Aula 01: Venv, Série com Pandas e Dataframes

Para começar, vamos relembrar alguns conceitos vistos na outra UC:

- **Variáveis**: são como "caixas" que usamos para guardar informações no nosso código. Mas, para a análise de dados, precisamos ir um pouco além.
    - **Qualitativas**: Representam qualidades, categorias ou características que não podem ser medidas numericamente. Exemplos: `cor`, `estado_civil`, `gênero`.
    - **Quantitativas**: Representam quantidades que podem ser medidas.
        - **Discretas**: Vêm de contagens. Exemplo: `número_de_alunos_em_uma_turma` (sempre um número inteiro).
        - **Contínuas**: Vêm de medições. Exemplo: `altura_de_uma_pessoa` (pode ter várias casas decimais, como 1.75).

- **População, Censo e Amostra**: em uma análise de dados, raramente conseguimos examinar todos os dados disponíveis.
    - **População**: O conjunto completo de todos os elementos que queremos estudar.
    - **Censo**: A coleta de dados de cada elemento da população.
    - **Amostra**: Um subconjunto da população que selecionamos para análise. A forma como escolhemos essa amostra é chamada de **amostragem**.

Ex:

Imagine que temos a seguinte série de dados: `["M", "F", "F", "M", "M"]`.

- Qual o tipo de variável que estamos lidando?
- Se essa é uma pequena parte do nosso conjunto total de dados, como a classificamos?

Preparando Seu Ambiente Virtual

Antes de tudo, precisamos organizar nosso espaço de trabalho. A melhor maneira de fazer isso é usando um **ambiente virtual**. Assim, cada projeto fica isolado, sem bagunçar a instalação do Python no seu computador.

Vamos juntos criar um ambiente virtual chamado `meu_projeto` usando o `venv`. Isso é feito no terminal do seu interpretador:

```
# Cria o ambiente virtual
python -m venv meu_projeto

# Ativa o ambiente (para Windows)
.\meu_projeto\Scripts\activate
```

```
# Desativa o ambiente (para Windows)
deactivate
```

Agora que nosso ambiente está ativado, vamos instalar as ferramentas que vamos usar. O `Pandas` é uma **biblioteca**, ou seja, um conjunto de códigos que outros desenvolvedores escreveram para nos ajudar. Então precisamos trazê-la para o venv:

```
# Instala a biblioteca pandas
pip install pandas
```

Para garantir que tudo foi instalado corretamente, e para manter o controle das nossas dependências, vamos usar o `pip freeze`. Ele lista tudo que está instalado no nosso ambiente.

```
# Salva as bibliotecas instaladas em um arquivo
pip freeze > requirements.txt
```

O arquivo `requirements.txt` é como uma lista de compras para o seu projeto. Ele permite que qualquer pessoa configure o mesmo ambiente com um único comando:

```
pip install -r requirements.txt
```

Pandas Series vs. Listas

Uma das estruturas de dados mais importantes no Pandas são as **Series**. Pense nela como uma coluna de uma planilha ou um array de uma dimensão, mas com um **índice** (rótulo) para cada valor.

Como exemplo, vamos usar o conceito de **dataset**, um conjunto de dados no qual estamos trabalhando:

```
import pandas as pd

idades_lista = [25, 30, 22, 28]

idades_series = pd.Series([25, 30, 22, 28])

print(f"Lista:\n{idades_lista}\n")
print(f"Series:\n{idades_series}")
```

A principal diferença é o índice. Na lista, só podemos **acessar** os valores pela posição. Numa Series, **além da posição, temos um índice explícito**.

Manipulando Séries

Vamos criar uma Series mais interessante, com notas de uma turma, usando índices personalizados:

```
notas = pd.Series([9.5, 8.0, 7.5, 9.0], index=['João', 'Maria', 'Pedro', 'Ana'])

print(notas)
```

Daqui, podemos realizar consultas mais específicas:

```
print(notas['Maria'])  # Acessando a nota de Maria
print(notas[2])        # Acessando a nota de Pedro
print(notas.mean())    # Calculando a média das notas
print(notas[notas > 8])  # Filtrando notas maiores que 8
print(notas.describe())  # Estatísticas descritivas das notas
```

Operações Aritméticas

Uma das grandes vantagens da Series é que podemos fazer operações matemáticas em todos os valores de uma vez, sem a necessidade de um loop:

```
# Aumentar todas as notas em 10%
notas_ajustadas = notas * 1.10
print(notas_ajustadas)
```

Operações entre duas Séries:

```
# Criar uma segunda Series para comparar
notas_exame_final = pd.Series([10.0, 7.5, 8.0, 9.5], index=['João', 'Maria', 'Pedro', 'Ana'])


# Calcular a diferença entre as notas
diferenca = notas_exame_final - notas_ajustadas
print(diferenca)
```

DataFrames

Se a Series é uma coluna, o **DataFrame** é uma tabela inteira; pense nele como a sua planilha do Excel. Um DataFrame é, na verdade, uma coleção de Series que compartilham o mesmo índice.

Atividade: DataFrame do Excel

Imagine que você recebeu uma planilha da gestão e precisa analisá-la. Vamos usar o Pandas para ler esse arquivo:

```
# Certifique-se de ter o arquivo 'nomedoseuarquivo.xlsx' na mesma pasta do seu código
# O pacote 'openpyxl' precisa estar instalado: pip install openpyxl

import pandas as pd

# Ler o arquivo Excel para um DataFrame
df = pd.read_excel('nomedoseuarquivo.xlsx')

# Exibir as primeiras 5 linhas do DataFrame
print(df.head())
```

Agora que temos o nosso DataFrame, podemos realizar algumas análises básicas:

- `df.sum()`: Calcula a soma de cada coluna numérica.
- `df.mean()`: Calcula a média de cada coluna numérica.
- `df.max()`: Encontra o valor máximo em cada coluna numérica.
- `df.min()`: Encontra o valor mínimo em cada coluna numérica.

Para ler uma **planilha específica** de um arquivo `.xlsx` que contém várias, podemos usar o parâmetro `sheet_name` na função `pandas.read_excel()`.

Você pode passar para o parâmetro:

1. O nome da planilha (como uma string).
2. O índice da planilha (como um número, onde 0 é a primeira planilha).

Ex:

```
import pandas as pd

# Ler a planilha chamada 'Vendas'
df_vendas = pd.read_excel('nomedoseuarquivo.xlsx', sheet_name='Vendas')

# Ler a segunda planilha (índice 1)
df_produtos = pd.read_excel('nomedoseuarquivo.xlsx', sheet_name=1)
```

Atividade Prática

Você recebeu um conjunto de dados de uma empresa de tecnologia voltada para aplicações de investimentos. O objetivo é responder algumas perguntas importantes:

- Quais são as máximas e mínimas de operação de compra e venda das transações?
- Qual CNPJ tem o ativo de maior valor?
- Qual valor total em transações de cada participante?

Usando os conceitos de DataFrame e os comandos que aprendemos, crie um script que carregue os dados e responda a essas perguntas.