



# COMO IMPORTAR ARQUIVOS CSV NO PANDAS (PYTHON)

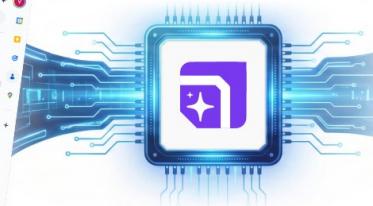
Guia completo de `read_csv()`: parâmetros essenciais (`sep`, `header`, `dtype`), encodings, dados ausentes, chunks, `parse_dates`, casos práticos (grandes arquivos, múltiplos CSVs), pandas vs csv nativo.

# Crie Ebooks técnicos incríveis em minutos com IA

Conheça a 1ª IA Especializada na criação de Ebooks **com código!**



Chega de formatar código no Google Docs



Deixe que nossa IA faça o trabalho pesado

 Syntax Highlight

 Adicione Banners Promocionais

 Edite em Markdown em Tempo Real

 Infográficos feitos por IA

**TESTE AGORA** 

 **Atualizado para Python 3.13 (Dezembro 2025)**

Pandas 2.2+ com `read_csv()` otimizado, `dtype_backend='pyarrow'` e casos práticos.

Olá, futuro **Cientista de Dados**!

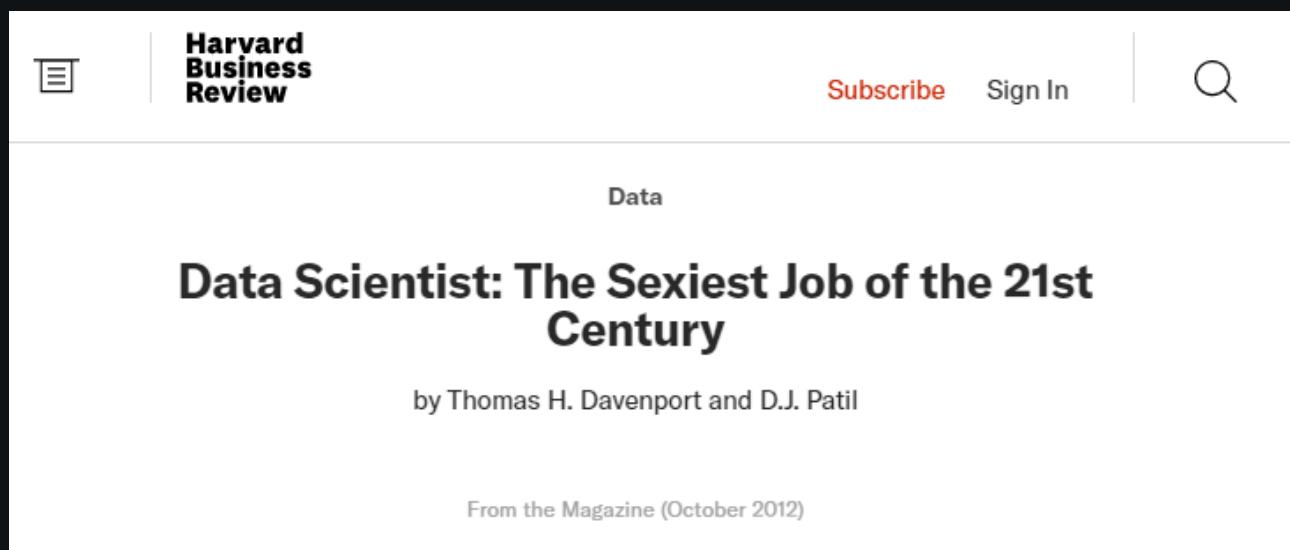
`pd.read_csv()` é a função mais usada em Data Science! Aprenda parâmetros essenciais, otimizações e casos reais.

Neste guia: -  **Parâmetros essenciais** - sep, header, dtype, encoding -  **Grandes arquivos** - chunksize, usecols -  **Dados ausentes** - na\_values, keep\_default\_na -  **Casos práticos** - Múltiplos CSVs, parse\_dates

No *post* de hoje vamos começar a nossa jornada nessa área que tem chamado muita atenção de profissionais e de empresas!

