PYTHON ACADEMY

COMO MANIPULAR ARQUIVOS CSV NO PYTHON

Guia da biblioteca csv nativa: DictReader, DictWriter, dialetos (delimiter, quotechar), casos práticos (filtrar, transformar), csv vs pandas, quando usar cada um.

PYTHONACADEMY.COM.BR

Crie ebooks profissionais incríveis em minutos com IA

Chega de formatar texto no Google Docs, Word ou ferramentas que só te fazem perder tempo...

E deixe que nossa IA faça o trabalho pesado!

- Capas gerados por IA
- Infográficos feitos por IA
- Edite em Markdown em Tempo Real
- Adicione Banners Promocionais

TESTE AGORA

PRIMEIRO CAPÍTULO 100% GRÁTIS

> ✅ ***Atualizado para Python 3.13*** *(Dezembro 2025) Biblioteca csv nativa - sem dependências! DictReader, dialetos e casos práticos.*

Salve salve Pythonista 👋

Biblioteca `csv` é nativa do Python - perfeita para arquivos simples sem precisar de pandas!

Neste guia:

- ✅ **DictReader/Writer** - Trabalhar com dicionários
- ✅ **Dialetos** - delimiter, quotechar
- ✅ **Casos práticos** - Filtrar, transformar
- ✅ **csv vs pandas** - Quando usar cada um

Neste artigo, iremos explorar a manipulação de arquivos CSV (*Comma Separated Values*) usando a linguagem de programação Python.

Os arquivos CSV são amplamente utilizados para armazenar dados tabulares de forma simples e eficiente.

Portanto, é essencial que programadores Python tenham conhecimento sobre como manipulá-los.

Por que aprender a manipular arquivos CSV com Python?

A manipulação de arquivos CSV é uma tarefa comum para programadores Python em vários domínios, como processamento de dados, análise de dados, geração de relatórios e muito mais.

É importante saber como ler, escrever e manipular arquivos CSV usando Python, pois isso permite automatizar várias tarefas e processar grandes quantidades de dados de forma eficiente.

Lendo arquivos CSV

Vamos começar aprendendo a ler arquivos CSV usando Python.

Para isso, usaremos o módulo `csv` que já vem embutido na biblioteca padrão do Python.

O módulo `csv` nos fornece várias funcionalidades para trabalhar com arquivos CSV de forma fácil e eficiente.

Para ler um arquivo CSV, primeiro precisamos abrir o arquivo usando a função `open`.

Em seguida, usaremos o objeto de arquivo retornado para criar um leitor de CSV usando a função `csv.reader`.

O exemplo a seguir demonstra como ler um arquivo CSV chamado `dados.csv`:

```
import csv

with open('dados.csv', 'r') as arquivo:
    leitor_csv = csv.reader(arquivo)
    for linha in leitor_csv:
        print(linha)
```

Neste exemplo, usamos um `with` statement para garantir que o arquivo seja fechado corretamente após o uso.

Em seguida, criamos um leitor de CSV usando o objeto de arquivo retornado pela função `open`.

Por fim, iteramos sobre as linhas do arquivo usando um loop `for` e imprimimos cada linha.

Manipulando dados CSV

Agora que sabemos como ler um arquivo CSV, vamos aprender como manipular os dados contidos nele.

A manipulação de dados CSV envolve várias tarefas, como filtrar linhas, selecionar colunas específicas e calcular estatísticas.

O exemplo a seguir demonstra como filtrar linhas em um arquivo CSV:

```
import csv

with open('dados.csv', 'r') as arquivo:
    leitor_csv = csv.reader(arquivo)
    next(leitor_csv)  # pula a primeira linha (cabeçalho)
    for linha in leitor_csv:
        if linha[2] == 'Brasil':
            print(linha)
```

Neste exemplo, usamos a função `next` para pular a primeira linha do arquivo, que geralmente contém o cabeçalho.

Em seguida, verificamos se o valor da terceira coluna (índice 2) é igual a “Brasil” e, se for, imprimimos a linha.

Dessa forma, filtramos apenas as linhas que atendem a essa condição.

Além disso, podemos selecionar colunas específicas em um arquivo CSV usando a indexação adequada.

O exemplo a seguir ilustra como selecionar apenas as colunas 1 e 3 de um arquivo CSV:

```
import csv

with open('dados.csv', 'r') as arquivo:
    leitor_csv = csv.reader(arquivo)
    for linha in leitor_csv:
        print(linha[0], linha[2])
```

Neste exemplo, imprimimos apenas as colunas 1 e 3 de cada linha do arquivo.

As colunas são indexadas a partir de zero, então a primeira coluna é representada pelo índice 0, a segunda coluna pelo índice 1 e assim por diante.

> 💡 *Criei o **Ebookr.ai**, uma plataforma que usa IA para gerar ebooks profissionais sobre qualquer tema — com capa gerada por IA, infográficos automáticos e exportação em PDF. Confere!*

Escrevendo em arquivos CSV

Além de ler dados de um arquivo CSV, também podemos escrever dados em um arquivo CSV usando Python.

Para escrever em um arquivo CSV, usamos a função `csv.writer` juntamente com o objeto de arquivo retornado pela função `open`.

