

# Manipulação de arquivos

**Guilherme Arthur de Carvalho**

Analista de sistemas

**<https://linktr.ee/decarvalhogui>**

# Objetivo Geral

Vamos aprender a importância dos arquivos, como abrir, ler, escrever e gerenciar arquivos em Python. Vamos trabalhar com os formatos txt e csv.

# Pré-requisitos

- ✓ Python e VSCode

# Conteúdo

- ☐ Introdução a manipulação de arquivos
- ☐ Abrindo e fechando arquivos
- ☐ Lendo de um arquivo
- ☐ Escrevendo em um arquivo
- ☐ Gerenciando arquivos e diretórios
- ☐ Tratamento de exceções em manipulação de arquivos

# Conteúdo

- ❑ Boas práticas na manipulação de arquivos
- ❑ Trabalhando com arquivos CSV

# Introdução a manipulação de arquivos

Manipulação de arquivos

# Por que precisamos manipular arquivos?

Os arquivos são essenciais para qualquer tipo de programação, pois fornecem um meio de armazenar e recuperar dados. Através da manipulação de arquivos, podemos persistir os dados além da vida útil de um programa específico.

# Conceito de arquivo em informática

Um arquivo é um container no computador onde as informações são armazenadas em formato digital. Existem dois tipos de arquivos que podemos manipular em Python: arquivos de texto e arquivos binários.



# Abrindo e fechando arquivos

Manipulação de arquivos

# Por que precisamos manipular arquivos?

Para manipular arquivos em Python, primeiro precisamos abri-los. Usamos a função `'open()'` para isso. Quando terminamos de trabalhar com o arquivo, usamos a função `'close()'` para liberar recursos.

# Exemplo de código:




```
1 file = open("example.txt", "r")  
2 # ... fazemos algo com o arquivo ...  
3 file.close()
```

# Modos de abertura de arquivo

Existem diferentes modos para abrir um arquivo, como somente leitura ('r'), gravação ('w') e anexar ('a'). O modo de abertura deve ser escolhido de acordo com a operação que iremos realizar no mesmo.

# Exemplo de código:



```
1  # Para ler um arquivo
2  file = open('example.txt', 'r')
3
4  # Para escrever em um arquivo
5  file = open('example.txt', 'w')
6
7  # Para anexar conteúdo a um arquivo existente
8  file = open('example.txt', 'a')
```

# Lendo de um arquivo

Manipulação de arquivos

# Introdução

Python fornece várias maneiras de ler um arquivo. Podemos usar `'read()'`, `'readline()'` ou `'readlines()'` dependendo de nossas necessidades.

# Método read:



```
1  # Ler todo o conteúdo do arquivo de uma vez  
2  file = open('example.txt', 'r')  
3  print(file.read())  
4  file.close()
```



# Método `readline` e `readlines`

O método `'readline()'` lê uma linha por vez, enquanto `'readlines()'` retorna uma lista onde cada elemento é uma linha do arquivo.

# Exemplo de código:



```
1  # Ler todo o conteúdo do arquivo de uma vez  
2  file = open('example.txt', 'r')  
3  print(file.read())  
4  file.close()
```

# Escrevendo em um arquivo

Manipulação de arquivos

# Introdução

Podemos usar `'write()'` ou `'writelines()'` para escrever em um arquivo. Lembre-se, no entanto, de abrir o arquivo no modo correto.

# Exemplo de código:



```
1 file = open('example.txt', 'w')  
2 file.write("Olá, mundo!")  
3 file.close()
```

# Gerenciando arquivos e diretórios

Manipulação de arquivos

# Introdução

Python também oferece funções para gerenciar arquivos e diretórios. Podemos criar, renomear e excluir arquivos e diretórios usando os módulos 'os' e 'shutil'.

# Exemplo de código:

```
1 import os
2 import shutil
3
4 # Criar um diretório
5 os.mkdir("exemplo")
6
7 # Renomear um arquivo
8 os.rename("old.txt", "new.txt")
9
10 # Remover um arquivo
11 os.remove("unwanted.txt")
12
13 # Mover um arquivo
14 shutil.move("source.txt", "destination.txt")
```



# Tratamento de exceções em manipulação de arquivos

Manipulação de arquivos

# Introdução

Tratar erros é uma parte importante da manipulação de arquivos. Python oferece uma variedade de exceções que nos permitem lidar com erros comuns.

# Exceções mais comuns

- **FileNotFoundError:** Lançada quando o arquivo que está sendo aberto não pode ser encontrado no diretório especificado.
- **PermissionError:** Lançada quando ocorre uma tentativa de abrir um arquivo sem as permissões adequadas para leitura ou gravação.