E não é à toa: o mercado está cada vez mais procurando profissionais com esse perfil, no Brasil e no mundo.

Até a [Harvard Business Review](#) concorda comigo!



The screenshot shows a mobile view of a Harvard Business Review article. At the top, there's a navigation bar with a menu icon, the HBR logo, 'Subscribe', 'Sign In', and a search icon. Below the header, the word 'Data' is centered. The main title of the article is 'Data Scientist: The Sexiest Job of the 21st Century', followed by the authors' names, 'by Thomas H. Davenport and D.J. Patil'. At the bottom of the visible area, it says 'From the Magazine (October 2012)'.

Esse post dá início à uma **série de posts** da biblioteca que é considerada por muitos um dos motivos de Python ser cada vez mais utilizado por cientista de dados: **o poderoso Pandas!** 

***Mas calma! Sem medo galera!***

Você vai ver que, apesar de versátil - como as boas bibliotecas pythonistas - Pandas é extremamente simples de utilizar.

Entendendo o básico de dataframes, series e índices, vocês dominarão o mundo do *data wrangling* (nome bonito para: **FUÇAR OS DADOS** 😊).

Mas antes de aprender isso tudo em outros posts, vamos direto ao assunto para quebrar logo o gelo e começar com o pé direito: importar arquivos CSV e visualizar os dados!

## Arquivo CSV

***Cara, não devo saber nada mesmo, o que é um arquivo CSV???***

Não tema meu amigo pythonista, um arquivos CSV nada mais é que um arquivo de texto com valores separados por vírgula (**Comma Separated Values**).

Ou seja, nada mais é que um arquivo onde cada linha possui valores separados por vírgulas. Bora de exemplo?

```
musica,artista,ano de lancamento,nome completo
Radioactive,Imagine Dragons,2012,Radioactive by Imagine Dragons
Good Vibrations,The Beach Boys,1966,Good Vibrations by The Beach Boys
A Day In The Life,The Beatles,1967,A Day In The Life by The Beatles
Message In A Bottle,The Police,1979,Message In A Bottle by The Police
Seven Nation Army,The White Stripes,2003,Seven Nation Army by The White Stripes
```

O arquivo mostra os dados de 5 músicas, com banda e ano de lançamento, além da última coluna que contém o nome completo.

Percebam que a primeira linha é o cabeçalho, com o nome descritivo de cada coluna.

### ***Viu? Moleza, não é?***

Claro que, na maioria dos casos práticos, os arquivos terão milhares ou mesmo milhões de linhas. Contudo, enquanto seu computador tiver memória para aguentar, o Pandas estará do seu lado!

Quando não estiver do nosso lado, bem, aí é assunto pra um outro post (deixem comentários aqui embaixo se gostariam de um post com esse assunto 😊)

Vamos então ao que interessa: **Pandas!!!**



***Ops, esse não, produtor!!***

# Instalação do Pandas

Caso você utilize o Anaconda, o Pandas já vem instalado por padrão.

Caso contrário, sigam-me os bons.

Vamos instalá-lo utilizando nosso bom e velho `pip`.

Primeiro, vamos ao básico: **ative** seu *ambiente virtual* para começar a instalação.

**COMO ASSIM** você não usa ambientes virtuais?! 😱

Se você é desses, já vai lá no post sobre **ambientes virtuais** pra começarmos direito!

**Para instalar o Pandas** (após ativar seu ambiente virtual), basta executar o comando:

```
pip install pandas
```

*Voilá*, temos nossa ferramenta pronta pra trabalhar!

