

# Usando Python para Gerenciar Dados

Marcos F. Silva      Bruno M. S. S. Melo  
João Paulo de F. Ramirez

23 de março de 2024

## Índice

<b>Apresentação do Curso</b>	<b>2</b>
Objetivo . . . . .	2
Exemplo de código em Pandas . . . . .	2
Exemplo de código em Polars . . . . .	3
Algumas referências bibliográficas . . . . .	3
Conteúdo programático . . . . .	3
Bases de dados utilizadas . . . . .	4
O processo de análise de dados . . . . .	4
<b>Parte 1 - Importação e exportação de dados</b>	<b>6</b>
Importação de dados contidos no Excel . . . . .	6
Exportando dados no formato Excel . . . . .	9

# Apresentação do Curso

## Objetivo

Usando bases de dados reais, o objetivo do curso é mostrar como utilizar a *linguagem* Python para fazer o gerenciamento dos dados. Esse processo também é conhecido por “data wrangling” e consiste em realizar a limpeza e pré-processamento de bases de dados de forma a deixá-las “prontas” para o trabalho de modelagem e obtenção das respostas que se deseja obter com a análise dos dados.

O curso tem foco nas bibliotecas Python como {pandas} e {polars} para manipulação de dados, além de {matplotlib} e {plotly} para visualização de dados.

Algumas bibliotecas-chave em Python para essas tarefas incluem: - pandas - polars - matplotlib - plotly - numpy - scipy - scikit-learn

## Exemplo de código em Pandas

```
import pandas as pd

# Criando um DataFrame de exemplo
df = pd.DataFrame({
    'nome': ['Alice', 'Bob', 'Carol'],
    'idade': [25, 30, 35]
})

print(df)
```

	nome	idade
0	Alice	25
1	Bob	30
2	Carol	35

## Exemplo de código em Polars

```
import polars as pl

# Criando um DataFrame de exemplo
df = pl.DataFrame({
    'nome': ['Alice', 'Bob', 'Carol'],
    'idade': [25, 30, 35]
})

print(df)
```

shape: (3, 2)

nome	idade
Alice	25
Bob	30
Carol	35

## Algumas referências bibliográficas

Para quem tiver interesse em avançar no estudo do Python, seguem alguns materiais que podem ser úteis:

- [Python Data Science Handbook](#)
- [Pandas Documentation](#)
- [Polars Documentation](#)
- [Matplotlib Documentation](#)
- [Seaborn Documentation](#)

## Conteúdo programático

Este curso foi pensado com o objetivo de mostrar os aspectos práticos envolvidos no processo de análise de dados tendo um enfoque nas operações que usualmente são realizadas nas bases de dados objeto de análise.

Este material está dividido em cinco partes:

Parte 1 - Importação e exportação dos dados  
Parte 2 - Operações típicas em bases de dados  
Parte 3 - Limpeza e arrumação dos dados  
Parte 4 - Introdução à visualização de dados  
Parte 5 - Análise exploratória e descritiva de dados

Essa divisão tem cunho meramente didático.

## Bases de dados utilizadas

Procuramos utilizar bases de dados semelhantes àquelas que o usuário possa lidar no seu dia a dia.

Especificamente serão utilizadas ao longo dessa apostila as seguintes bases de dados:

- `ISP2020.xlsx` - esta base de dados, em formato do Excel, é disponibilizada pelo Mi-nistério da Previdência Social e contém informações sobre o indicador de situação previdenciária de todos os Regimes Próprios de Previdência Social - RPPS do Brasil.
- `fundos_investimentos.xlsx` - esta base de dados contém uma relação de fundos de investimentos e seu enquadramento conforme norma do Conselho Monetário Nacional.
- `UG123400_Jan18-Set18_Cta621201101.xls` - base de dados relativa ao razão contábil da conta 621201101 extraída da UG 23400 - RIOPREVIDENCIA.
- `dair_br2023.Rds` - esta base de dados contém informações sobre os investimentos realizados pelos RPPS de todo o Brasil. Contém informações para o ano de 2023. Foi obtida a partir da API do CADPREV.

## O processo de análise de dados

A análise de dados pode ser vista como um processo onde o analista, partindo de uma questão que quer responder a partir da análise dos dados, identifica os dados necessários e faz a **importação** desses dados para o Python (naturalmente pode ser utilizada qualquer outra linguagem de programação ou *softawre* para análise de dados).

