AULA 14 · PROGRAMAÇÃO EM PYTHON

Pandas

Análise de dados em Python

Da planilha à análise — em uma linha de código.

CEFET-MG · Engenharia · Disciplina de Programação em Python

MOTIVAÇÃO

Por que aprender Pandas?

Dados são o novo petróleo

— mas só geram valor quando você consegue analisá-los rapidamente.

Ciência de Dados

Limpeza, exploração e estatística sobre grandes volumes

Machine Learning

Pré-processamento de features antes de treinar modelos

Negócios

Relatórios, BI, análise financeira e dashboards

Pesquisa & Engenharia

Análise de sensores, simulações e experimentos

Pandas é o **padrão de fato** para manipular dados tabulares em Python.

Netflix · Spotify · Google · IBM · Bancos · NASA — todos usam Pandas em produção.

INTRODUÇÃO

O que é Pandas?

Pandas é uma biblioteca Python para **análise e manipulação de dados tabulares**.

- Criado por Wes McKinney em 2008, na AQR Capital Management
- Open source — mantido pela comunidade
- Escrito em C / Cython — performance próxima de C nativo
- Trabalha sobre o NumPy (arrays numéricos)

Pandas vs NumPy

PANDAS	NUMPY
DataFrame / Series	Array (ndarray)
Heterogêneo	Homogêneo
Rótulos (índices)	Posições numéricas
Pensado p/ dados reais	Pensado p/ matemática

Na prática: NumPy faz a matemática, Pandas organiza os dados.

SETUP

Instalando e importando

1. Instalação

```
# Pip
pip install pandas

# Já vem instalado no Google Colab! ✓
```

2. Importação convencional

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

Ecossistema

1. **NumPy** — Cálculo numérico (base do Pandas)
2. **Matplotlib** — Gráficos básicos
3. **Seaborn** — Gráficos estatísticos bonitos
4. **scikit-learn** — Machine Learning
5. **Jupyter / Colab** — Notebooks interativos

💡 **Dica:** vamos usar o Google Colab — abra **colab.research.google.com** e crie um novo notebook.

As duas estruturas que você precisa conhecer

Series

Estrutura 1-D (uma coluna)

0	rock
1	pop
2	jazz
3	rock

índice valores

- ✓ uma única coluna de dados
- ✓ cada coluna de um DataFrame é uma Series
- ✓ todos os elementos do mesmo tipo

```
s = pd.Series(['rock',
               'pop',
               'jazz'])
```

DataFrame

Estrutura 2-D (tabela)

	artist	year	pop
0	Beyoncé	2022	85
1	Drake	2023	92
2	Adele	2021	88

- ✓ várias colunas (tabela tipo planilha)
- ✓ cada coluna pode ter um tipo diferente
- ✓ **é o que você usa em 95% dos casos**

```
df = pd.DataFrame({
  'artist': ['Beyoncé', 'Drake'],
  'year':   [2022, 2023] })
```

Criando e acessando uma Series

Criando uma Series

```
# A partir de uma lista
temperaturas = pd.Series([28, 31, 29, 33, 27])

# Com índice customizado
vendas = pd.Series(
    [120, 200, 150],
    index=['jan', 'fev', 'mar'])
```

Acessando valores

```
vendas['fev']        # 200
vendas[['jan','mar']] # subset
vendas[vendas > 130] # filtro
```

O índice é a chave

Toda Series tem um índice — pode ser numérico ou de rótulos.

índice	valor
jan	120
fev	200
mar	150

📌 Acesso por rótulo, não por posição.

📌 Operações alinham automaticamente pelos índices.

Criando um DataFrame

A partir de um dicionário (mais comum)

```
df = pd.DataFrame({
    'artist': ['Beyoncé', 'Drake', 'Adele'],
    'year':   [2022, 2023, 2021],
    'pop':    [85, 92, 88]
})
```

Resultado:

	artist	year	pop
0	Beyoncé	2022	85
1	Drake	2023	92
2	Adele	2021	88

A partir de um arquivo (você vai usar muito)

```
df = pd.read_csv('spotify.csv')
df = pd.read_excel('vendas.xlsx')
df = pd.read_json('api_response.json')
```

Salvando um DataFrame

```
df.to_csv('saida.csv', index=False)
```

📌 **Conceito-chave**

Cada coluna de um DataFrame é uma Series.

Linhas têm um índice (0, 1, 2... ou rótulos customizados).

Colunas também têm um índice (os nomes).

INSPEÇÃO

Primeiro contato com os dados

A primeira coisa que você faz ao carregar dados:

`df.head()`

Primeiras 5 linhas (df.head(10) p/ 10)

`df.tail()`

Últimas 5 linhas

`df.shape`

(linhas, colunas) → ex.: (10000, 18)

`df.info()`

Tipos, nulos, memória — visão geral

`df.describe()`

Estatísticas (média, std, min, max...)