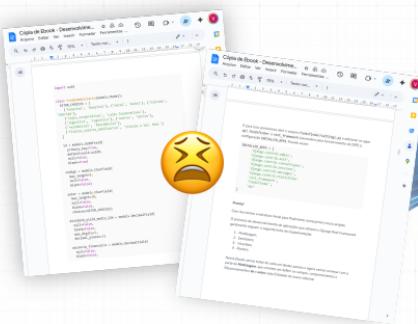
Agora escolha sua melhor IDE, seu notebook jupyter ou mesmo a linha de comando e **mãos à obra** - ou melhor - **mãos ao código** 💪



*Estou construindo o **DevBook**, uma plataforma que usa IA para criar e-books técnicos — com código formatado e exportação em PDF. Não deixe de conferir!*

## Crie Ebooks técnicos incríveis em minutos com IA

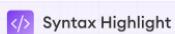
Conheça a 1ª IA Especializada na criação de Ebooks **com código!**



Chega de formatar código no Google Docs



Deixe que nossa IA faça o trabalho pesado



Syntax Highlight



Adicione Banners Promocionais



Edita em Markdown em Tempo Real



Infográficos feitos por IA

**TESTE AGORA! PRIMEIRO CAPÍTULO 100% GRÁTIS** 

## Importar Arquivo CSV com Pandas

Primeramente temos que importar a biblioteca!

Sabemos que na hora de importar qualquer biblioteca em Python, podemos criar um apelido (alias) para facilitar o acesso.

No caso do Pandas, o padrão é chamá-lo carinhosamente de 'pd':

```
>>> import pandas as pd
```

Com o Pandas importado, vamos utilizar o método `read_csv()`, passando como parâmetro o arquivo CSV que queremos importar.

No nosso caso, vamos usar o arquivo *bandas.csv*, que você pode [clicar aqui para baixá-lo][bandas-csv].

O método retorna uma estrutura tabular que é a base do Pandas, chamada de **Dataframe**.

Um dataframe nada mais é que uma estrutura bidimensional, com linhas e colunas.

Imagine uma planilha do Excel ou uma tabela do seu banco de dados e terá uma boa noção de como ele se comporta:

```
>>> import pandas as pd  
>>> musicas = pd.read_csv('./bandas.csv')
```

### ***Como assim, só isso?***

Sim, jovem padawan!! Mais mole, só sentando na gelatina.

### ***Mas... O que aconteceu?***

Nosso querido *pandas* leu o arquivo CSV, já entendeu que a primeira linha é o cabeçalho, e separou as colunas direitinho pra você!

### ***Ainda duvida?***

Vamos rodar o comando abaixo para verificar as 5 primeiras linhas do nosso *Dataframe* com músicas do arquivo CSV importado:

```
>>> import pandas as pd  
>>> musicas = pd.read_csv('./bandas.csv')  
>>> musicas.head()
```

E o resultado será:

	musica	artista	ano de lancamento		
	nome completo				
0	Radioactive	Imagine Dragons	2012		
	Radioactive by Imagine Dragons				
1	Good Vibrations	The Beach Boys	1966	Good	
	Vibrations by The Beach Boys				
2	A Day In The Life	The Beatles	1967	A	
	Day In The Life by The Beatles				
3	Message In A Bottle	The Police	1979	Mes-	
	sage In A Bottle by The Police				
4	Seven Nation Army	The White Stripes	2003	Seven Na-	
	tion Army by The White Stripes				

**Uai... E essa primeira coluna com valores de 0 a 4, que não estava no arquivo original!?!?**

Esse padawan é esperto! 😊

Veremos num próximo post que o Pandas não cria “nomes” somente para as colunas, mas também para as linhas!!

Esse “nome” é chamado de **índice** e será muito importante para realizarmos seleções de partes dos dados de um *Dataframe* Pandas.

Maaaaas... Cenas para um próximo capítulo

Claro que esse CSV já estava todo arrumadinho, limpinho e cheiroso. Mas *não se engane*: nem sempre será assim.

Um dos trabalhos do cientista de dados é **limpar os dados**, que consiste em:

- Consertar codificação de dados de entrada;
- Converter tipos;
- Configurar separadores de decimais;
- Limpar caracteres indesejados;
- Tratar dados faltantes entre outros vários problemas.

