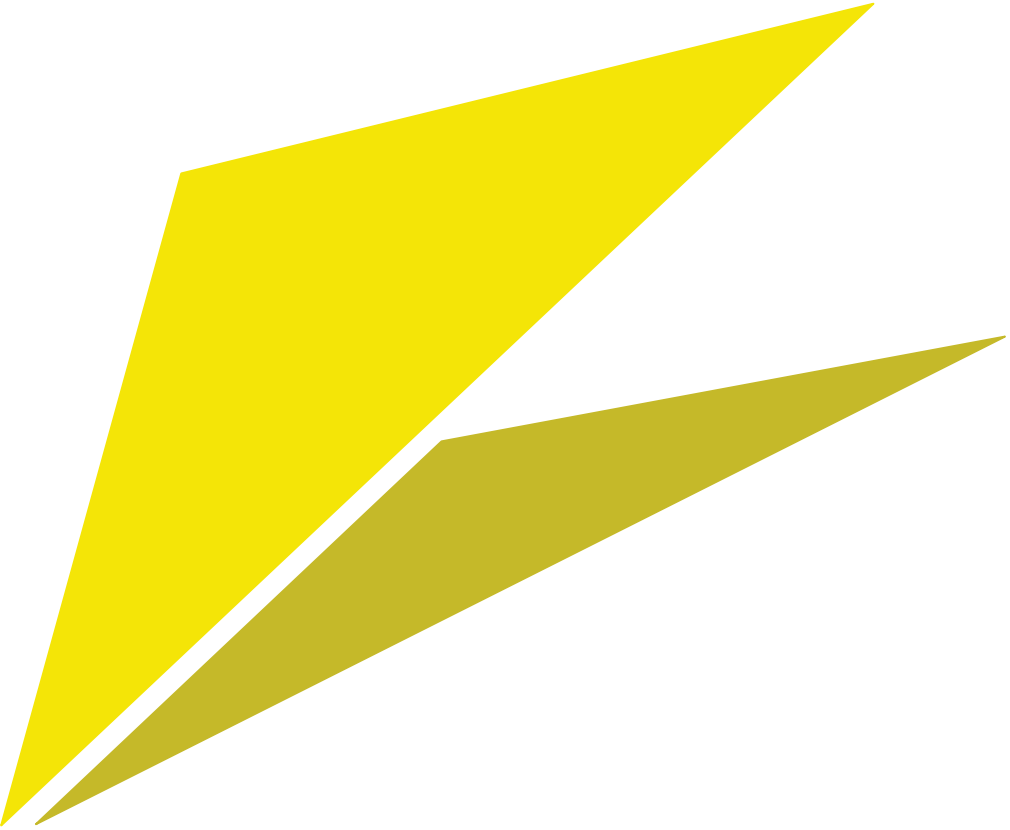


Data Manipulation with Pandas

Grupo 01 - FEA.dev

Sumário

Transforming DataFrames - Introdução

Pandas é uma biblioteca do Python para manipulação e visualização de dados.

Importação da Biblioteca '*Pandas*'

import panda **as** pd

nome_variavel1 = pd.DataFrame(nome_variavel2)

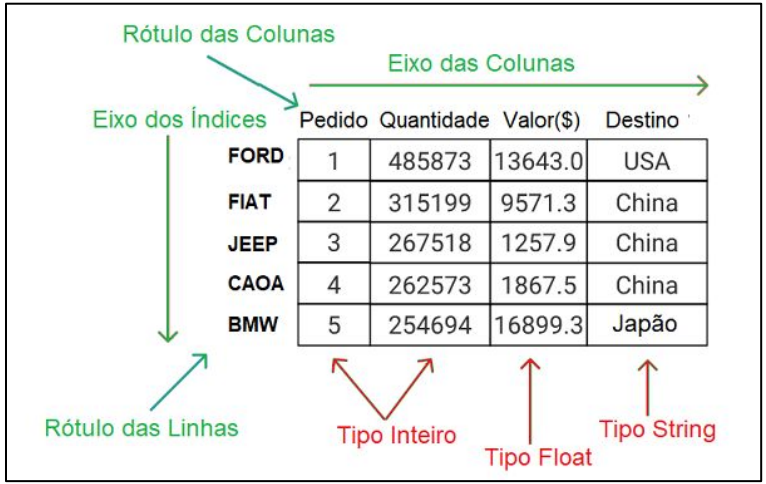
Alguns Métodos do Pandas

.head()
.info()
.shape
.sort_values
    .sort_values + ascending=[True]

Transforming DataFrames - DataFrame

Data Frame

DataFrame - Teoria



DataFrame - Prática

```
In [15]: print(dataframe)
```

```
    Nome  Nota Aprovado
0 Ricardo   4.0      Não
1   Pedro   7.0      Sim
2 Roberto   5.5      Não
3  Carlos   9.0      Sim
```

```
In [ ]:
```

Transforming DataFrames - Exemplos dos Métodos

Alguns exemplos sobre a utilização dos métodos da biblioteca Pandas na linguagem de programação Python

Estrutura para os métodos

print(nome_variavel.MÉTODO)

ex:

print(nome_variavel.shape)

```
import pandas as pd
alunos = {'Nome':['Ricardo', 'Pedro', 'Roberto', 'Carlos'],
          'Nota':[4, 7, 5.5, 9,],
          'Aprovado':['Não','Sim','Não','Sim']}
```

```
dataframe = pd.DataFrame(alunos)
```

```
print(dataframe.shape)
```

```
(4, 3)
```