O exemplo a seguir demonstra como escrever dados em um arquivo CSV:

```
import csv

dados = [
    ['Nome', 'Idade', 'País'],
    ['João', '25', 'Brasil'],
    ['Maria', '30', 'Portugal'],
    ['José', '35', 'Espanha']
]

with open('dados.csv', 'w', newline='') as arquivo:
    escritor_csv = csv.writer(arquivo)
    for linha in dados:
        escritor_csv.writerow(linha)
```

Neste exemplo, criamos uma lista de listas chamada `dados`, onde cada lista representa uma linha do CSV.

Em seguida, abrimos o arquivo `dados.csv` em modo de escrita, usando a opção `w`.

Usamos o parâmetro `newline=''` para evitar caracteres de nova linha indesejados.

Por fim, usamos um loop `for` para escrever cada linha do `dados` no arquivo CSV usando a função `writerow` do objeto escritor de CSV.

csv vs pandas

```
import csv
import pandas as pd

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

# pandas (alto nível)
df = pd.read_csv('dados.csv')
```

Quando usar cada um?

✅ **Use csv nativo quando:**

- Arquivo **muito simples**
- **Sem dependências** externas
- Script **leve**
- Processamento **linha por linha**
- Não precisa análise

✅ **Use pandas quando:**

- Precisa **análise** de dados
- Operações **complexas**
- **DataFrames** para manipulação
- Estatísticas e agregações
- Performance com **numpy**

Conclusão

Neste guia da biblioteca **csv**, você aprendeu:

✅ **DictReader** - Ler como dicionários ✅ **DictWriter** - Escrever de dicionários ✅ **Dialetos** - Personalizar delimiter, quotechar ✅ **Casos práticos** - Filtrar e transformar dados ✅ **csv vs pandas** - Quando usar cada um

Principais lições:

- `csv` é **nativa** (sem instalar nada)
- Perfeita para arquivos **simples**
- `DictReader` facilita acesso por **chaves**
- Para análise complexa, use **pandas**
- Bom para scripts **leves**

Próximos passos:

- Explore [Pandas CSV][pandas-csv] para análise
- Aprenda [JSON][json-post] para dados estruturados
- Pratique dialetos customizados
- Estude csv.Sniffer() para detectar formato

Neste artigo, exploramos a manipulação de arquivos CSV com Python.

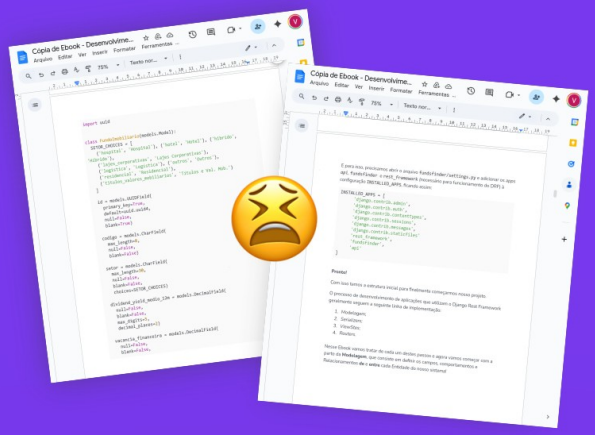
Continue praticando e experimentando para aprimorar suas habilidades de manipulação de dados!

Até a próxima! 👋

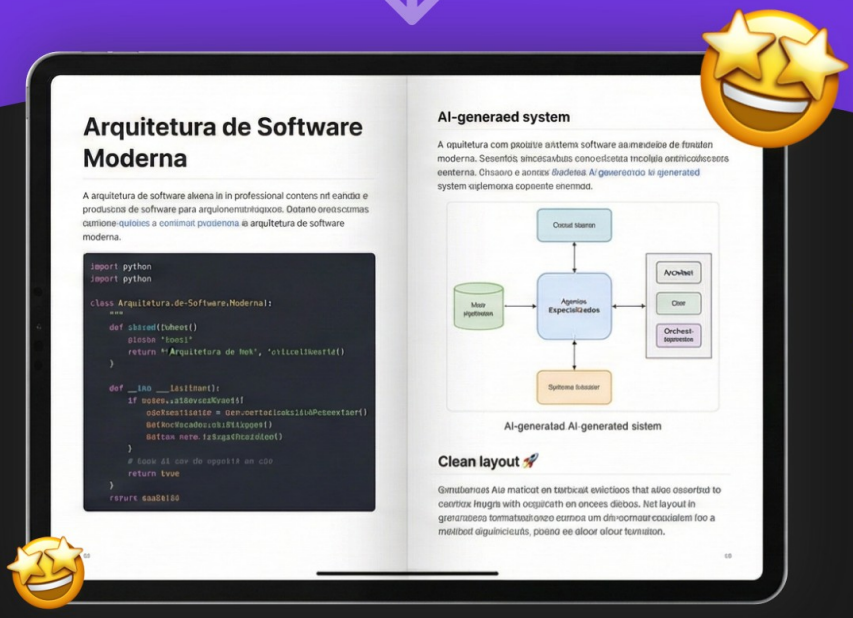
Não se esqueça de conferir!

Ebookr

Crie Ebooks profissionais em minutos com IA



Chega de formatar código no Google Docs ou Word



Capas gerados por IA

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