

Pandas: Guia Básico

Do básico ao essencial

Claudio Scheer

8 de agosto de 2025

pandas: o que é?

- Biblioteca de código aberto para **análise e manipulação** de dados
- Estruturas principais: **Series** (1D) e **DataFrame** (2D)
- Integração com [Matplotlib](#) e o ecossistema Python
- Leitura/escrita em vários formatos: CSV, Excel, JSON, SQL
- Ferramentas para limpeza, seleção, agrupamento e agregação

- ❶ O que é Pandas?
- ❷ Instalação e Importação
- ❸ Estruturas: Series e DataFrame
- ❹ Operações Básicas
- ❺ Leitura e Escrita (CSV/Excel/JSON)
- ❻ Seleção e Filtragem
- ❼ Limpeza de Dados
- ❽ Operações com Dados
- ❾ Agrupamento e Agregação
- ❿ Visualização
- ⓫ Merge/Join/Concat
- ⓬ Datas
- ⓭ Dicas e Recursos
- ⓮ Exercícios

Instalação e Importação

```
1 # via pip
2 pip install pandas
3
4 # via conda
5 conda install pandas
```

```
1 import pandas as pd
```

Series (1D)

Ideia

Uma **Series** é como uma coluna de valores com um índice.

```
1 # Series simples
2 numeros = pd.Series([10, 20, 30, 40, 50])
3 print(numeros)
4
5 # índice personalizado
6 frutas = pd.Series(['Maçã', 'Banana', 'Laranja'], index=['a','b','c'])
7 print(frutas)
8
9 # a partir de dicionário
10 vendas = pd.Series({'Jan': 1000, 'Fev': 1500, 'Mar': 1200})
11 print(vendas['Jan']) # 1000
```

DataFrame (2D)

Ideia

Um **DataFrame** é como uma planilha: linhas e colunas com rótulos.

```
1 dados = {
2     'Nome': ['Ana', 'João', 'Maria', 'Pedro'],
3     'Idade': [25, 30, 28, 35],
4     'Cidade': ['SP', 'RJ', 'BH', 'SP'],
5     'Salario': [5000, 6000, 5500, 7000]
6 }
7 df = pd.DataFrame(dados)
8 print(df.head())
9
10 # a partir de lista de listas
11 dados_lista = [['Ana', 25, 'SP'], ['João', 30, 'RJ']]
12 df2 = pd.DataFrame(dados_lista, columns=['Nome', 'Idade', 'Cidade'])
```

Operações Básicas

```
1 df.head()      # primeiras linhas
2 df.tail()      # últimas linhas
3 df.info()      # tipos e memória
4 df.shape       # (linhas, colunas)
5 df.columns     # nomes das colunas
6 df.index       # índice das linhas
7 df.dtypes      # tipos por coluna
```

```
1 df_num = pd.DataFrame({
2     'Vendas': [100, 150, 200, 180, 220],
3     'Custos': [ 50, 70, 90, 85, 100]
4 })
5 df_num.describe() # contagem, média, desvio, min, quartis, max
6 df_num.mean()     # médias por coluna
```

```
1 # leitura
2 df_csv = pd.read_csv('arquivo.csv')
3 df_excel = pd.read_excel('arquivo.xlsx')
4 df_json = pd.read_json('arquivo.json')
5
6 # escrita
7 df.to_csv('saida.csv', index=False)
8 df.to_excel('saida.xlsx', index=False)
9 df.to_json('saida.json', orient='records')
```


Seleção e Filtragem

```
1 # colunas
2 idades = df['Idade']
3 subconjunto = df[['Nome', 'Salario']]
4
5 # linhas por posição / rótulo
6 df.iloc[0]          # primeira linha
7 df.loc[0:2]         # linhas 0 a 2 (inclusivo)
8
9 # por condição
10 maiores_30 = df[df['Idade'] > 30]
11 sp_jovens = df[(df['Cidade'] == 'SP') & (df['Idade'] < 30)]
12
13 # avançado
14 df.loc[0:2, ['Nome', 'Idade']]
15 df.query('Idade > 25 and Salario < 6000')
```

Limpeza de Dados

```
1 df_faltantes = pd.DataFrame({
2     'A': [1, 2, float('nan'), 4],
3     'B': [5, float('nan'), float('nan'), 8]
4 })
5
6 df_faltantes.isnull().sum()    # contar NaN por coluna
7 df_faltantes.dropna()         # remover linhas com NaN
8 df_faltantes.dropna(axis=1)   # remover colunas com NaN
9
10 # preencher
11 df_faltantes.fillna(0)
12 df_faltantes.fillna(df_faltantes.mean(numeric_only=True))
13 df_faltantes.fillna(method='ffill')
```