`df.columns / df.dtypes`

Lista de colunas e seus tipos

Fluxo de exploração rápida (3 linhas):

```
df = pd.read_csv('arquivo.csv')   # 1. carrega
df.head()                          # 2. olhar os dados
df.info(); df.describe()           # 3. entender colunas e estatísticas
```

Selecionando colunas

Uma coluna → retorna uma Series

```
df['artist']        # forma recomendada
df.artist           # atalho (só nomes válidos)
```

Várias colunas → retorna um DataFrame

```
df[['artist', 'year']]

# Note os colchetes duplos!
# Lista dentro de colchetes.
```

⚠ **Pegadinha:** `df['a', 'b']` é diferente de **`df[['a', 'b']]`**. O primeiro dá erro.

Visualizando

artist	year	pop
Beyoncé	2022	85
Drake	2023	92
Adele	2021	88
Ed Sheeran	2017	95

`df['artist']`

(coluna destacada)

Você pode operar direto sobre a coluna:

```
df['pop'].mean()
df['year'].max()
```

Selecionando linhas: .loc e .iloc

.loc[ ]

Acesso por RÓTULO

```
# uma linha
df.loc[0]

# várias linhas (lista)
df.loc[[0, 2, 5]]

# linha + coluna
df.loc[0, 'artist']

# fatia (INCLUI o fim!)
df.loc[0:3]   # 4 linhas
```

.iloc[ ]

Acesso por POSIÇÃO (inteiro)

```
# primeira linha
df.iloc[0]

# última linha
df.iloc[-1]

# linha + coluna (posição)
df.iloc[0, 2]

# fatia (NÃO inclui o fim)
df.iloc[0:3]  # 3 linhas
```

Filtros — o jeito mais poderoso

Use uma máscara booleana para filtrar linhas:

Filtro simples

```
# Músicas com popularidade > 90
df[df['pop'] > 90]

# Músicas de um artista específico
df[df['artist'] == 'Drake']
```

Múltiplas condições

```
# AND  →  &
df[(df['year'] >= 2020) & (df['pop'] > 80)]

# OR   →  |
df[(df['artist'] == 'Drake') | (df['artist'] == 'Adele')]
```

Padrões úteis

`.isin([...])`
está dentro de uma lista

`.between(a, b)`
está no intervalo [a, b]

`.str.contains('rock')`
string contém

`.str.startswith('A')`
começa com

`.isna() / .notna()`
é (ou não é) nulo

`~ (til)`
NEGA a condição

⚠ **Use `&` `|`** (não **`and` `or`**).

E sempre coloque cada condição entre parênteses!

MANIPULAÇÃO

Criando e modificando colunas

Atribua a uma coluna nova ou existente — Pandas resolve o resto.

1. Valor constante

```
df['active'] = True
df['source'] = 'spotify'
```

2. Derivada de outras colunas

```
# duração em segundos a partir de ms
df['dur_sec'] = df['duration_ms'] / 1000

# combinando colunas
df['total'] = df['nota1'] + df['nota2']
```

3. Condicional

```
import numpy as np
df['hit'] = np.where(
    df['pop'] >= 80, 'hit', 'flop'
)
```

4. Renomear ou remover

```
df = df.rename(columns={'pop': 'popularity'})

df = df.drop(columns=['source'])
df = df.drop(index=[0, 1])
```

💡 **`drop()`** e **`rename()`** retornam um novo DataFrame — você precisa atribuir de volta a **`df`**.

MANIPULAÇÃO

Valores faltantes (NaN)

Dados reais TÊM valores faltantes. Lidar com eles é parte do trabalho.

Detectar

```
df.isna()                # bool por célula
df.isna().sum()          # nulos por coluna
df.isna().any(axis=1)    # linhas com algum nulo
```

Remover

```
df.dropna()                  # qualquer NaN
df.dropna(subset=['pop'])    # só se 'pop' for NaN
```

Preencher

```
df.fillna(0)
df['pop'].fillna(df['pop'].mean())
```

O que é NaN?

Not a Number — marcador padrão do Pandas para dados faltantes.

artist	year	pop
Beyoncé	2022	85
Drake	*NaN*	92
Adele	2021	*NaN*
–	2020	70

3 estratégias:

1. Ignorar (apenas para análises rápidas)
2. Remover linhas/colunas com muitos nulos
3. Preencher com média, mediana, valor fixo

Ordenando dados

.sort_values()

Ordena pelos valores de uma coluna

```
# crescente (padrão)
df.sort_values('pop')

# decrescente
df.sort_values('pop',
               ascending=False)

# por várias colunas
df.sort_values(
  ['year', 'pop'],
  ascending=[True, False])
```

.sort_index()

Ordena pelo índice (linhas ou colunas)

```
df.sort_index()

# por nome das colunas
df.sort_index(axis=1)
```

Caso de uso comum: ranking

```
# Top 10 mais populares
(df.sort_values('pop',
                ascending=False)
   .head(10))
```

ANÁLISE

Estatísticas em uma linha

Pandas tem dezenas de funções estatísticas prontas.

