Aula 10: Bibliotecas e Análise de Dados

Se você já se perguntou como o Python lida com grandes volumes de dados, como os que você encontraria em uma planilha do Excel ou em um banco de dados, a resposta quase sempre envolve o Pandas.

Pense no Pandas como uma versão aprimorada do Excel, só que dentro do Python. É uma ferramenta fundamental para o kit de habilidades dentro de *Ciência de Dados* e *Análise de Dados*. Com ele, podemos:

- Carregar dados de arquivos (como `.csv` ou `.xlsx`).
- Limpar e organizar esses dados.
- Analisar e encontrar padrões.
- Manipular e transformar os dados.
- Salvar os resultados.

A página oficial da biblioteca é: https://pandas.pydata.org/

Importando a Biblioteca

Primeiro, precisamos ter certeza de que o Pandas está instalado (se não estiver, use `pip install pandas` no seu terminal Python).

No Python, a convenção da comunidade é importar o pandas usando o "alias" (apelido) `pd`. Vamos fazer isso e também verificar a versão que estamos usando, tratando possíveis erros:

```
# --- 1. Importação e Verificação do Pandas ---

import sys # Usaremos o sys para ver a versão do Python

# A convenção é "import pandas as pd"
try:
    print("Tentando importar a biblioteca Pandas...")
    import pandas as pd
    print("Pandas importado com sucesso!")

    # É uma boa prática verificar a versão
    print(f"Versão do Pandas instalada: {pd.__version__}")
    print(f"Versão do Python: {sys.version.split()[0]}")

except ImportError:
    print("\n--- ERRO ---")
    print("A biblioteca Pandas não foi encontrada.")
    print("Por favor, instale-a usando o comando no seu terminal:")
    print("pip install pandas")
except Exception as e:
    print(f"Ocorreu um erro inesperado: {e}")
finally:
```

```
    print("Verificação de setup concluída.\n")
```

Series

O Pandas tem duas estruturas de dados principais. A primeira e mais simples é a **Série.**

Nesse caso, seria como uma única coluna de uma planilha, ou um dicionário. Ela possui **valores** e um **índice** (que funciona como as chaves do dicionário).

Vamos criar uma Série com cargos e salários:

```
# --- 2. Criando uma Série (Estrutura 1D) ---

try:
    print("--- Criando uma Série (1D) com Cargos e Salários ---")

    dados_salarios = {
        'Analista de Dados': 7000.50,
        'Cientista de Dados': 12000.00,
        'Engenheiro de Dados': 11000.00,
        'Analista de BI': 6500.00
    }

    # Criamos a Série a partir do dicionário
    serie_salarios = pd.Series(dados_salarios)

    print("\n--- Série Criada ---")
    print(serie_salarios)

    # Explorando a estrutura
    print("\n--- Explorando a Estrutura da Série ---")
    print(f"Índice (Chaves): {list(serie_salarios.index)}")
    print(f"Valores: {list(serie_salarios.values)}")

    # Acessando um valor pelo índice (chave)
    print("\n--- Acessando um Valor ---")
    print(f"Salário do Cientista de Dados: R$ {serie_salarios['Cientista de Dados']:.2f}")

except NameError:
    print("\nERRO: O pandas (pd) não foi importado corretamente.")
except Exception as e:
    print(f"Ocorreu um erro ao criar a Série: {e}")
finally:
    print("Exploração de Séries concluída.\n")
```

DataFrames

A estrutura mais utilizada do Pandas é o **DataFrame**. Pense nele como a planilha inteira: uma tabela com múltiplas linhas e colunas - uma matriz.

Vamos criar um DataFrame do zero com 3 colunas: nome do filme, ano de lançamento e gênero:

```
# --- 3. Criando um DataFrame (Estrutura 2D) ---

try:
    print("--- Criando um DataFrame (2D) de Filmes ---")

    dados_filmes = {
        'nome_do_filme': ['O Poderoso Chefão', 'Interestelar', 'Parasita', 'Matrix'],
        'ano_de_lancamento': [1972, 2014, 2019, 1999],
        'genero': ['Criminal', 'Ficção Científica', 'Suspense', 'Ficção Científica']
    }

    # Criamos o DataFrame a partir do dicionário
    df_filmes = pd.DataFrame(dados_filmes)

    print("\n--- DataFrame Criado ---")
    print(df_filmes)

    print("\n--- Informações do DataFrame (.info()) ---")
    df_filmes.info()

except NameError:
    print("\nERRO: O pandas (pd) não foi importado corretamente.")
except Exception as e:
    print(f"Ocorreu um erro ao criar o DataFrame: {e}")
finally:
    print("Criação de DataFrame concluída.\n")
```

Leitura de Dados Externos

Criar dados manualmente é bom para o aprendizado, mas o poder do Pandas está em **ler dados do mundo real**. Uma das formas mais comuns de armazenar dados de tabela é em arquivos `.csv` (Valores Separados por Vírgula).

Vamos usar um dataset de exemplo do **Kaggle.com** (site que armazena datasets para compartilhamento e competições de Ciência de Dados).

