

2025



Apostila

Python Básico para Análise de Dados em Finanças Públicas

Instrutor: Pedro Ferreira Galvão Neto


12 A 16 DE MAIO DE 09:00 - 11:00 AM

Gerência da Dívida Pública e Receita Extratributária
Superintendência Financeira
Subsecretaria do Tesouro Estadual

ECONOMIA
Secretaria de Estado
da Economia

GOVERNO DE
GOIÁS
O ESTADO QUE DÁ CERTO

✦ Apresentação	3
🚀 Instalação e Ambiente de Trabalho	3
Link para Notebook do curso no Google Colab:	3
🎓 Aula 1 – Fundamentos da Linguagem Python	4
Variáveis e tipos de dados	4
Operadores matemáticos	4
Listas	4
Dicionários	4
Estruturas de decisão	5
Laços de Repetição	5
Exercício prático:	5
📊 Aula 2 – Manipulação de Dados com Pandas	6
Instalação do Pandas (caso esteja usando ambiente local):	6
Importando o pandas	6
Lendo dados de um arquivo CSV	6
Lendo dados de um arquivo Excel	6
Explorando o DataFrame	6
Selecionando dados.....	6
Agrupamento e soma por grupo	7
Ordenação	7
Exercício prático:	7
📄 Aula 3 – Limpeza e Tratamento de Dados	8
Identificando dados ausentes.....	8
Preenchendo valores ausentes.....	8
Removendo registros com dados faltantes.....	8
Corrigindo tipos de dados	8
Padronizando textos.....	8
Corrigindo formatação de números com vírgula	8
Detectando duplicatas.....	8
Removendo duplicatas	9
Exercício prático:	9
☑️ Aula 4 – Visualização de Dados com Matplotlib e Seaborn	10
Instalação (caso esteja usando ambiente local).....	10

Importando bibliotecas	10
Gráfico de linha (evolução da receita).....	10
Gráfico de barras (municípios com maior receita).....	10
Gráfico de dispersão	10
Boxplot (distribuição de receitas).....	10
Exercício prático:	11
 Aula 5 – Exportação e Automação de Relatórios com Python	12
Exportando dados para CSV e Excel	12
Gerando gráficos e salvando como imagem	12
Criando relatórios automáticos com f-strings	12
Automatizando com laços (um relatório por município)	12
Exportando múltiplos DataFrames para uma planilha Excel.....	13
Exercício prático:	13

🚀 Apresentação

Este curso tem como objetivo capacitar profissionais que atuam na área de administração de finanças públicas a utilizarem a linguagem Python como ferramenta de apoio à análise de dados. Ao longo das aulas, serão exploradas as principais bibliotecas e técnicas para leitura, tratamento, visualização e interpretação de bases de dados reais utilizadas no setor público.

O curso parte do básico e não exige experiência prévia em programação. É voltado para analistas, técnicos e gestores que desejam aprimorar sua capacidade de extrair informações relevantes a partir de dados financeiros.

🚀 Instalação e Ambiente de Trabalho

Antes de começar, é importante garantir que você tenha acesso a um ambiente adequado para execução dos códigos em Python. Existem duas opções recomendadas:

Opção 1: Google Colab (recomendado)

- Gratuito, online, sem necessidade de instalação
- Acesse: <https://colab.research.google.com>
- Requer apenas uma conta Google

Opção 2: Instalar o Anaconda

- Pacote completo com Python, Jupyter Notebook e bibliotecas
- Download: <https://www.anaconda.com/products/distribution>
- Recomendado para quem deseja manter o ambiente localmente

Também será útil ter um editor de texto leve instalado, como o VS Code (<https://code.visualstudio.com/>), caso deseje explorar códigos fora do Jupyter/Colab.

