Manipulação de Arquivos em Python – Um Guia Prático para Cientistas de Dados

Olá! Bem-vindo(a) à nossa newsletter semanal sobre Ciência de Dados!

Como cientistas de dados, nosso trabalho muitas vezes envolve a manipulação de grandes volumes de dados. Não raramente, nos deparamos com arquivos que precisam ser ajustados manualmente antes de serem usados em ferramentas como **Pandas** ou **Polars**. A **linguagem Python** oferece uma série de funcionalidades nativas para manipulação de arquivos em um nível mais "baixo", e entender como utilizá-las pode ser muitas vezes fundamental para resolver problemas específicos que as bibliotecas tradicionais não cobrem.

Nesta newsletter, vamos explorar algumas das principais funções de manipulação de arquivos em Python, além de pacotes úteis, como o **OS**. Isso permitirá que você tenha um controle mais granular sobre a leitura, escrita e manipulação de arquivos em seus projetos de ciência de dados.

A tabela abaixo traz os comandos mais utilizados para a manipulação de arquivos em Python, explicando como eles funcionam, suas opções e exemplos práticos para o contexto de ciência de dados.

Comando	Descrição	Opções	Exemplo na Ciência de Dados
`open()`	Abre um arquivo para leitura, escrita ou anexação.	"r": Leitura, "w": Escrita, "a": Anexar, "b": Binário	`file = open('dados.csv', 'r')` Este comando abre o arquivo de dados para leitura e pode ser usado para processar o arquivo linha por linha antes da importação.
`read()`	Lê o conteúdo de um arquivo.	read(size): Lê até o tamanho especificado (em bytes).	`conteudo = file.read(100)` Lê os primeiros 100 caracteres de um arquivo de texto. Isso pode ser útil para pré-visualizar a estrutura dos dados antes de carregar o arquivo inteiro.
`write()`	Escreve dados em um arquivo.	Sem opções diretas, mas deve ser utilizado com modos de abertura como "w" ou "a".	`file.write('nova linha de dados')` Escreve uma nova linha no arquivo de saída, útil quando se quer registrar o progresso ou salvar transformações intermediárias de dados.
`readlines()`	Lê todas as linhas do arquivo e retorna como uma lista.	Sem opções, lê todas as linhas.	`linhas = file.readlines()` Permite a leitura de todas as linhas de um arquivo de uma só vez, facilitando a manipulação de arquivos grandes que precisam ser processados linha por linha.
`seek()`	Move o cursor do arquivo para uma posição específica.	seek(offset): Define a posição dentro do arquivo.	`file.seek(0)` Reposiciona o cursor no início do arquivo, útil para reiniciar a leitura após um processamento parcial de dados.
`close()`	Fecha o arquivo.	Sem opções, usado para finalizar a manipulação de arquivos.	`file.close()` Fecha o arquivo após a leitura ou gravação, garantindo que todas as operações de escrita tenham sido concluídas.
`os.rename()`	Renomeia um arquivo ou diretório.	os.rename(src, dst)	`os.rename('antigo.csv', 'novo.csv')` Usado para renomear arquivos de dados após o processamento ou limpeza.
`os.remove()`	Remove um arquivo.	os.remove(path)	`os.remove('dados_antigos.csv')` Útil para excluir arquivos de dados obsoletos ou temporários que não são mais necessários.
`os.path.exists()`	Verifica se um arquivo ou diretório existe.	os.path.exists(path)	`if os.path.exists('dados.csv'):` Usado para verificar se o arquivo está disponível antes de iniciar o processamento, evitando erros de "arquivo não encontrado".

Como a Manipulação de Arquivos é Importante para Cientistas de Dados? Muitas vezes, trabalhar diretamente com funções de alto nível como `pandas.read_csv()` não é suficiente. Arquivos corrompidos, grandes ou não estruturados podem exigir uma manipulação mais específica. A habilidade de trabalhar diretamente com funções nativas do Python permite que você ajuste arquivos manualmente, lide com exceções e melhore a qualidade dos dados antes mesmo de carregá-los em suas ferramentas de análise.

Por exemplo, imagine que você receba um arquivo de log de transações em formato **.txt**. Esse arquivo precisa ser filtrado por palavras-chave ou reorganizado antes de ser convertido em um `dataframe`. Usar as funções nativas de leitura e escrita permite que você processe o arquivo linha por linha, limpe dados indesejados e o formate corretamente para o uso posterior.

Além disso, em ambientes de produção, onde dados são continuamente gerados e manipulados, o uso de funções como `os.rename()` ou `os.remove()` pode ser útil para renomear ou excluir arquivos antigos, mantendo um fluxo de trabalho automatizado e limpo.

Dessa forma, dominar a manipulação de arquivos em Python é essencial para garantir que os dados estejam prontos para análise e que todo o processo de tratamento seja otimizado. Ao utilizar essas ferramentas de maneira estratégica, você garante mais flexibilidade, controle e precisão na organização dos dados, o que impacta diretamente na qualidade das suas análises. As possibilidades de automação e melhorias no fluxo de trabalho são vastas, permitindo que você leve seus projetos de ciência de dados a um novo nível.

Se tiver dúvidas ou sugestões, não hesite em entrar em contato. Fique atento à nossa próxima edição, onde continuaremos a explorar as últimas inovações e ferramentas que estão moldando o futuro da Ciência de Dados.

Saudações,

Prof. Dr. Dilermando Piva Jr.
Coordenador de Ciência de Dados para Negócios - Fatec Votorantim
E-mail: f301.cdn@fatec.sp.gov.br