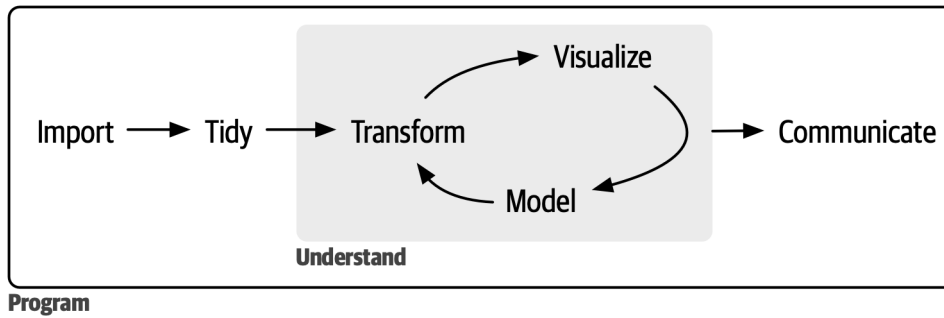
Usualmente esses dados não estarão organizados num formato no qual já seja possível realizar análises, sendo frequentemente necessário **arrumar**, **organizar** (do inglês, *tidy up the data*) esses dados.

Então o analista pode identificar a necessidade de **transformar** algumas variáveis da base de dados e **combinar** a base com outras, para gerar mais informações.

A identificação de relações entre as variáveis da base de dados, sua distribuição, a possível identificação de valores atípicos, a obtenção de novos *insights*, podem ser enormemente facilitados com a **visualização** dos dados.

Com isso o analista já está em condições de criar **modelos** que expliquem o comportamento observado das variáveis e certamente deverá **reportar**, **comunicar**, esses resultados a outras pessoas interessadas na resposta à questão que motivou toda a análise conduzida pelo analista.

Todo esse processo pode ser sintetizado na seguinte figura:<sup>1</sup>



---

<sup>1</sup>Fonte: [R for Data Science, 2nd Ed.](#)

## Parte 1 - Importação e exportação de dados

Toda análise de dados vai ter início com a importação para Python dos dados necessários à análise a ser realizada. Para a importação de dados, existem diversas opções de bibliotecas a depender do formato do arquivo no qual os dados estão armazenados.

Uma opção muito comum é termos dados armazenados em planilhas eletrônicas, em particular o Excel. Mas diversas outras opções de arquivos para armazenamento de dados existem como, por exemplo, `csv` e `json`.

Para a importação de dados, o `{pandas}` é uma biblioteca muito utilizada, permitindo a leitura de dados em diferentes formatos, como texto, Excel, SQL, entre outros.

Para a importação de dados no formato do Excel, o `{pandas}` faz uso de bibliotecas específicas para este fim. Uma das mais utilizadas chama-se `{openpyxl}` (para o formato mais atual `xlsx`) e `{xlrd}` (para o formato mais legado `xls`).

Para instalar o `{pandas}` e `{openpyxl}`, você pode usar o gerenciador de pacotes `pip`. Observe que a sintaxe é bem simples. O comando `-U` no `pip install` é opcional e indica ao `pip` para atualizar os pacotes a serem instalados para a versão mais recente disponível. Se os pacotes já estiverem instalados, o `pip` irá verificar se há uma versão mais recente disponível no Python Package Index (PyPI). Se houver, o `pip` irá desinstalar as versões atualmente instaladas e instalar as versões mais recentes.

```
!pip install -U pandas openpyxl xlrd
```

Depois de instalar as bibliotecas, é necessário importá-las para que suas funções sejam disponibilizadas para uso.

```
# Importando a biblioteca Pandas
import pandas as pd
```

Observe que a menos que queiramos utilizar diretamente a biblioteca `{openpyxl}`, não é necessário importá-la diretamente uma vez que o `Pandas`, quando necessitar realizar uma operação de leitura ou escrita para arquivos do Excel irá procurar, automaticamente, uma biblioteca instalada no sistema que sirva a este fim. No caso em tela, ele encontrará a biblioteca `{openpyxl}` e importará suas funcionalidades sem que precisemos fazê-lo de forma explícita.

### Importação de dados contidos no Excel

Para a importação de dados armazenados em arquivos Excel, o `Pandas` possui a função `read_excel` que permite a leitura de arquivos no formato `xlsx`.

