Persistência de Dados

Prof. Gabriel Rodrigues Caldas de Aquino

gabrielaquino@ic.ufrj.br
Instituto de Computação - Universidade Federal do Rio de Janeiro

Compilado em:
October 15, 2025

Persistência de Dados

Problema

Os dados gerados em uma execução do código são perdidos quando o programa é encerrado.

Solução

Se precisamos usar os dados no futuro, é necessário armazená-los de forma persistente.

Mecanismos de Persistência de Dados:

- Arquivos (TXT, CSV, JSON, XML)
- Bancos de dados (SQLite, PostgreSQL, MySQL)

Persistência em Arquivos de Texto

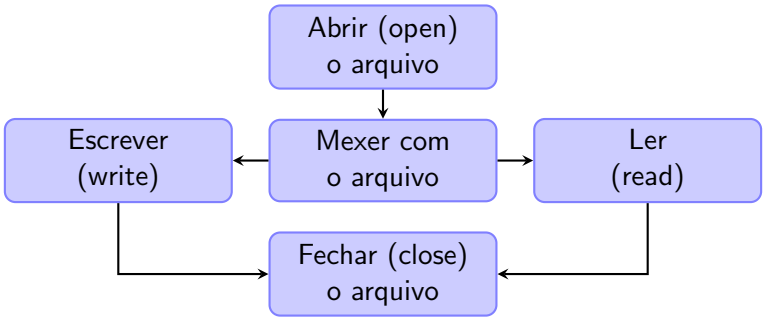
Características

- Formatos comuns: `.txt`, `.csv`, `.json`
- Armazenamento disco rígido
- Manipulação via objetos da classe `File`

Para manipular temos um "protocolo" de Uso

Três etapas: **Abrir**, **Manipular** e **Fechar**

Fluxo para se fazer a manipulação de arquivos



Abrindo Arquivos em Python

Método `open()`

Usado para abrir um arquivo. Requer dois parâmetros principais:

```
arquivo = open("nome_do_arquivo.txt", "modo_de_abertura")
```

Parâmetros

- **Nome do arquivo**:
  - Caminho completo ou relativo
- **Modo de abertura**:
  - `"r"` - Leitura
  - `"w"` - Escrita

Exemplos

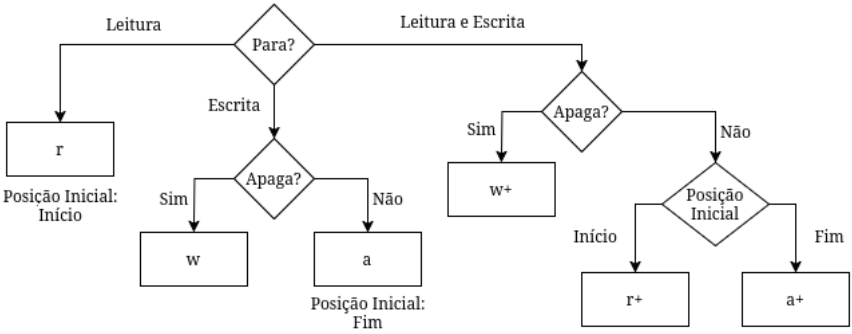
```
# Para escrita
arq = open("dados.txt", "w")

# Para leitura
arq = open("dados.txt", "r")
```

Modos de Abertura de Arquivos

O modo de abertura define como interagiremos com o arquivo:

Modo	**Descrição**	**Existência do Arquivo**
`r`	Leitura apenas	Deve existir
`w`	Escrita	Cria se não existir
`a`	Escrita no final	Cria se não existir
`r+`	Leitura e escrita	Deve existir
`w+`	Leitura e escrita	Cria se não existir
`a+`	Leitura e escrita no final	Cria se não existir

Modos de Abertura de Arquivo



Abrindo Arquivos - Modo de abertura 'r'

- Método read: ler dados de um arquivo que foi previamente aberto para leitura
- Para abrir um arquivo para leitura:
  - Precisamos dar o nome de um arquivo existente
  - Em seguida indicar o modo de leitura *r* no momento da abertura

Abrindo o arquivo dados.txt

```
arquivo = open("dados.txt", "r")
conteudo = arquivo.read()
print(conteudo)
arquivo.close()
```

Lendo linha por linha

- Em arquivos com múltiplas linhas podemos usar for loop para facilitar o tratamento linha por linha

Exemplo de leitura linha por linha

```
arquivo="dados.txt"
arq = open(arquivo, "r")
numero_da_linha = 1
for linha in arq:
    print(f"Linha {numero_da_linha}: {linha}")
    numero_da_linha = numero_da_linha + 1
arq.close()
```

Escrever em Arquivo: write() - Modo de abertura 'w'

Método write()

```
arq.write("conteúdo que será escrito no arquivo")
```

Passos para escrita em arquivo

1. Abrir o arquivo em modo de escrita (`'w'`)
2. Passar o conteúdo como string para `write()`
3. Fechar o arquivo

Escrita Básica em Arquivos

Exemplo de escrita em arquivo

```
arq = open("nomes.txt", "w")
arq.write("Gabriel, Pedro, Manoel")
arq.close()
```

Importante!