`.sum()`
Soma

`.mean()`
Média

`.median()`
Mediana

`.min()` `.max()`
Mínimo e máximo

`.std()` `.var()`
Desvio-padrão e variância

`.count()`
Contagem (ignora NaN)

`.unique()`
Valores únicos

`.nunique()`
Quantos valores únicos

`.value_counts()`
Frequência de cada valor

`.describe()`
Tudo isso de uma vez!

Exemplos:

```
df['pop'].mean()                  # média da popularidade
df['artist'].value_counts().head(5)  # 5 artistas mais frequentes
df[['pop','year']].describe()     # estatísticas das colunas numéricas
```

ANÁLISE

groupby — agrupando e agregando

O comando mais poderoso do Pandas: agrupar e aplicar estatísticas.

Modelo mental: split → apply → combine

1. SPLIT

Divide o DataFrame em grupos pela coluna escolhida

2. APPLY

Aplica uma função em cada grupo (sum, mean, count...)

3. COMBINE

Junta os resultados em um único DataFrame

Exemplo: popularidade média por artista

```
df.groupby('artist')['pop'].mean()

# Várias estatísticas de uma vez:
df.groupby('artist')['pop'].agg(['mean', 'max', 'count'])

# Agrupando por várias colunas:
df.groupby(['year', 'genre'])['pop'].mean()
```

VISUALIZAÇÃO

Gráficos diretos do DataFrame

Pandas usa o Matplotlib por baixo — você só precisa de .plot()

```
import matplotlib.pyplot as plt

# Linha (padrão)
df['pop'].plot()

# Histograma
df['pop'].plot(kind='hist', bins=20)

# Barras horizontais
top = df.groupby('artist')['pop'].mean()
top.nlargest(10).plot(kind='barh')

# Dispersão
df.plot(kind='scatter',
        x='energy', y='dance')

plt.show()
```

Tipos comuns (parâmetro kind=)

`'line'`	tendência ao longo do tempo
`'bar' / 'barh'`	comparar categorias
`'hist'`	ver distribuição de valores
`'box'`	ver outliers e quartis
`'scatter'`	relação entre 2 colunas
`'pie'`	proporção de um total

RESUMO

Cheatsheet — o que levar pra prova da vida real

Carregar e inspecionar

```
pd.read_csv('arq.csv')
df.head() · df.tail()
df.info() · df.describe()
df.shape · df.columns
```

Selecionar

```
df['col']
df[['col1', 'col2']]
df.loc[linha, col]
df.iloc[i, j]
df[df['x'] > 10]
```

Transformar

```
df['new'] = df['a'] / df['b']
df.rename(columns={...})
df.drop(columns=[...])
df.fillna(0) · df.dropna()
```

Agregar e ordenar

```
df['col'].mean() / sum() / count()
df.value_counts()
df.sort_values('col')
df.groupby('col').mean()
df.plot(kind='bar')
```

Imprima isso. Cole na parede. Você vai consultar muito.

HORA DE PRATICAR

Spotify Top Hits

Análise exploratória de um catálogo musical real

Você vai usar tudo o que aprendemos: read_csv, head, filtros, groupby, sort_values e plot.

2000
músicas

18
atributos

6
desafios

ATIVIDADE PRÁTICA

O que vamos fazer

O dataset

Spotify Top Hits — músicas populares no Spotify com atributos de áudio extraídos pela própria API.

Principais colunas:

`artist`	Nome do artista	`song`	Nome da música
`year`	Ano de lançamento	`popularity`	0–100 (calculado pelo Spotify)
`danceability`	0–1 — quão dançante é	`energy`	0–1 — intensidade percebida
`tempo`	BPM (batidas por minuto)	`duration_ms`	Duração em milissegundos
`genre`	Gênero principal	`explicit`	True / False

Como funciona

1. **Abra o notebook no Colab**
   Importe o arquivo .ipynb compartilhado
2. **Faça upload do CSV**
   Arquivo spotify_top_hits.csv
3. **Acompanhe as células guiadas**
   Exploração + 6 conceitos demonstrados
4. **Resolva os desafios**
   6 problemas progressivos no final

📂 `Arquivos: A14-atividade-spotify.ipynb + spotify_top_hits.csv`

RECAPITULANDO

Você sai daqui sabendo...

- ✓ Diferença entre Series e DataFrame
- ✓ Como carregar dados de CSV/Excel
- ✓ Selecionar e filtrar linhas e colunas
- ✓ Lidar com valores faltantes (NaN)
- ✓ Agrupar e agregar com .groupby()
- ✓ Gerar gráficos rápidos com .plot()

Próxima aula:

Pandas avançado — merge, pivot_table, séries temporais e introdução ao Matplotlib + Seaborn.

Dúvidas? Pergunte agora ou no fórum da disciplina.