Manipulação de Arquivos em Python

✨Introdução

Manipular arquivos é essencial quando queremos salvar dados de forma permanente em um sistema. Ao contrário das variáveis, que são armazenadas temporariamente na memória RAM, os arquivos podem guardar informações mesmo após o programa ser encerrado.

Nesta apostila, você vai revisar cinco conteúdos fundamentais sobre leitura, gravação, atualização e adição de dados em arquivos `.txt`, com explicações detalhadas e comentadas.

🦜Conteúdo 1: Leitura de Arquivo Simples

🦗Objetivo:

Aprender a abrir e ler o conteúdo de um arquivo `.txt` simples.

🐒Código:

```
# leitura de arquivo
with open("texto.txt", "r", encoding="utf-8") as f:
    texto = f.read()

# saída de dados
print(texto)
```

💐Explicação:

- `open("texto.txt", "r")`: abre o arquivo em modo de leitura.
- `encoding="utf-8"`: evita problemas com acentos.
- `with`: garante que o arquivo será fechado automaticamente.
- `f.read()`: lê o conteúdo completo do arquivo.

🦜Resultado esperado:

O conteúdo do arquivo é exibido no terminal.

🦜Conteúdo 2: Abrir Arquivo com Nome Informado pelo Usuário

🦗Objetivo:

Permitir ao usuário digitar o nome de um arquivo para abri-lo.

🐒 Código:

```
import os

while True:
    try:
        arquivo = input("Informe o nome do arquivo (sem extensão):
").strip().lower()

        with open(f"{arquivo}.txt", "r", encoding="utf-8") as f:
            arquivo_aberto = f.read()
        os.system("cls" if os.name == "nt" else "clear")

        print(arquivo_aberto)
        while True:
            prosseguir = input("Deseja abrir outro arquivo? (s/n):
").strip().lower()
            if prosseguir == "s" or prosseguir == "n":
                break
            else:
                print("Opção inválida.")
                continue
        match prosseguir:
            case "s":
                continue
            case "n":
                break
    except Exception as e:
        print(f"Não foi possível ler o arquivo. {e}.")
        continue
```

💐 Explicação:

- Usa `input()` para permitir ao usuário escolher o arquivo.
- `os.system("cls")`: limpa a tela no Windows ou Linux.
- `match-case`: decide se continua ou encerra.
- `try/except`: previne travamentos se o arquivo não existir.

🦜 Conteúdo 3: Gravação de Novo Arquivo

🦗 Objetivo:

Criar um novo arquivo `.txt` com nome e conteúdo digitados pelo usuário.

🐒 Código:

```
import os
```

```
while True:
    try:
        novo_texto = input("Digite o texto:\n")
        nome_arquivo = input("Dê o nome do arquivo (sem extensão):
").strip().lower()
        with open(f"44_manipular_arquivo_parte_03/{nome_arquivo}.txt", "w",
encoding="utf-8") as f:
            f.write(novo_texto)
        os.system("cls" if os.name == "nt" else "clear")
        print(f"{nome_arquivo}.txt gravado com sucesso.")
        while True:
            prosseguir = input("Deseja gravar novo arquivo? (s/n):
").strip().lower()
            if prosseguir == "s" or prosseguir == "n":
                break
            else:
                print("Opção inválida.")
                continue
        match prosseguir:
            case "s":
                continue
            case "n":
                break
    except Exception as e:
        print(f"Não foi possível gravar arquivo. {e}.")
        continue
```

💡 Explicação:

- Usa o modo "w" para gravar (sobrescreve se o arquivo existir).
- Salva arquivos dentro de uma pasta específica.
- Permite criar vários arquivos em sequência.

🦜 Conteúdo 4: Atualizar (Sobrescrever) Conteúdo de um Arquivo

🦗 Objetivo:

Mostrar o conteúdo atual de um arquivo e permitir sua substituição completa.

🐒 Código:

```
try:
    arquivo = input("Informe o nome do arquivo (sem extensão):
").strip().lower()
    with open(f"{arquivo}.txt", "r", encoding="utf-8") as f:
        texto = f.read()
    print(texto)
```

```
    novo_texto = input("Digite o texto:\n")

    with open(f"{arquivo}.txt", "w", encoding="utf-8") as f:
        f.write(novo_texto)
except Exception as e:
    print(f"Não foi possível atualizar arquivo. {e}.")
```

💐 Explicação:

- Exibe o conteúdo do arquivo antes de sobrescrever.
- Usa o modo "w" para apagar o texto antigo e gravar o novo.

🦜 Conteúdo 5: Acrescentar Texto ao Final do Arquivo

🦗 Objetivo:

Manter o texto anterior e adicionar um novo ao final do arquivo.

🐒 Código:

```
try:
    arquivo = input("Informe o nome do arquivo: ").strip().lower()
    with open(f"{arquivo}.txt", "r", encoding="utf-8") as f:
        texto = f.read()
    print(f"Texto gravado:\n{texto}")

    novo_texto = input("Digite o novo texto:\n")
    nova_gravacao = f"{texto}\n{novo_texto}"

    with open(f"{arquivo}.txt", "w", encoding="utf-8") as f:
        f.write(nova_gravacao)
    print("Gravação feita com sucesso.")

    with open(f"{arquivo}.txt", "r", encoding="utf-8") as f:
        texto_final = f.read()
    print(f"Texto final: {texto_final}")
except Exception as e:
    print(f"Não foi possível atualizar o conteúdo. {e}.")
```

💐 Explicação:

- Lê o texto existente.
- Cria uma nova string combinando o texto antigo + o novo com quebra de linha.
- Grava essa combinação no arquivo usando o modo "w".
- É uma forma de simular o modo de adição.

🐟Conclusão

Com esses cinco conteúdos, você agora domina: - Leitura de arquivos existentes - Gravação de arquivos novos - Sobrescrita de conteúdo - Adição de novo conteúdo - Tratamento de erros e interação com o usuário

Continue praticando, experimente criar funções para automatizar essas tarefas, e lembre-se de sempre tratar erros com `try/except` para evitar travamentos no programa.

Bons estudos! 🚀