Guia Explicativo: Pandas I

1. Pandas: Fundamentos Básicos

O que é o Pandas e por que é amplamente utilizado.

O Pandas é uma biblioteca Python amplamente utilizada para análise e manipulação de dados. É essencial em tarefas de ciência de dados e análise de dados devido à sua eficiência e facilidade de uso. O Pandas fornece estruturas de dados poderosas, como DataFrames e Series, que simplificam a organização e a análise de dados.

```
import pandas as pd
```

2. Noções Gerais de DataFrame

Introdução ao conceito de DataFrame: estrutura de dados tabular bidimensional.

DataFrames são a estrutura central do Pandas. Eles são semelhantes a tabelas de banco de dados ou planilhas, com linhas e colunas. DataFrames permitem organizar, filtrar e manipular dados de maneira eficiente.

```
# Criando um DataFrame a partir de um dicionário
data = {'Nome': ['Alice', 'Bob', 'Charlie', 'David'],
        'Idade': [25, 30, 35, 40]}
df = pd.DataFrame(data)

# Visualizando os primeiros registros
df.head()
```

Output:

```
      Nome  Idade
0    Alice     25
1      Bob     30
2  Charlie     35
3    David     40
```

3. Series: DataFrames Unidimensionais

Entendendo o conceito de Series: estrutura unidimensional semelhante a um array ou coluna de uma planilha.

Series são estruturas unidimensionais que armazenam dados, semelhantes a colunas de uma planilha. Elas são frequentemente usadas para representar uma única variável.

```
# Criando uma Series a partir de uma lista
idades = pd.Series([25, 30, 35, 40])

# Acessando elementos de uma Series
primeira_idade = idades[0]
```

Output:

```
0    25
1    30
2    35
3    40
dtype: int64

Primeira idade: 25
```

4. Operações Básicas com DataFrames e Series

Compreensão dos tipos de dados (dtypes) em um DataFrame. Uso do método describe() para obter estatísticas descritivas. Aplicação de funções básicas como mean(), sum() e outras em colunas.

```
# Verificando os tipos de dados das colunas
tipos_de_dados = df.dtypes

# Obtendo estatísticas descritivas
estatisticas = df.describe()

# Calculando a média das idades
media_idades = idades.mean()
```

Output:

```
Nome      object
Idade      int64
dtype: object

            Idade
count    4.000000
mean    32.500000
std      6.454972
min     25.000000
25%     28.750000
50%     32.500000
75%     36.250000
max     40.000000

Média das idades: 32.5
```

5. Índices em DataFrames

Explicação sobre o conceito de índices e como eles são usados no Pandas. Acesso a valores específicos usando índices e labels de colunas. Renomeando índices com o método rename(). Excluindo índices usando drop().

```
# Definindo um índice personalizado
df.set_index('Nome', inplace=True)

# Acessando valores específicos usando o índice
valor_alice = df.loc['Alice', 'Idade']

# Renomeando um índice
df.rename(index={'Alice': 'Alicia'}, inplace=True)

# Excluindo um índice
df.drop('Alicia', inplace=True)
```

Output:

```
       Idade
Nome
Bob       30
Charlie   35
David     40

Idade de Alice: 25

       Idade
Nome
Alicia     25
Bob        30
Charlie    35
David      40
```

6. Exibição e Manipulação de Colunas

Acessando colunas por nome. Seleção de múltiplas colunas usando uma lista de nomes. Realização de slices de intervalos em DataFrames.

```
# Acessando uma coluna por nome
idades = df['Idade']

# Selecionando múltiplas colunas
subconjunto = df[['Nome', 'Idade']]

# Realizando slices de intervalos
intervalo = df[1:3]
```

Output:

```
Nome
Bob        30
Charlie    35
Name: Idade, dtype: int64

         Nome  Idade
Bob        30
Charlie    35

         Nome  Idade
Nome
Bob        Bob     30
Charlie  Charlie     35
```