Podemos já tratar alguns desses problemas no momento da importação dos dados, no próprio método `read_csv()`

Um exemplo de parâmetro é dizer para o Pandas que não queremos que a primeira linha seja considerada o cabeçalho, mas sim parte dos dados.

Para isso, podemos executar o método `read_csv()` com o parâmetro *header* com o valor *None*:

```
>>> musicas_sem_cabecalho = pd.read_csv('./bandas.csv', header=None)
>>> musicas_sem_cabecalho.head()
```

Com esse parâmetro, temos o resultado:

```
0          1          2
0          3
0      musica      artista  ano de lancamen-
        to          nome completo
1  Radioactive  Imagine Dragons      2012  Radioac-
    tive by Imagine Dragons
2  Good Vibrations  The Beach Boys      1966  Good Vibra-
    tions by The Beach Boys
3  A Day In The Life  The Beatles      1967  A Day In
    The Life by The Beatles
4  Message In A Bottle  The Police      1979  Message In
    A Bottle by The Police
```

Percebam que agora temos outros nomes para cada coluna! O Pandas cria os nomes automaticamente, começando do 0.

## Bônus: e exportar para CSV?

Uma vez com o *Dataframe* Pandas em memória (após a execução do `read_csv()`), podemos escrever um arquivo CSV de saída utilizando o método `to_csv()`, da seguinte forma:

```
>>> musicas_sem_cabecalho.to_csv('./bandas_sem_cabecalho.csv')
```

**Prontinho!** temos nosso arquivo CSV exportado! 😊

## Casos Práticos Avançados

### 1. Ler Arquivos Grandes em Chunks

```
import pandas as pd

# Arquivo muito grande? Use chunks!
chunk_size = 10000
chunks = []

for chunk in pd.read_csv('vendas_gigante.csv', chunksize=chunk_size):
    # Processar cada chunk
    chunk_filtrado = chunk[chunk['valor'] > 100]
    chunks.append(chunk_filtrado)

df_final = pd.concat(chunks, ignore_index=True)
print(f"Total de linhas: {len(df_final)}")
```

## 2. Combinar Múltiplos CSVs

```
import pandas as pd
from pathlib import Path

# Ler todos os CSVs de uma pasta
pasta = Path('./dados/')
arquivos_csv = pasta.glob('vendas_*.csv')

dfs = [pd.read_csv(arquivo) for arquivo in arquivos_csv]
df_completo = pd.concat(dfs, ignore_index=True)
print(df_completo.shape)
```

## 3. Parse de Datas Automático

```
import pandas as pd

df = pd.read_csv(
    'vendas.csv',
    parse_dates=['data_venda', 'data_entrega'], # Converte para datetime
    date_format='%Y-%m-%d' # Pandas 2.0+
)

print(df.dtypes)
# data_venda      datetime64[ns]
# data_entrega   datetime64[ns]
```

## 4. Otimizar Tipos de Dados

```
import pandas as pd

# Especificar dtypes economiza memória
df = pd.read_csv(
    'dados.csv',
    dtype={
        'id': 'int32',
        'quantidade': 'int16',
        'preco': 'float32',
        'categoria': 'category'
    }
)

print(df.memory_usage(deep=True))
```

## Pandas vs CSV Nativo

```
import pandas as pd
import csv

# CSV Nativo (baixo nível)
with open('dados.csv', 'r') as f:
    reader = csv.DictReader(f)
    dados = [row for row in reader]

# Pandas (alto nível) - RECOMENDADO!
df = pd.read_csv('dados.csv')
```

## Quando usar cada um?

 **Use Pandas quando:** - Precisa **análise** de dados - Quer **manipulação** fácil - Trabalha com **DataFrames** - Precisa **performance** (numpy backend)

 **Use csv nativo quando:** - Arquivo **muito simples** - **Sem dependências** externas - Apenas **leitura** básica - Script **leve**