Link para Notebook do curso no Google Colab:

https://colab.research.google.com/drive/11B8jo_CvARYBdO9_dulBviR9WVGkT20p?usp=drive_link

Aula 1 – Fundamentos da Linguagem Python

Objetivos:

- Apresentar a sintaxe e estrutura básica da linguagem Python
- Trabalhar com tipos de dados e estruturas de controle
- Realizar operações matemáticas simples

Tópicos abordados:

Variáveis e tipos de dados

```
nome = "João"  
idade = 35  
salario = 2500.75  
ativo = True
```

Operadores matemáticos

```
soma = 10 + 5  
subtracao = 10 - 2  
multiplicacao = 4 * 3  
divisao = 20 / 4
```

Listas

```
idades = ["Goiânia", "Anápolis", "Rio Verde"]  
idades.append("Catalão")  
print(idades[0]) # Goiânia
```

Dicionários

```
municipio = {  
    "nome": "Goiânia",  
    "populacao": 1500000,  
    "receita": 3200000000  
}  
print(municipio["nome"])
```

Estruturas de decisão

```
if salario > 3000:  
    print("Salário alto")  
else:  
    print("Salário médio ou baixo")
```

Laços de Repetição

```
for cidade in cidades:  
    print(cidade)  
  
# Exemplo com while  
contador = 0  
while contador < 3:  
    print(cidades[contador])  
    contador += 1
```

Exercício prático:

- Crie um dicionário representando um município com as chaves: nome, população, receita, despesa.
- Calcule e imprima o déficit ou superávit do município com base na receita e despesa.

Na próxima aula, aprenderemos a trabalhar com bases de dados reais usando a biblioteca **pandas**, que é uma das ferramentas mais importantes da análise de dados em Python.

Aula 2 – Manipulação de Dados com Pandas

Objetivos:

- Introduzir a biblioteca `pandas` e sua estrutura de dados principal: `DataFrame`
- Carregar dados a partir de arquivos CSV e Excel
- Realizar operações de seleção, filtro e agrupamento

Instalação do Pandas (caso esteja usando ambiente local):

```
pip install pandas openpyxl
```

Importando o pandas

```
import pandas as pd
```

Lendo dados de um arquivo CSV

```
df = pd.read_csv("dados_receita.csv", sep=";")  
print(df.head()) # Exibe as 5 primeiras linhas
```

Lendo dados de um arquivo Excel

```
df = pd.read_excel("dados_despesa.xlsx")
```

Explorando o DataFrame

```
print(df.columns)      # Lista as colunas  
print(df.shape)        # Dimensões (linhas, colunas)  
print(df.info())       # Tipo de cada coluna  
print(df.describe())   # Estatísticas descritivas
```

Selecionando dados

```
# Selecionar uma coluna
print(df["Receita"])

# Filtrar linhas
df_go = df[df["UF"] == "GO"]

# Selecionar múltiplas colunas
df_subset = df[["Município", "Receita"]]
```

Agrupamento e soma por grupo

```
df_agrupado = df.groupby("UF")["Receita"].sum()
print(df_agrupado)
```

Ordenação

```
df_ordenado = df.sort_values(by="Receita", ascending=False)
```

Exercício prático:

- Carregue um arquivo CSV contendo dados de receitas por município.
 - Selecione apenas os dados do seu estado.
 - Agrupe os dados por tipo de receita e calcule o total.
 - Exiba os 5 municípios com maior receita.
-

Aula 3 – Limpeza e Tratamento de Dados

Objetivos:

- Detectar e corrigir dados ausentes ou inconsistentes
- Padronizar formatos de texto e números
- Preparar dados para análise com maior confiabilidade

Identificando dados ausentes

```
print(df.isnull().sum()) # Total de valores nulos por coluna
```

Preenchendo valores ausentes

```
df["Receita"] = df["Receita"].fillna(0) # Preenche com zero
```

Removendo registros com dados faltantes

```
df = df.dropna() # Remove linhas com qualquer valor ausente
```

Corrigindo tipos de dados

```
df["Ano"] = df["Ano"].astype(int)  
df["Receita"] = df["Receita"].astype(float)
```

Padronizando textos

```
df["Município"] = df["Município"].str.upper() # Tudo maiúsculo  
df["UF"] = df["UF"].str.strip() # Remove espaços extras
```

Corrigindo formatação de números com vírgula

```
df["Receita"] = df["Receita"].str.replace(",", ".").astype(float)
```

Detectando duplicatas

```
duplicados = df[df.duplicated()]  
print(duplicados)
```

Removendo duplicatas

```
df = df.drop_duplicates()
```

Exercício prático:

- Carregue uma base com problemas simulados (valores nulos, formatos errados, duplicatas).
- Limpe os dados aplicando os conceitos acima.
- Gere um resumo estatístico dos dados limpos com **describe()**.