7. Lidando com Dados Ausentes e Duplicados

Identificação de valores nulos usando isna() e isnull(). Remoção de valores nulos com dropna() e preenchimento com fillna(). Detecção e remoção de duplicatas com duplicated() e drop_duplicates().

```
# Criando um DataFrame com dados ausentes
dados = {'Nome': ['Alice', 'Bob', 'Charlie', 'David'],
         'Idade': [25, None, 35, 40]}
df_ausente = pd.DataFrame(dados)

# Verificando valores nulos
valores_nulos = df_ausente.isna()

# Removendo linhas com valores nulos
df_sem_ausentes = df_ausente.dropna()

# Preenchendo valores nulos com a média
df_ausente['Idade'].fillna(df_ausente['Idade'].mean(), inplace=True)

# Identificando duplicatas
duplicatas = df.duplicated()

# Removendo duplicatas
df_sem_duplicatas = df.drop_duplicates()
```

Output:

```
     Nome  Idade
0  Alice   25.0
1    Bob    NaN
2  Charlie   35.0
3   David   40.0

     Nome  Idade
0  Alice   25.0
2  Charlie   35.0
3   David   40.0

     Nome  Idade
0  Alice   25.0
1    Bob   33.333333
2  Charlie   35.0
3   David   40.0

     Nome  Idade
0  Alice   25
1    Bob   30
2  Charlie   35
3   David   40
```

8. Operações com Colunas e Transformações

Adição de novas colunas baseadas em operações com colunas existentes. Aplicação de funções a elementos de uma coluna usando apply(). Realização de operações vetorizadas entre colunas.

```
# Adicionando uma nova coluna calculada
df['Idade_quadrado'] = df['Idade'] ** 2

# Aplicando uma função a uma coluna
def classificar_idade(idade):
    if idade < 30:
        return 'Jovem'
    else:
        return 'Adulto'

df['Classificação'] = df['Idade'].apply(classificar_idade)

# Realizando uma operação vetorizada
df['Idade_dobrada'] = df['Idade'] * 2
```

Output:

	Nome	Idade	Idade_quadrado	Classificação	Idade_dobrada
0	Alice	25	625	Jovem	50
1	Bob	30	900	Adulto	60
2	Charlie	35	1225	Adulto	70
3	David	40	1600	Adulto	80

Conclusão

O Pandas é uma ferramenta poderosa para trabalhar com dados tabulares em Python. Com este guia, você aprendeu os conceitos básicos, desde a criação de DataFrames e Series até a manipulação de índices, dados ausentes e duplicados, e operações com colunas. À medida que você avança em sua jornada de aprendizado, explore as capacidades avançadas do Pandas para análise de dados mais sofisticada. O Pandas se tornará uma ferramenta indispensável em suas atividades de análise e ciência de dados.

Exemplos práticos:

- **Análise de Dados de Vendas Online**:

Imagine que você trabalha para uma loja de comércio eletrônico e deseja analisar os dados de vendas online. Você tem um arquivo CSV com informações sobre transações, incluindo data, produtos vendidos, preço, etc. Usando o Pandas, você pode ler o arquivo, agrupar as vendas por data, calcular a receita total diária e criar visualizações gráficas, como gráficos de barras, para acompanhar o desempenho das vendas ao longo do tempo.

```
import pandas as pd

# Lê os dados do arquivo CSV
df = pd.read_csv("dados_de_vendas.csv")

# Agrupa as vendas por data e calcula a receita diária
vendas_diarias = df.groupby("Data")["Preço"].sum()

# Cria um gráfico de barras para visualizar as vendas
vendas_diarias.plot(kind="bar")
```

- **Análise de Dados Financeiros**:

Suponha que você seja um analista financeiro e deseja analisar o desempenho de um portfólio de ações. Você pode obter dados de preços de ações de várias fontes, como o Yahoo Finance, e usar o Pandas para calcular métricas financeiras, como o retorno diário, o desvio padrão e o índice de Sharpe, para avaliar o risco e o desempenho do portfólio.

```
import pandas as pd

# Obtém dados de preços de ações
precos_acao = pd.read_csv("dados_de_acoes.csv")

# Calcula o retorno diário
precos_acao["Retorno"] = precos_acao["Preço"].pct_change()

# Calcula o desvio padrão dos retornos
desvio_padrao = precos_acao["Retorno"].std()

# Calcula o índice de Sharpe
retorno_medio = precos_acao["Retorno"].mean()
sharpe_ratio = (retorno_medio - 0.03) / desvio_padrao
```

Esses exemplos ilustram como o Pandas é usado em cenários do dia a dia para analisar dados e obter insights valiosos. A flexibilidade do Pandas o torna uma ferramenta essencial para profissionais de diversas áreas.