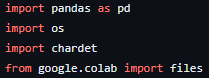
Manual do Usuário - Integração e Processamento de Dados com Python

**Introdução**

Este manual orienta sobre como integrar e processar dados de arquivos CSV usando Python. As etapas a seguir descrevem como carregar arquivos, verificar suas informações, integrar dados de múltiplas fontes e exportar os resultados.

1. Importação de Bibliotecas Necessárias

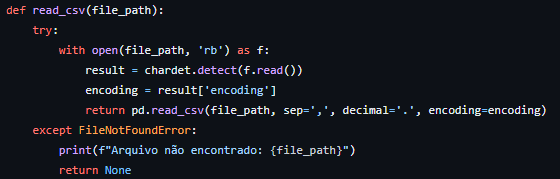


Pandas: Manipulação de dados.

os: Operações no sistema de arquivos. chardet: Detecção de encoding.

files: Carregamento e download de arquivos no Google Colab.

1. Função Para Leitura de Arquivos em CSV



read\_csv: Lê arquivos CSV com detecção automática de encoding e tratamento de erros.

1. Carregar Arquivos do Sistema Local para o Ambiente do Colab



files.upload(): Permite ao usuário carregar arquivos locais para o ambiente do Colab.

1. Verificar os Nomes dos Arquivos Carregados



Identifica os arquivos carregados com base em seus nomes.

1. Abrir os Arquivos Necessários



Utiliza a função read\_csv para abrir os arquivos.

1. Verificar se os DataFrames Foram Carregados Corretamente



Confirma que todos os arquivos foram carregados com sucesso.

1. Exibir Informações sobre o Arquivo de Rotas



Mostra informações detalhadas sobre o DataFrame de rotas.

1. Exibir Informações sobre o Arquivo de Clientes



Mostra informações detalhadas sobre o DataFrame de clientes.

1. Exibir Informações sobre o Arquivo de Fábricas



Mostra informações detalhadas sobre o DataFrame de fábricas.

1. Integrar os DataFrames de Rotas, Clientes e Fábricas



Realiza a integração dos DataFrames com base em colunas comuns.

1. Exibir Informações sobre o DataFrame Final



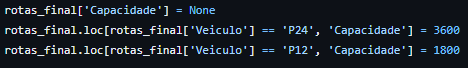
Mostra informações detalhadas sobre o DataFrame final integrado.

1. Visualizar as Primeiras Linhas do DataFrame Final



Exibe as cinco primeiras linhas do DataFrame final para verificação.

1. Criar Nova Coluna com Condição



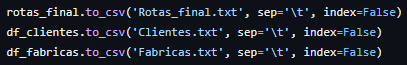
Adiciona uma nova coluna "Capacidade" ao DataFrame final com valores baseados em condições.

1. Criar Nova Coluna com Cálculo



Adiciona uma coluna "Produtividade" calculada a partir de outras colunas.

1. Exportar DataFrames para Arquivos de Texto



Exporta os DataFrames para arquivos de texto com separador de tabulação.

1. Mensagem de Sucesso na Criação dos Arquivos TXT



Informa ao usuário que os arquivos foram criados com sucesso.

1. Baixar os Arquivos para o Sistema Local



Permite ao usuário baixar os arquivos gerados para o sistema local

.

1. Mensagem de Erro ao Carregar Arquivos



Informa ao usuário se houve algum erro ao carregar os arquivos.

# Manual do Usuário - Otimização de Frete Utilizando PuLP

## Este manual fornece uma explicação detalhada do código Python para otimização de frete usando a biblioteca PuLP. O objetivo é guiar o usuário através de cada etapa do processo, desde a instalação da biblioteca até a solução do problema de otimização.

Etapa 1: Instalação da Biblioteca PuLP

Antes de iniciar a otimização, é necessário instalar a biblioteca PuLP, que facilita a modelagem de problemas de otimização linear em Python.



Explicação: O comando !pip install pulp instala a biblioteca PuLP no ambiente Python.

### Etapa 2: Importação da Biblioteca PuLP

Após a instalação, importe a biblioteca PuLP para o seu script.



Explicação: from pulp import \* importa todas as funções e classes da biblioteca PuLP.

### Etapa 3: Definição do Problema de Otimização

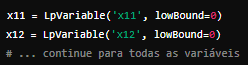
Crie um problema de otimização linear com o objetivo de minimizar os custos.



Explicação: LpProblem('otimizadofrete', LpMinimize) cria um problema de otimização chamado "otimizadofrete" com o objetivo de minimização (LpMinimize).

### Etapa 4: Definição das Variáveis de Decisão

Defina as variáveis de decisão do problema, que representam as quantidades de frete a serem otimizadas.



Explicação: LpVariable('x11', lowBound=0) define uma variável de decisão chamada x11 com um limite inferior de 0, indicando que a quantidade de frete não pode ser negativa.

### Etapa 5: Definição da Função Objetivo

Defina a função objetivo que o modelo deve minimizar. A função objetivo é o custo total de frete.



Explicação: Esta linha adiciona a função objetivo ao problema, somando os custos ponderados pelas variáveis de decisão.

