Arquivos

Arquivos

▼ Abrir e fechar arquivos (**open( )** e **close( )**)

- A função `open()` abre um arquivo e retorna um objeto de arquivo. O método `close()` fecha o arquivo após o uso
- Exemplo:

```
arquivo = open("exemplo.txt", "w")  # Abre um arquivo para escrita ("v
arquivo.write("Olá, mundo!")  # Escreve no arquivo
arquivo.close()  # Fecha o arquivo
```

> ! **Sempre** que `open()` for utilizado para abrir um arquivo, é importante lembrar de fechar o arquivo ao término das operações, utilizando `close()`

▼ Context Manager (**with open( )**)

- Uma alternativa para não precisar utilizar `open()` e `close()` em sequência, é utilizar `with open()` no lugar, pois essa instrução fecha o arquivo automaticamente ao final das operações
- Exemplo de uso:

```
with open("exemplo.txt", "r") as arquivo:
    print(arquivo.readline())  # Lê todo o conteúdo do arquivo
# Arquivo é encerrado ao final
```

- Caso o Context Manager não fosse utilizado, seria necessário lembrar de utilizar para encerrar o arquivo ao final. Exemplo:

```
open("exemplo.txt", "r") as arquivo:
print(arquivo.readline())  # Lê todo o conteúdo do arquivo
arquivo.close()  # Fecha o arquivo
```

▼ Modos de abertura - tabela (**“w”**, **“r”**, **“a”**, **“x”**, **“b”**, **“t”**, **“+”**)

Modo	Descrição
`"r"`	Leitura (erro se o arquivo não existir)
`"w"`	Escrita (cria ou sobrescreve o arquivo). Cada vez que for executado, o arquivo será reescrito do 0
`"a"`	Adiciona conteúdo ao final do arquivo. Se o arquivo não existir, ele é criado
`"x"`	Cria um novo arquivo. Se o arquivo já existir, ocorre um erro
`"b"`	Modo binário (`"rb"` , `"wb"` , etc.)
`"t"`	Modo texto (padrão)
`"r+"`	Leitura e escrita (erro se o arquivo não existir) `"w+"` Escrita e leitura (cria ou sobrescreve) `"a+"` Adição e leitura (cria se não existir)
`"w+"`	Escrita e leitura (cria ou sobrescreve)
`"a+"`	Adição e leitura (cria se não existir)

▼ Métodos para leitura de arquivos (**read( )**, **readline( )** e **readlines( )**)

- Para realizar a leitura de arquivos, existem métodos como:
  - `read()` para ler o conteúdo inteiro do arquivo:
  - `readline()` para ler uma linha por vez do arquivo:
  - `readlines()` realiza a leitura, retornando uma lista de linhas:
- Exemplo de uso:

```
with open("exemplo.txt", "r") as arquivo:
    print(arquivo.readline())  # Lê a primeira linha
    print(arquivo.readlines())  # Retorna todas as linhas em uma lista
```

▼ Métodos para leitura de arquivos (**write("texto")** e **writelines(lista)**)

- Para realizar a escrita em arquivos, existem métodos como:
  - `write()` para escrever um texto específico no arquivo:
  - `writelines()` para escrever uma lista de strings, podendo ser utilizada uma em cada linha do arquivo:
- Exemplo de uso:

```
with open("exemplo.txt", "w") as arquivo:
    print(arquivo.write("Primeira linha\n"))
    print(arquivo.writelineslines(["Segunda linha\n", "Terceira linha\n"])
```

- Também é possível utilizar `"w"` para adicionar conteúdo ao final do arquivo sem apagá-lo:

```
with open("exemplo.txt", "a") as arquivo:
    print(arquivo.write("Nova linha adicionada!\n"))
```

▼ Mover o cursor de leitura/escrita (**seek( )**)

- O método `open()` é utilizado para mover o cursor de leitura/escrita dentro de um arquivo aberto. Isso permite controlar exatamente de onde a próxima operação de leitura ou escrita será realizada
- Sintaxe: `seek(offset, whence)`
  - `offset`: Indica o número de bytes a serem movidos
  - `whence`: Define a posição de referência para a movimentação:
    - `0`: Início do arquivo (padrão).
    - `1`: Posição atual do curso
    - `2`: Fim do arquivo

> 💡 É utilizado principalmente o `seek(0, 0)`, sendo usado em casos de retornar o cursor de leitura/escrita para o topo do arquivo

- Exemplo de uso do `seek(0, 0)`:

```
with open("exemplo.txt", "w") as arquivo:  # Criando um arquivo de ex
    arquivo.write("Linha 1\nLinha 2\nLinha 3")

# Abrindo o arquivo para leitura
with open("exemplo.txt", "r") as arquivo:
    arquivo.readline()  # Lê a primeira linha
    arquivo.seek(0, 0)  # Move o cursor para o início do arquivo
    arquivo.readline()  # Lê a primeira linha novamente
```

▼ Definir a codificação de caracteres (*encoding="utf-8"*)

- O parâmetro *encoding* no `with open()` define a codificação de caracteres utilizada ao ler ou escrever um arquivo. Isso é essencial para garantir que caracteres especiais sejam interpretados corretamente, especialmente em diferentes sistemas operacionais

> ❗ *encoding = "utf-8"* é o padrão na maioria das situações, sendo recomendado para compatibilidade com caracteres especiais, como ç ou ã

- Exemplo de sintaxe:

```
with open("exemplo.txt", "w", encoding="utf-8") as arquivo:
    # Operações com o arquivo
```