Biblioteca Pandas

Uma Breve Introdução

https://pandas.pydata.org/getting_started.html

Agenda da Apresentação

1. **O Problema:** Por que precisamos de uma ferramenta como o Pandas?
2. **A Solução:** O que é Pandas e suas estruturas de dados.
3. **Mãos à Obra:** Carregando e explorando dados.
4. **Manipulação:** Selecionando, filtrando e transformando dados.
5. **Análise:** Agregando informações com `groupby`.
6. **Conclusão:** Resumo e próximos passos.

1. O Problema: Dados em Python Puro

Manipular dados tabulares (como planilhas) usando apenas listas ou dicionários é possível, mas pode ser...

- **Verborrágico:** Muito código para tarefas simples.
- **Lento:** Operações não são otimizadas para grandes volumes.
- **Complexo:** Cálculos entre colunas e agregações são trabalhosos.

```
# Exemplo: Calcular média de salários em uma lista de dicionários
dados = [
    {'nome': 'Ana', 'salario': 5000},
    {'nome': 'Bruno', 'salario': 7200},
    {'nome': 'Carlos', 'salario': 4500}
]

total_salarios = sum(d['salario'] for d in dados)
media = total_salarios / len(dados)
print(f"A média salarial é: {media:.2f}") # Funciona, mas imagine com 1 milhão de linhas...
```

2. A Solução: Biblioteca Pandas

Pandas é a ferramenta definitiva para manipulação e análise de dados em Python.

- **Rápida e Eficiente:** Construída sobre NumPy, é otimizada para performance.
- **Poderosa e Flexível:** Facilita a leitura, limpeza, transformação, combinação e análise de dados.
- **Integrada:** É a base do ecossistema de Data Science em Python (Scikit-learn, Matplotlib, Seaborn).

Analogia: Pense no Pandas como um **Excel superpoderoso dentro do Python**, totalmente programável.

Instalação e Importação Padrão:

```
# No seu terminal ou prompt de comando
pip install pandas
```

```
# No seu código Python
import pandas as pd
```

Estruturas de Dados Fundamentais

Series

Um array **unidimensional** com rótulos (índice). É como uma **única coluna** de uma planilha.

```
s = pd.Series([10, 20, 30], index=['a', 'b', 'c'])
print(s)
# a    10
# b    20
# c    30
# dtype: int64
```

DataFrame

A estrutura principal! Uma tabela **bidimensional** com linhas e colunas rotuladas. É como uma **planilha inteira**.

```
data = {'Fruta': ['Maçã', 'Banana'], 'Preço': [2.5, 1.8]}
df = pd.DataFrame(data)
print(df)
#      Fruta  Preço
# 0     Maçã    2.5
# 1   Banana    1.8
```

3. Mãos à Obra: Criando nosso DataFrame

Para nossos exemplos, vamos criar um DataFrame de funcionários. Em um cenário real, carregaríamos de um arquivo com `pd.read_csv('arquivo.csv')`.

```
import pandas as pd
import numpy as np

# Seed para resultados reproduzíveis
np.random.seed(42)
data = {
    'Nome': ['Ana', 'Bruno', 'Carlos', 'Daniela', 'Eduardo', 'Fernanda'],
    'Departamento': ['Vendas', 'TI', 'Vendas', 'RH', 'TI', 'Vendas'],
    'Salario': np.random.randint(3000, 8000, 6),
    'Idade': np.random.randint(25, 50, 6)
}
df = pd.DataFrame(data)
```

Nosso `DataFrame` `df` está pronto para ser explorado!

Explorando o DataFrame (Inspeção Inicial)

Primeiros passos para "sentir" os dados.

- `.head(n)`: Mostra as primeiras `n` linhas (padrão 5).
- `.tail(n)`: Mostra as últimas `n` linhas (padrão 5).
- `.shape`: Retorna uma tupla `(linhas, colunas)`.
- `.columns`: Lista os nomes das colunas.

```
print(df.head(3))
#      Nome Departamento  Salario  Idade
# 0     Ana       Vendas     6873     48
# 1   Bruno           TI     3708     37
# 2  Carlos       Vendas     7219     25

print(df.shape)
# (6, 4)
```

Explorando o DataFrame (Análise Descritiva)

- `.info()`: **Essencial!** Um resumo técnico: tipo de dado por coluna, contagem de valores não nulos e uso de memória.

```
df.info()
# <class 'pandas.core.frame.DataFrame'>
# RangeIndex: 6 entries, 0 to 5
# Data columns (total 4 columns):
#  #   Column        Non-Null Count  Dtype
# ---  ------        --------------  -----
#  0   Nome          6 non-null      object
#  1   Departamento  6 non-null      object
#  2   Salario       6 non-null      int64
#  3   Idade         6 non-null      int64
# dtypes: int64(2), object(2)
# memory usage: 320.0+ bytes
```

- `.describe()`: Resumo estatístico para colunas numéricas (média, desvio padrão, mínimo, máximo, quartis).

```
df.describe()
#             Salario      Idade
# count      6.000000   6.000000
# mean    5676.666667  37.666667
# std     1625.333771   9.688481
# min     3466.000000  27.000000
# 25%     4418.000000  29.000000
# 50%     6268.000000  38.500000
# 75%     6690.000000  45.750000
# max     7426.000000  48.000000
```

4. Manipulação: Selecionando Colunas

Você pode selecionar uma ou mais colunas usando colchetes `[]`.