Aggregating DataFrames

1. Síntese Estatística

Funções: `.mean(),.median(),.max()`
`.std(),.quantile()`

Sintaxe: `df["coluna"].mean()`

2. Contagem

Função: .drop_duplicates(`subset="name`)

	Country	Inflation Rate
0	Brazil	10.06
1	Brazil	10.06
2	Chile	4.19
3	Argentina	64.00
4	Uruguay	7.51

```
: df1 = df.drop_duplicates()
display(df1)
```

	Country	Inflation Rate
0	Brazil	10.06
2	Chile	4.19
3	Argentina	64.00
4	Uruguay	7.51

Aggregating DataFrames

1. Contagem (2)

Funções: `.value_counts()`

- `sort=True`

2. Síntese Estatística Agrupada

dataframe.groupby(“column”)[“column2”].mean

```
>>> df = pd.DataFrame({'Animal': ['Falcon', 'Falcon',
...                               'Parrot', 'Parrot'],
...                    'Max Speed': [380., 370., 24., 26.]})
>>> df
   Animal  Max Speed
0  Falcon      380.0
1  Falcon      370.0
2  Parrot       24.0
3  Parrot       26.0
>>> df.groupby(['Animal']).mean()
        Max Speed
Animal
Falcon      375.0
Parrot       25.0
```

Selecionando dados

1. Loc

Função: `.loc[]`

Sintática: `df.loc[“nome linha”,”nome coluna”]`

Intervalo de linhas

```
animal.loc[(“Aves”, “Arara”):(“Mamíferos”,”Macaco”)]
```

Intervalo de colunas

```
animal.loc[:, “Classe”:”Espécie”]
```

Ambos

```
animal.loc[(“Aves”, “Arara”):(“Mamíferos”,”Macaco”),“Classe”:”Espécie”]
```

2. Iloc

Função: `.iloc[]`

Sintática: `df.iloc[linha,coluna]`

Intervalo de linhas

```
animal.iloc[4:8]
```

Intervalo de colunas

```
animal.iloc[:,3:7]
```

Ambos

```
animal.loc[4:8,3:7]
```

Índices

1. Escolher coluna como índice

Função: `set_index()`

Sintática: `df.set_index("coluna")`

Obs: podemos definir mais de uma coluna como índice `df.set_index(["coluna1", "coluna2")`

2. Ordenando

Função: `sort_index()`

Sintática: `df.sort_index(level = ["coluna1","coluna2"], ascending =[True,False])`

- Índice: coluna que será referência para o dataframe (primeira)
- Facilita a escolha de dados

ex:

Sem index

```
animal[animal["Espécie"].isin(["Gato","Cachorro"])]
```

Com index

```
animal.set_index("Espécie")
animal.loc[["Gato",'"Cachorro"]]
```

Tabela dinâmica

Função: `pivot_table()`

Sintática: `df.pivot_table(“ dados linhas”,index = “”,colums =””)`

Todas as funções utilizadas para Dataframe podem ser usadas em tabelas dinâmicas

```
animal1 = animal.pivot_table
(“peso”,index = “Espécie”, columns
= “peso”)
```

Loc

```
animal1.loc[“Gato”:”Cachorro”]
```

Média

```
animal1.mean(axis = “columns”)
```

Creating and Visualizing DataFrames - Visualizando os dados

HISTOGRAMAS

```
dados['coluna'].hist(bins=5)
Plt.show
```

EM LINHA

```
seu_dataframe.plot(x = 'eixo x', y = 'eixo y', kind = 'line')
```

EM BARRA

```
Valor_medio = df.groupby('variável categórica')['variável numérica'].mean()
Valor_medio.plot(kind = 'bar', title = 'título')
Plt.show
```

SCATTER PLOTS

```
Dataframe.plot(x = 'eixo x', y = 'eixo x', kind = 'scatter')
```

Creating and Visualizing DataFrames - Valores Faltando

.isna() mostra se o valor está ausente ou não

(True está ausente, False está presente)

.isna().any() `mostra se há algum valor ausente nas colunas`

.isna().sum() mostra a quantidade de valores ausentes nas colunas

.dropna() remove as linhas com valores ausentes

.fillna(número) todos os NaNs serão substituídos pelo valor escrito no parênteses

Creating and Visualizing DataFrames - Criando DataFrames

Dois métodos principais:

lista de dicionário

ex: Lista_de_dicionários = [{dicionario1}, {dicionario2}, {dicionario3}]

A chave de cada dicionário se refere a coluna.

Ex: {'name':'Alissa', idade:'18'}

A coluna vai ser 'name' e o valor 'Alissa'; coluna 'idade', valor '18'

dicionário de listas

ex:

Dicionario_de_listas = {'nome':['Alissa', 'Arthur'], 'curso':['Cont','Adm']}

Tabela = pd.DataFrame(dicionário_de_listas)

- 'nome' será o título da coluna e 'Alissa' e 'Arthur' serão os valores; curso será outra coluna.

Creating and Visualizing DataFrames - Lendo e Escrevendo CSVs

- Arquivos com valores separados por vírgulas

- Separando

pd. read_csv('arquivo.csv')

- Transformando em CSV

arquivo.to_csv('novo_arquivo.csv')