# Exceções mais comuns

- **IOError**: Lançada quando ocorre um erro geral de E/S (entrada/saída) ao trabalhar com o arquivo, como problemas de permissão, falta de espaço em disco, entre outros.
- **UnicodeDecodeError**: Lançada quando ocorre um erro ao tentar decodificar os dados de um arquivo de texto usando uma codificação inadequada.

# Exceções mais comuns

- **UnicodeEncodeError:** Lançada quando ocorre um erro ao tentar codificar dados em uma determinada codificação ao gravar em um arquivo de texto.
- **IsADirectoryError:** Lançada quando é feita uma tentativa de abrir um diretório em vez de um arquivo de texto.

# Exemplo de código:



```
1 try:
2     file = open('non_existent_file.txt', 'r')
3 except FileNotFoundError:
4     print("Arquivo não encontrado.")
```

# Boas práticas na manipulação de arquivos

Manipulação de arquivos

# Introdução

Ao manipular arquivos em Python existem algumas boas práticas que podemos seguir, vamos abordar as principais.



# Bloco with

Use o gerenciamento de contexto (context manager) com a declaração 'with'. O gerenciamento de contexto permite trabalhar com arquivos de forma segura, garantindo que eles sejam fechados corretamente, mesmo em caso de exceções.

# Exemplo de código:



```
1 with open('arquivo.txt', 'r') as arquivo:  
2     # Faça operações de leitura/gravação no arquivo
```

# Verifique se o arquivo foi aberto com sucesso

É recomendado verificar se o arquivo foi aberto corretamente antes de executar operações de leitura ou gravação nele.

# Exemplo de código:

```
1 try:
2     with open('arquivo.txt', 'r') as arquivo:
3         # Faça operações de leitura/gravação no arquivo
4 except IOError:
5     print('Não foi possível abrir o arquivo.')
```

# Use a codificação correta

Certifique-se de usar a codificação correta ao ler ou gravar arquivos de texto. O argumento 'encoding' da função 'open()' permite especificar a codificação.

# Exemplo de código:



```
1 with open('arquivo.txt', 'r', encoding='utf-8') as arquivo:  
2     # Operações de leitura com codificação UTF-8  
3  
4 with open('arquivo.txt', 'w', encoding='utf-8') as arquivo:  
5     # Operações de escrita com codificação UTF-8
```

# Trabalhando com arquivos CSV

Manipulação de arquivos

# Introdução

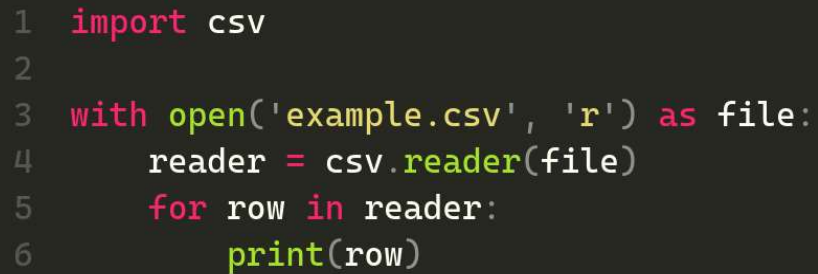
Vamos aprender sobre arquivos CSV, um formato de arquivo amplamente utilizado para armazenar dados tabulares. CSV é a sigla para 'Comma Separated Values'.



# Lendo arquivos CSV

Python fornece um módulo chamado 'csv' para lidar facilmente com arquivos CSV.

# Exemplo de código:



```
1 import csv
2
3 with open('example.csv', 'r') as file:
4     reader = csv.reader(file)
5     for row in reader:
6         print(row)
```

# Escrevendo em arquivos CSV

Da mesma forma podemos utilizar o módulo 'csv' para escrever em arquivos CSV.

# Exemplo de código:

```
1 import csv
2
3 with open('example.csv', 'w', newline='') as file:
4     writer = csv.writer(file)
5     writer.writerow(["nome", "idade"])
6     writer.writerow(["Ana", 30])
7     writer.writerow(["João", 25])
```

# Práticas recomendadas

- Usar `csv.reader` e `csv.writer` para manipular arquivos CSV.
- Fazer o tratamento correto das exceções.
- Ao gravar arquivos CSV definir o argumento `newline=''` no método `'open'`.

# Dúvidas?

> Fórum/Artigos - <https://web.dio.me/articles>

# Exemplo de código:

Insira sua imagem dentro deste espaço  
(retire o retângulo azul, ele deverá ser utilizado  
somente para referência)