- `'w'` = write (escrita)
- Sempre feche o arquivo após escrever
- Cada `write()` grava o conteúdo exato
- Se o arquivo "nomes.txt" **já existe**, ele será **sobrescrito**

Arquivos com mais de uma linha

Leitura de Arquivos Linha por Linha

- Arquivos com múltiplas linhas contêm `\n` escondido

Exemplo Prático

Pedro\n
Antonio\n
Maria\n
Jose\n

Pulando linha na escrita em arquivos

Código de exemplo

```
arq = open("nomes.txt", "w")
arq.write("Gabriel\nPedro\nManoel")
arq.close()
```

O que acontece?

- `\n` é o **caractere especial** para quebra de linha
- Quando escrito no arquivo, ele:
  - Finaliza a linha atual
  - Move o cursor para a próxima linha

Fechando o Arquivo: close()

Método close()

```
arq.close()  # Fecha o arquivo após uso
```

Por que fechar arquivos?

- **Libera recursos do sistema**: Arquivos abertos consomem memória
- **Garante a escrita completa**: Dados podem ficar em buffer
- **Evita corrupção**: Previne acesso concorrente indevido
- **Libera o arquivo**: Permite que outros programas o acessem

Modo de Abertura 'a' (Append)

Funcionamento do modo "a"

```
arquivo = open("dados.txt", "a")  # Modo append
arquivo.write("Novo conteúdo\n")
arquivo.close()
```

Características

- **Abre para escrita** no final do arquivo
- Ponteiro no **fim do arquivo** (só escreve no final)

Gerenciamento de Arquivos com `with open`

Sintaxe Básica

```
with open("arquivo.txt", "modo") as arquivo:
    # Bloco de código
```

Exemplo Prático

```
with open("dados.txt", "r") as arq:
    conteudo = arq.read()
    print(conteudo)
```

Facilidade

- O `with` deixa arquivo aberto.
- Ao sair do bloco, o `close()` é automático.

Controlando a Posição com `seek()`

O que é `seek()`?

Método que permite mover o "cursor" de leitura/escrita para qualquer posição no arquivo

```
arquivo.seek(offset)
```

Parâmetros

- **offset**: Número de bytes para mover

Exemplos

```
# Ir para o byte 10
arquivo.seek(10)
```

Modo de Abertura r+

Funcionamento do modo "r+"

```
with open("arquivo.txt", "r+") as arquivo:
    # Operações de leitura E escrita
    conteudo = arquivo.read()  # Lê
    arquivo.write("novo texto")  # Escreve
```

Características

- Permite **leitura e escrita** no mesmo arquivo
- Não apaga o arquivo na hora de abrir
- Posição inicial: **início do arquivo**

Modo de Abertura w+ (Escrita e Leitura)

Funcionamento do modo "w+"

```
with open("arquivo.txt", "w+") as arquivo:
    arquivo.write("Conteúdo inicial\n")
    arquivo.seek(0)  # Volta ao início para leitura
    conteudo = arquivo.read()
    print(conteudo)
```

Características Principais

- **Apaga conteúdo existente**
- Posição inicial: **início do arquivo**

Cuidado!

Sempre use `seek()` antes de ler após escrever

Modo de Abertura a+ (Append e Leitura)

Funcionamento do modo "a+"

```
with open("arquivo-teste.txt", "a+") as arquivo:
    arquivo.write("Nova entrada\n")  # Escreve no FINAL
    arquivo.seek(0)                  # Volta ao início
    texto = arquivo.read()       # Lê todo conteúdo
    print(texto)
```

Características

- **Não apaga** conteúdo existente
- Posição inicial: **Final do arquivo**

Tratamento de Exceções com Arquivos

Por que tratar exceções?

Lidar com arquivos pode lançar exceções inesperadas que devem ser gerenciadas.

Principais exceções com arquivos

- `FileNotFoundError`: Arquivo não existe ou caminho incorreto
- `IOError`: Erros gerais de entrada/saída (disco cheio, permissões)

Tratamento de Exceções com Arquivos

Código de Exemplo

```
try:
    f = open("arquivo-inexistente.txt", "r")
    conteudo = f.read()
    f.close()
    print(conteudo)
except FileNotFoundError:
    print("Arquivo não existente")
```

Demonstração: Tratando Arquivo Corrompido

Código de Tratamento

```
try:
    f = open("arquivo-corrompido.txt", "r")
    conteudo = f.read()
    f.close()
    print(conteudo)
except IOError:
    print("Erro: Problema ao ler o arquivo")
```