Pandas: Guia Básico

Do básico ao essencial

Claudio Scheer

9 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

Índice

Instalação e Importação

```
# via pip
pip install pandas

# via conda
conda install pandas
```

```
import pandas as pd
```

Series (1D)

Ideia

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

```
# Series simples
numeros = pd.Series([10, 20, 30, 40, 50])
print(numeros)

# índice personalizado
frutas = pd.Series(['Maçã', 'Banana', 'Laranja'], index=['a','b','c'])
print(frutas)

# a partir de dicionário
vendas = pd.Series({'Jan': 1000, 'Fev': 1500, 'Mar': 1200})
print(vendas['Jan'])  # 1000
```

DataFrame (2D)

Ideia

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

```
dados = {
    'Nome': ['Ana', 'João', 'Maria', 'Pedro'],
    'Idade': [25, 30, 28, 35],
    'Cidade': ['SP', 'RJ', 'BH', 'SP'],
    'Salario': [5000, 6000, 5500, 7000]
}
df = pd.DataFrame(dados)  # criar DataFrame
print(df.head())          # visualizar primeiras linhas

# a partir de lista de listas
dados_lista = [['Ana', 25, 'SP'], ['João', 30, 'RJ']]
df2 = pd.DataFrame(dados_lista, columns=['Nome','Idade','Cidade'])  #
    DataFrame com colunas
```

Operações Básicas

```
df.head()      # primeiras linhas
df.tail()      # últimas linhas
df.info()      # tipos e memória
df.shape       # (linhas, colunas)
df.columns     # nomes das colunas
df.index       # índice das linhas
df.dtypes      # tipos por coluna
```

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

Leitura e Escrita

```python
# leitura
df_csv   = pd.read_csv('arquivo.csv')
df_excel = pd.read_excel('arquivo.xlsx')
df_json  = pd.read_json('arquivo.json')

# escrita
df.to_csv('saida.csv', index=False)
df.to_excel('saida.xlsx', index=False)
df.to_json('saida.json', orient='records')
```

Seleção e Filtragem

```python
# colunas
idades = df['Idade']
subconjunto = df[['Nome','Salario']]

# linhas por posição / rótulo
df.iloc[0]       # primeira linha
df.loc[0:2]      # linhas 0 a 2 (inclusivo)

# por condição
maiores_30 = df[df['Idade'] > 30]
sp_jovens = df[(df['Cidade'] == 'SP') & (df['Idade'] < 30)]

# avançado
df.loc[0:2, ['Nome','Idade']]
df.query('Idade > 25 and Salario < 6000')
```

Limpeza de Dados

```
df_faltantes = pd.DataFrame({
  'A':[1,2,float('nan'),4],
  'B':[5,float('nan'),float('nan'),8]
})

df_faltantes.isnull().sum()   # contar NaN por coluna
df_faltantes.dropna()         # remover linhas com NaN
df_faltantes.dropna(axis=1)   # remover colunas com NaN

# preencher
df_faltantes.fillna(0)
df_faltantes.fillna(df_faltantes.mean(numeric_only=True))
df_faltantes.fillna(method='ffill')
```

Duplicatas, Renomear e Tipos

```python
df_dup = pd.DataFrame({'Nome':['Ana','João','Ana'], 'Idade':[25,30,25]})  # dados com repetição
df_dup.duplicated()    # identificar duplicatas (True/False)
df_dup.drop_duplicates()  # remover duplicatas

# renomear
df_renomeado = df.rename(columns={'Nome':'NomeCompleto', 'Idade':'Anos'})

# tipos
df['Idade'] = df['Idade'].astype(float)
df['Salario'] = pd.to_numeric(df['Salario'], errors='coerce')
```

Operações com Dados

```python
# novas colunas
df['Bonus'] = df['Salario'] * 0.10
df['SalarioTotal'] = df['Salario'] + df['Bonus']

# coluna condicional
df['Senioridade'] = df['Idade'].apply(lambda x: 'Sênior' if x >= 30 else
    'Júnior')

# remover coluna
df = df.drop('Bonus', axis=1)  # ou: del df['Bonus']
```

Ordenação

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

Agrupamento e Agregação

```python
df_vendas = pd.DataFrame({  # exemplo de vendas
  'Vendedor':['Ana','João','Ana','João','Maria','Maria'],
  'Produto':['A','A','B','B','A','B'],
  'Quantidade':[10,15,20,25,30,35],
  'Valor':[100,150,200,250,300,350]
})

g = df_vendas.groupby('Vendedor')  # agrupar por vendedor
g.sum()    # somar por grupo
g.mean()   # média por grupo

# múltiplas agregações
agg = df_vendas.groupby('Vendedor').agg({'Quantidade':'sum',
    'Valor':['mean','sum']})  # agregações
```

Pivot Table

```
pivot = df_vendas.pivot_table(  # tabela dinâmica (pivot)
  values='Valor', index='Vendedor', columns='Produto',
  aggfunc='sum', fill_value=0
)
print(pivot)  # visualizar pivot
```

Visualização (básico)

```
import matplotlib.pyplot as plt

df_plot = pd.DataFrame({  # dados para plotagem
  'Mes':['Jan','Fev','Mar','Abr','Mai'],
  'Vendas':[100,120,140,110,160],
  'Custos':[ 80, 90,100, 85,110]
})

df_plot.plot(x='Mes', y='Vendas', kind='line')  # gráfico de linha
plt.title('Vendas Mensais')
plt.show()  # mostrar gráfico

df_plot.plot(x='Mes', y=['Vendas','Custos'], kind='bar')  # gráfico de barras
plt.title('Vendas vs Custos')
plt.show()  # mostrar gráfico
```

Merge, Join e Concat

```python
# concatenação
df1 = pd.DataFrame({'A':[1,2], 'B':[3,4]})  # primeiro DataFrame
df2 = pd.DataFrame({'A':[5,6], 'B':[7,8]})  # segundo DataFrame
pd.concat([df1, df2], ignore_index=True)     # concatenar linhas

# junção (merge)
esq = pd.DataFrame({'ID':[1,2,3], 'Nome':['Ana','João','Maria']})  # tabela à esquerda
dir = pd.DataFrame({'ID':[1,2,4], 'Salario':[5000,6000,7000]})     # tabela à direita

pd.merge(esq, dir, on='ID')            # inner join
pd.merge(esq, dir, on='ID', how='left')  # left join
pd.merge(esq, dir, on='ID', how='outer') # outer join
```

Trabalhando com Datas

```python
datas = pd.date_range('2024-01-01', periods=10, freq='D')  # criar datas diárias
df_tempo = pd.DataFrame({'Data': datas, 'Valor': range(100, 110)})  # DataFrame temporal
df_tempo.set_index('Data', inplace=True)  # usar Data como índice

df_tempo.resample('W').mean()     # média semanal
df_tempo.rolling(window=3).mean() # média móvel (3 períodos)

df_tempo['Ano'] = df_tempo.index.year   # extrair ano
df_tempo['Mes'] = df_tempo.index.month  # extrair mês
```

Dicas Rápidas

- 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

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

Recursos Úteis

- Documentação Oficial: pandas.pydata.org (https://pandas.pydata.org)
- 10 Minutes to Pandas: tutorial rápido (https://pandas.pydata.org/pandas-docs/stable/user_guide/10min.html)
- Cheat Sheet: referência rápida (https://pandas.pydata.org/Pandas_Cheat_Sheet.pdf)
- Kaggle Learn: cursos práticos (https://www.kaggle.com/learn/pandas)

Exercícios Práticos

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

Obrigado!

Perguntas?