Uma coluna (retorna uma `Series`):

```
df['Salario']
# 0    6873
# 1    3708
# 2    7219
# Name: Salario, dtype: int64
```

Múltiplas colunas (retorna um `DataFrame`):

Note os colchetes duplos `[[]]`

```
df[['Nome', 'Salario']]
#       Nome  Salario
# 0      Ana     6873
# 1    Bruno     3708
# ...
```

Manipulação: Selecionando Linhas com `.loc` e `.iloc`

- `.loc`: Acessa por **rótulo** (nome do índice).
- `.iloc`: Acessa por **posição inteira** (índice numérico).

```
# Usando .loc para pegar a linha com índice 0
df.loc[0]
# Nome                  Ana
# Departamento       Vendas
# Salario              6873
# Name: 0, dtype: object

# Usando .iloc para pegar a primeira linha (posição 0)
df.iloc[0]
# Pegando as linhas de índice 1 a 3 (inclusive)
df.loc[1:3]
# Pegando as linhas nas posições de 1 a 3 (exclusivo)
df.iloc[1:4]
```

O Superpoder: Filtro Condicional

Esta é uma das funcionalidades mais poderosas. Seleciona linhas baseadas em uma ou mais condições.

Pergunta: Quais funcionários têm salário maior que R$ 6.000?

```
df[df['Salario'] > 6000]
#       Nome Departamento  Salario  Idade
# 0      Ana       Vendas     6873     48
# 2    Carlos       Vendas     7219     25
# 4   Eduardo           TI     6069     25
```

Pergunta: Quais funcionários do departamento de "Vendas" têm mais de 30 anos?

```
df[(df['Departamento'] == 'Vendas') & (df['Idade'] > 30)]
#       Nome Departamento  Salario  Idade
# 0      Ana       Vendas     6873     48
# 5  Fernanda      Vendas     7602     30 # Idade >= 30, então não entra
# (Corrigindo o exemplo para ser mais claro)
# No nosso caso, o resultado seria só a Ana.
```

Use `&` para "E", `|` para "OU" e `~` para "NÃO".

Criando e Modificando Colunas

É muito simples criar novas colunas a partir das existentes.

Criando uma coluna de bônus de 10% do salário:

```
df['Bonus'] = df['Salario'] * 0.10
df.head()
#      Nome Departamento  Salario  Idade   Bonus
# 0     Ana       Vendas     6873     48   687.3
# 1   Bruno           TI     3708     37   370.8
```

Lidando com dados ausentes (se houvesse):

```
df.isnull().sum() # Conta valores nulos por coluna
# df.dropna() # Remove linhas com valores nulos
# df.fillna(0) # Preenche valores nulos com 0
```

5. Análise: Agregando Dados com `groupby`

O método `groupby` segue o paradigma **"Split-Apply-Combine"** (Dividir-Aplicar-Combinar). Essencial para sumarizar dados.

Pergunta: Qual a média de salário por departamento?

1. **Split:** Divide o DataFrame em grupos ("Vendas", "TI", "RH").
2. **Apply:** Aplica uma função a cada grupo (ex: `mean()` na coluna `Salario`).
3. **Combine:** Combina os resultados em um novo DataFrame.

```
media_salario_depto = df.groupby('Departamento')['Salario'].mean()
print(media_salario_depto)
# Departamento
# RH          4438.000000
# TI          4888.500000
# Vendas      7231.333333
# Name: Salario, dtype: float64
```

Outras agregações: `.sum()`, `.count()`, `.max()`, `.min()`.

Visualização Rápida e Integrada

Pandas se integra nativamente com Matplotlib para criar visualizações rápidas diretamente do DataFrame ou Series.

```
import matplotlib.pyplot as plt

# Usando o resultado do nosso groupby anterior
media_salario_depto.plot(kind='bar',
                         title='Média de Salário por Departamento',
                         ylabel='Salário Médio (R$)')
plt.xticks(rotation=0) # Evita que os nomes fiquem de lado
plt.grid(axis='y', linestyle='--')
plt.show()
```

6. Resumo e Próximos Passos

O que vimos hoje:

- Pandas resolve o problema de manipulação de dados em Python.
- Estruturas principais: `DataFrame` e `Series`.
- **Exploração:** `.head()`, `.info()`, `.describe()`.
- **Seleção:** `[]`, `.loc[]`, `.iloc[]`.
- **Filtro Condicional:** `df[df['coluna'] > valor]`.
- **Agregação:** O poder do `.groupby()`.

Para onde ir agora?

- **Pratique!** Use datasets do Kaggle ou dados abertos do governo.
- **Documentação Oficial:** É sua melhor amiga.
- Explore funções para combinar DataFrames: `pd.merge()`, `pd.concat()`.
- Aprofunde-se em manipulação de datas e séries temporais.

Obrigado!

Perguntas?