Aula 4 – Visualização de Dados com Matplotlib e Seaborn

Objetivos:

- Aprender a criar gráficos simples com matplotlib
- Utilizar seaborn para gráficos estatísticos
- Interpretar visualmente dados financeiros públicos

Instalação (caso esteja usando ambiente local)

```
pip install matplotlib seaborn
```

Importando bibliotecas

```
import matplotlib.pyplot as plt
import seaborn as sns
```

Gráfico de linha (evolução da receita)

```
plt.plot(df["Ano"], df["Receita"])
plt.title("Evolução da Receita ao Longo dos Anos")
plt.xlabel("Ano")
plt.ylabel("Receita")
plt.show()
```

Gráfico de barras (municípios com maior receita)

```
top_mun = df.sort_values(by="Receita", ascending=False).head(5)
plt.bar(top_mun["Município"], top_mun["Receita"])
plt.title("Top 5 Municípios por Receita")
plt.xticks(rotation=45)
plt.show()
```

Gráfico de dispersão

```
sns.scatterplot(data=df, x="Despesa", y="Receita", hue="UF")
plt.title("Receita vs Despesa por UF")
plt.show()
```

Boxplot (distribuição de receitas)

```
sns.boxplot(data=df, x="UF", y="Receita")
plt.title("Distribuição de Receita por UF")
plt.xticks(rotation=45)
plt.show()
```

Exercício prático:

- Gere um gráfico de linha com a evolução da receita de um município ao longo do tempo.
- Crie um gráfico de barras comparando a despesa total de 5 municípios.
- Visualize a dispersão entre receita e população com seaborn.

Na próxima aula, vamos aprender como exportar e automatizar relatórios com Python, preparando seus resultados para apresentação e compartilhamento.

Aula 5 – Exportação e Automação de Relatórios com Python

Objetivos:

- Aprender a exportar dados tratados para diferentes formatos
- Automatizar a criação e salvamento de relatórios
- Gerar arquivos prontos para apresentação e compartilhamento

Exportando dados para CSV e Excel

```
# Exportar para CSV
df.to_csv("dados_limpos.csv", index=False, sep=";")

# Exportar para Excel
df.to_excel("dados_relatorio.xlsx", index=False)
```

Gerando gráficos e salvando como imagem

```
plt.figure(figsize=(8, 4))
plt.plot(df["Ano"], df["Receita"])
plt.title("Receita ao Longo dos Anos")
plt.savefig("grafico_receita.png") # Salva o gráfico como imagem
plt.close()
```

Criando relatórios automáticos com f-strings

```
municipio = "Goiânia"
receita_total = df[df["Municipio"] == municipio]["Receita"].sum()

relatorio = f"""
Relatório de Receita Pública - {municipio}
-----
Receita Total: R$ {receita_total:,.2f}
"""

with open("relatorio_goiania.txt", "w") as file:
    file.write(relatorio)
```

Automatizando com laços (um relatório por município)

```
municipios = df["Municipio"].unique()

for mun in municipios:
    total = df[df["Municipio"] == mun]["Receita"].sum()
    texto = f"Município: {mun}\nTotal da Receita: R$ {total:,.2f}\n"

    with open(f"relatorio_{mun}.txt", "w") as f:
        f.write(texto)
```

Exportando múltiplos DataFrames para uma planilha Excel

```
with pd.ExcelWriter("relatorio_completo.xlsx") as writer:
    df.to_excel(writer, sheet_name="Dados Limpos", index=False)
    df.groupby("UF")[["Receita", "Despesa"]].sum().to_excel(writer, sheet_name="Resumo UF")
```

Exercício prático:

- Exporte os dados tratados para um arquivo Excel com múltiplas abas: uma com todos os dados, outra com o resumo por estado.
- Crie um gráfico da evolução da despesa de um município e salve-o como imagem.
- Automatize a geração de um relatório de texto simples para os 5 municípios com maior receita.