## Conclusão

Neste guia de `read_csv()`, você aprendeu:

-  **Parâmetros essenciais** - sep, header, dtype, encoding
-  **Grandes arquivos** - chunksize para processar em partes
-  **Múltiplos CSVs** - concat() para unir DataFrames
-  **Parse\_dates** - Conversão automática de datas
-  **Otimização** - dtype para economizar memória

**Principais lições:** - `read_csv()` é **extremamente flexível** - Use `chunksize` para arquivos **gigantes** - `dtype` economiza **memória** - `parse_dates` converte strings para **datetime** - Pandas é **mais poderoso** que csv nativo

**Próximos passos:** - Aprenda [Séries](#) - Explore [DataFrames](#) - Pratique `pd.to_csv()` para exportar - Estude `pd.read_excel()` para Excel

Primeiro contato com Pandas, o que achou? Fácil, difícil?

Comenta aqui embaixo suas dores e/ou sugestões! A gente quer te ouvir 😊

Esse foi somente o primeiro de muitos posts, pois esse assunto é muito extenso e importante para quem está iniciando a carreira de cientista de dados.

Vamos juntos que o caminho é longo, mas possível e gratificante!

Antes de terminar: sabia que a gente está no [Instagram](#)?

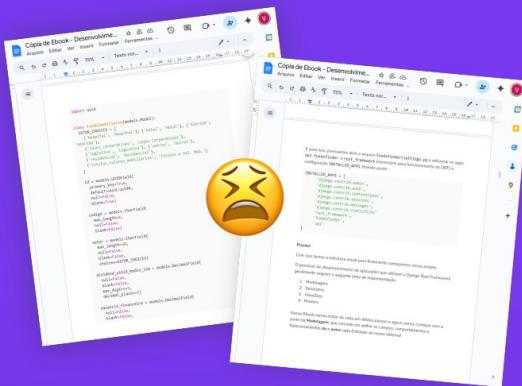
Vai lá e deixe seu comentário, interaja com a gente! Tem posts complementares ao blog e muitos desafios: procurem nos destaque 😊

Até o próximo post!



# Crie Ebooks técnicos em minutos com IA

Conheça a 1ª IA Especializada na criação de Ebooks **com código!**



Chega de formatar código no Google Docs



**Syntax Highlight**

**Arquitetura de Software Moderna**

A arquitetura de software atrela-se ao profissional contendo no eixo da produção de software para arquiteturas modernas. Dotando os usuários com o conhecimento da arquitetura de software moderna.

```
import python
import python

class Arquitetura_de_Software_Moderna:
    ...
    def share(self):
        pass
    ...
    return "Arquitetura de Net", "arquitetureId"
}

def __init__(self):
    if user.is_superuser():
        self.user = user
        self.id = id
        self.name = name
        self.description = description
        self.type = type
        self.shareable = shareable
    ...
    # Envio AI para gerar o layout
    return type
}
resource saabell0
```

**AI-generated system**

A arquitetura com propósito atrela-se ao mecanismo de fusões modernas. Seus sistemas evoluem constantemente incluindo outras tecnologias externas. Chama-se de sistema AI-generated ou gerado por sistema.

**Clean layout**

Gerenciamento de layout é fundamental para garantir que o conteúdo seja legível e organizado. O layout é gerado automaticamente, tornando-o mais fácil de ser lido.

```
graph TD
    User[User] --> Data[Data]
    Data --> Agentes[Agentes Especializados]
    Agentes --> System[System]
    System --> Arch[Architect]
    System --> Dev[Dev]
    System --> Orchest[Orchestration]
    Arch <--> Dev
    Dev <--> Orchest
```

**AI-generated AI-generated system**

**Infográficos feitos por IA**

Adicione Banners Promocionais

Deixe que nossa IA faça o trabalho pesado

Edite em Markdown em Tempo Real

**TESTE AGORA**

PRIMEIRO CAPÍTULO 100% GRÁTIS