Dataset: 700 Classic Disco Tracks (com dados do Spotify)

Você precisará baixar o arquivo `.csv` deste link e colocá-lo na mesma pasta do seu script Python. (O arquivo se chama `ClassicDisco.csv`).

```
# --- 4. Lendo um Arquivo CSV ---

# Defina o nome do arquivo que você baixou
nome_do_arquivo_csv = 'ClassicDisco.csv'
df_disco = None # Inicializamos o DataFrame como None

try:
    print(f"--- Lendo o arquivo '{nome_do_arquivo_csv}' ---")

    # O comando mágico para ler CSV!
    df_disco = pd.read_csv(nome_do_arquivo_csv)

    print("Arquivo lido com sucesso!")

    # Vamos espiar os dados. .head() mostra as 5 primeiras linhas
    print("\n--- As 5 primeiras linhas do DataFrame (.head()) ---")
    print(df_disco.head())

    # .info() nos dá um resumo das colunas e tipos de dados
    print("\n--- Informações do DataFrame (.info()) ---")
    df_disco.info()

except FileNotFoundError:
    print(f"\n--- ERRO: Arquivo não encontrado ---")
    print(f"O arquivo '{nome_do_arquivo_csv}' não foi encontrado no diretório.")
    print("Por favor, baixe o CSV do Kaggle e coloque-o na mesma pasta do script.")
except NameError:
    print("\nERRO: O pandas (pd) não foi importado corretamente.")
except Exception as e:
    print(f"Ocorreu um erro ao ler o arquivo CSV: {e}")
finally:
    print("Leitura de CSV concluída.\n")
```

Manipulação de Dados

Agora que temos nossos dados de Disco em um DataFrame (df_disco), podemos começar a explorá-lo e manipulá-lo:

```
# --- 5. Manipulando o DataFrame ---

# Este 'if' garante que o código só execute se o CSV foi lido com sucesso
if df_disco is not None:
    try:
        print("--- Explorando e Manipulando Dados ---")

        # 1. Ajustando a exibição (útil para ver mais linhas)
        # pd.options.display.max_rows = 10
        # print("\n--- Exibindo com max_rows=10 ---")
        # print(df_disco)
```

```
        # 2. Selecionando uma coluna específica (retorna uma Série)
        print("\n--- Selecionando a coluna 'Track Name' (as 5 primeiras) ---")
        print(df_disco['Track Name'].head())

        # 3. Filtragem de Dados
        # Vamos encontrar todas as músicas do 'ABBA'
        print("\n--- Filtrando: Músicas do 'ABBA' ---")
        # Criamos uma condição (filtro)
        filtro_abba = df_disco['Artist Name(s)'] == 'ABBA'
        df_abba = df_disco[filtro_abba]

        print(df_abba[['Artist Name(s)', 'Track Name', 'Album Name']])

        # 4. Criando uma nova coluna
        # Vamos criar uma coluna "Popularity_Category"
        print("\n--- Criando uma nova coluna 'Popularity_Category' ---")

        # Exemplo simples: se a popularidade (no Spotify) for > 60, é 'Alta'
        # Nota: Isso é uma introdução. Usaremos .apply() ou .loc[] no futuro.
        # Por enquanto, vamos criar uma coluna baseada em outra.
        df_disco['Popularity_x10'] = df_disco['Popularity'] * 10
        print(df_disco[['Track Name', 'Popularity', 'Popularity_x10']].head())

        # 5. Salvando em um novo arquivo CSV
        # Vamos salvar nosso DataFrame filtrado do ABBA
        print("\n--- Salvando dados filtrados em 'abba_disco_tracks.csv' ---")
        # index=False evita que o pandas salve o índice do DataFrame como uma coluna
        df_abba.to_csv('abba_disco_tracks.csv', index=False)
        print("Arquivo 'abba_disco_tracks.csv' salvo com sucesso!")

    except KeyError as e:
        print(f"\n--- ERRO de Chave ---")
        print(f"A coluna {e} não foi encontrada no DataFrame.")
        print("Verifique se o nome da coluna está digitado corretamente.")
    except Exception as e:
        print(f"Ocorreu um erro durante a manipulação: {e}")
    finally:
        print("Manipulação de dados concluída.\n")
else:
    print("O DataFrame 'df_disco' não foi carregado. As etapas de manipulação foram puladas.")
```

Arquivos Excel

- Para ler: `df = pd.read_excel('meu_arquivo.xlsx')`
- Para salvar: `df.to_excel('meu_novo_arquivo.xlsx', index=False)`

(Para isso funcionar, trabalhamos em conjunto com outra biblioteca: `pip install openpyxl`)

Atividade Prática

Escolha um dataset (pode ser o de Disco ou qualquer outro do Kaggle que lhe interesse) e tente aplicar o que aprendemos hoje:

1. Carregue o dataset (use `try/except` para tratar o `FileNotFoundError`).
2. Inspecione os dados (`.head()`, `.info()`).
3. Faça uma **filtragem** (ex: músicas com popularidade acima de 70, ou filmes de um gênero específico).
4. Crie uma **nova coluna** (pode ser baseada em outra coluna ou um valor fixo).
5. **Salve** o seu novo DataFrame (filtrado e com a nova coluna) em um arquivo `.CSV` diferente.