Vamos iniciar fazendo a importação dos dados contidos no arquivo excel `ISP2020.xlsx`, que contém 9 abas (ou planilhas). Estamos interessados nos dados da planilha “GRUPOS”.

Para fazer a importação dos dados, vamos utilizar a função `read_excel()` do pacote `pandas` que já foi anteriormente carregado.

```
# Importando a planilha "GRUPOS"
grupos = pd.read_excel("./dados/ISP2020.xlsx", sheet_name="GRUPOS")
```

#### Aviso

Caso tentem executar esse código com a planilha aberta vocês obterão uma mensagem de erro!

Importado o conjunto de dados, é sempre recomendado dar uma conferida para ver se a importação ocorreu conforme esperado. Para fazer essa checagem, podemos utilizar as funções `info()`, `head()` e `tail()`.

```
# Verificando as informações básicas do DataFrame
print(grupos.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2154 entries, 0 to 2153
Data columns (total 8 columns):
#   Column                                Non-Null Count  Dtype
---  -
0   ENTE                                2154 non-null   object
1   GRUPO                              2154 non-null   object
2   SUBGRUPO                           2154 non-null   object
3   SEGURADOS ATIVOS                   2154 non-null   int64
4   APOSENTADOS                        2154 non-null   int64
5   PENSIONISTAS                       2154 non-null   int64
6   ESTRUTURA DE MATURIDADE DE MASSA  2154 non-null   float64
7   FONTE                              2154 non-null   object
dtypes: float64(1), int64(3), object(4)
memory usage: 134.8+ KB
None
```

```
# Inspeção dos registros iniciais da base de dados
print(grupos.head(2))
```

	ENTE	GRUPO	SUBGRUPO	SEGURADOS ATIVOS	\
0	ABADIA DE GOIÁS - GO	PEQUENO PORTE	MENOR MATURIDADE		325
1	ABADIÂNIA - GO	PEQUENO PORTE	MENOR MATURIDADE		415

	APOSENTADOS	PENSIONISTAS	ESTRUTURA DE MATURIDADE DE MASSA	FONTE
0	36	7	7.558140	DRAA2020
1	49	24	5.684932	DRAA2016

```
# Inspeção dos registros finais da base de dados
print(grupos.tail(2))
```

	ENTE	GRUPO	SUBGRUPO	SEGURADOS ATIVOS	\
2152	XANGRI-LÁ - RS	MÉDIO PORTE	MENOR MATURIDADE		833
2153	ZACARIAS - SP	PEQUENO PORTE	MENOR MATURIDADE		242

	APOSENTADOS	PENSIONISTAS	ESTRUTURA DE MATURIDADE DE MASSA	FONTE
2152	75	7	10.158537	DIPRO6/2020
2153	46	10	4.321429	DIPRO4/2020

Aparentemente, os dados foram importados sem maiores problemas.



### 💡 Exercício

1. Importar os dados contidos na planilha RESULTADO. Fazer a inspeção dos dados conforme mostrado. Coloque os dados no objeto `resultado`.
2. Importar os dados contidos no arquivo `fundos_investimentos.xlsx`. Fazer a inspeção dos dados.

### 💡 Desafio

Importar os dados contidos no arquivo `UG123400_Jan18-Set18_Cta621201101.xls`. Coloque os dados no objeto `razao` e faça as checagens recomendadas.



## Exportando dados no formato Excel

Para exportar dados que já estejam no Python, não será necessário instalar nenhum pacote adicional uma vez que a biblioteca `{openpyxl}`, que utilizamos para importar dados do Excel, também possui a funcionalidade de exportação. De forma análoga à vista na seção anterior, o Pandas encapsula essa funcionalidade na função `write_excel()`

Vamos exportar para uma planilha do excel o arquivo do razão contábil que acabamos de importar.

```
pd.write_excel(razao, path="dados/razao.xlsx", index=False)
```

### Dica

No Python, os objetos possuem algumas funções a eles associadas. Essas funções são chamadas de **métodos**. No contexto do Pandas, os objetos que armazenam tabelas possuem o método `to_excel()` que poderia ter sido chamado alternativamente à função `write_excel`.

O exemplo a seguir ilustra a exportação para o Excel por meio de uma chamada ao método `to_excel()` associado ao objeto `razao`.

```
razao.to_excel(razao, path="dados/razao.xlsx", index=False)
```