Duplicatas, Renomear e Tipos

```
1 df_dup = pd.DataFrame({'Nome': ['Ana', 'João', 'Ana'], 'Idade': [25, 30, 25]})
2 df_dup.duplicated()
3 df_dup.drop_duplicates()
4
5 # renomear
6 df_renomeado = df.rename(columns={'Nome': 'NomeCompleto', 'Idade': 'Anos'})
7
8 # tipos
9 df['Idade'] = df['Idade'].astype(float)
10 df['Salario'] = pd.to_numeric(df['Salario'], errors='coerce')
```

Operações com Dados

```
1 # novas colunas
2 df['Bonus'] = df['Salario'] * 0.10
3 df['SalarioTotal'] = df['Salario'] + df['Bonus']
4
5 # coluna condicional
6 df['Senioridade'] = df['Idade'].apply(lambda x: 'Sênior' if x >= 30 else
    'Júnior')
7
8 # remover coluna
9 df = df.drop('Bonus', axis=1) # ou: del df['Bonus']
```

Ordenação

```
1 # ordenação
2 df.sort_values('Salario')
3 df.sort_values('Salario', ascending=False)
4 df.sort_values(['Cidade', 'Idade'], ascending=[True, False])
5 df.sort_index()
```

Agrupamento e Agregação

```
1 df_vendas = pd.DataFrame({
2     'Vendedor': ['Ana', 'João', 'Ana', 'João', 'Maria', 'Maria'],
3     'Produto': ['A', 'A', 'B', 'B', 'A', 'B'],
4     'Quantidade': [10, 15, 20, 25, 30, 35],
5     'Valor': [100, 150, 200, 250, 300, 350]
6 })
7
8 g = df_vendas.groupby('Vendedor')
9 g.sum()
10 g.mean()
11
12 # múltiplas agregações
13 agg = df_vendas.groupby('Vendedor').agg({'Quantidade': 'sum',
14     'Valor': ['mean', 'sum']})
```

Pivot Table

```
1 pivot = df_vendas.pivot_table(  
2     values='Valor', index='Vendedor', columns='Produto',  
3     aggfunc='sum', fill_value=0  
4 )  
5 print(pivot)
```

Visualização (básico)

```
1 import matplotlib.pyplot as plt
2
3 df_plot = pd.DataFrame({
4     'Mes': ['Jan', 'Fev', 'Mar', 'Abr', 'Mai'],
5     'Vendas': [100, 120, 140, 110, 160],
6     'Custos': [80, 90, 100, 85, 110]
7 })
8
9 df_plot.plot(x='Mes', y='Vendas', kind='line')
10 plt.title('Vendas Mensais')
11 plt.show()
12
13 df_plot.plot(x='Mes', y=['Vendas', 'Custos'], kind='bar')
14 plt.title('Vendas vs Custos')
15 plt.show()
```


Merge, Join e Concat

```
1 # concatenação
2 df1 = pd.DataFrame({'A':[1,2], 'B':[3,4]})
3 df2 = pd.DataFrame({'A':[5,6], 'B':[7,8]})
4 pd.concat([df1, df2], ignore_index=True)
5
6 # junção (merge)
7 esq = pd.DataFrame({'ID':[1,2,3], 'Nome':['Ana','João','Maria']})
8 dir = pd.DataFrame({'ID':[1,2,4], 'Salario':[5000,6000,7000]})
9
10 pd.merge(esq, dir, on='ID') # inner
11 pd.merge(esq, dir, on='ID', how='left')
12 pd.merge(esq, dir, on='ID', how='outer')
```

```
1  datas = pd.date_range('2024-01-01', periods=10, freq='D')
2  df_tempo = pd.DataFrame({'Data': datas, 'Valor': range(100, 110)})
3  df_tempo.set_index('Data', inplace=True)
4
5  df_tempo.resample('W').mean()      # média semanal
6  df_tempo.rolling(window=3).mean() # média móvel
7
8  df_tempo['Ano'] = df_tempo.index.year
9  df_tempo['Mes'] = df_tempo.index.month
```

- Prefira **vetorização** a loops explícitos
- Use `query()` para filtros legíveis
- Para arquivos grandes, leia em *chunks*
- Use tipo `category` para colunas com poucos valores distintos

```
1 # filtro claro (bom)
2 df.query('col1 > 10 and col2 < 20')
```

- Documentação Oficial: pandas.pydata.org
- 10 Minutes to Pandas: tutorial rápido
- Cheat Sheet: referência rápida
- Kaggle Learn: cursos práticos

- ❶ Crie um DataFrame com 5 produtos (nome, preço, quantidade) e calcule o valor total do estoque.
- ❷ Leia um CSV, limpe valores faltantes, agrupe por categoria e crie um gráfico de barras.
- ❸ Faça merge de dois DataFrames, crie uma tabela dinâmica (pivot) e exporte para Excel.

Obrigado!

Perguntas?