### Etapa 6: Definição das Restrições

Adicione as restrições do problema, que representam as limitações nas quantidades de frete e outras condições.



Explicação: += adiciona uma restrição ao problema. Por exemplo, x11 + x12 + ... + x151 <= 90000000 assegura que a soma dessas variáveis não exceda 90 milhões.

### Etapa 7: Solução do Problema

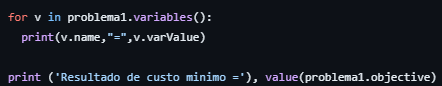
Resolva o problema de otimização usando o solver padrão da PuLP.



Explicação: problema1.solve() resolve o problema de otimização usando o solver padrão.

### Etapa 8: Impressão dos Resultados

Após resolver o problema, imprima os valores das variáveis de decisão e o custo mínimo total.



Explicação: O loop for v in problema1.variables(): itera sobre todas as variáveis do problema, imprimindo seus nomes e valores otimizados.print(‘Resultado de custo mínimo =’, value(problema1.objective)) imprime o valor da função objetivo, que é o custo mínimo total.

**MANUAL DO USUÁRIO - MY SQL 8.0**

Etapa 1: Atualiza a tabela ROTAS e define como ‘CIF’ todas as células da coluna ‘FOB’ cujo ‘Valor do frete’ for maior que 0:

Uma imagem contendo Interface gráfica do usuário

Descrição gerada automaticamente

***Explicação: Este comando UPDATE é usado para modificar dados existentes em uma tabela. Neste caso, estamos atualizando a tabela ROTAS. A cláusula SET define que a coluna FOB será atualizada para o valor 'CIF' nas linhas onde a coluna Valor\_do\_frete for maior que 0. Isso significa que estamos alterando as células da coluna FOB para 'CIF' apenas nas rotas em que há um custo de frete associado.***

Etapa 2: Seleciona e altera a tabela rotas, adicionando a coluna ‘Preço’:

Logotipo, nome da empresa

Descrição gerada automaticamente

***Explicação:******O comando ALTER TABLE é utilizado para modificar a estrutura de uma tabela existente. Neste caso, estamos adicionando uma nova coluna chamada Preço à tabela ROTAS. O tipo de dado DECIMAL(10, 2) indica que esta coluna vai armazenar números decimais com um máximo de 10 dígitos, dos quais 2 são casas decimais. Essa alteração permite que você armazene e manipule informações de preço diretamente na tabela ROTAS.***

Etapa 3: Calcula e atualiza valores da tabela preço:

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente

***Explicação:******Este comando UPDATE é utilizado para modificar dados existentes na coluna Preço da tabela ROTAS. O cálculo Preço = Valor\_unitário \* Quantidade é um exemplo hipotético que demonstra como você pode calcular o valor da coluna Preço com base em outras colunas da mesma tabela. Isso permite atualizar dinamicamente os preços conforme as quantidades e valores unitários mudam.***

Etapa 4: Adiciona a coluna produtividade:

Logotipo

Descrição gerada automaticamente com confiança baixa

***Explicação:******O comando ALTER TABLE novamente é usado para modificar a estrutura da tabela ROTAS. Aqui, estamos adicionando uma nova coluna chamada produtividade à tabela. O tipo de dado INT indica que esta coluna vai armazenar valores inteiros, que podem ser usados para registrar e calcular a produtividade de rotas ou outros aspectos relacionados.***

Etapa 5: Atualiza a tabela rotas e define os valores de produtividade e também o tamanho e capacidade dos veículos:

Interface gráfica do usuário

Descrição gerada automaticamente com confiança média

***Explicação:******Este comando UPDATE é usado para modificar dados existentes na tabela ROTAS. Aqui, estamos definindo os valores das colunas produtividade, tamanho\_veiculo e capacidade\_veiculo com base em cálculos ou valores específicos. A cláusula SET permite que você atualize várias colunas ao mesmo tempo, facilitando a atribuição de informações como produtividade, tamanho e capacidade dos veículos associados às rotas.***

Etapa 6: Calcula a média da coluna ‘Preço’:

Texto

Descrição gerada automaticamente

***Explicação: Este comando SELECT é usado para recuperar dados de uma tabela, neste caso, a tabela ROTAS. A função AVG(Preço) calcula a média dos valores na coluna Preço. O uso do alias Media\_preco renomeia o resultado da média para facilitar a referência posterior. Isso permite que você obtenha rapidamente uma média dos preços registrados nas rotas.***

Etapa 7: Exclui as colunas: emissão, entrega, ano, mês e atualiza a tabela:

Tabela

Descrição gerada automaticamente com confiança média

***Explicação:******O comando ALTER TABLE com a cláusula DROP COLUMN é usado para remover colunas específicas de uma tabela existente. Neste caso, estamos removendo as colunas emissao, entrega, ano e mes da tabela ROTAS. Isso é útil quando essas colunas não são mais necessárias ou estão sendo consolidadas de alguma forma, ajudando a simplificar a estrutura da tabela e reduzir o armazenamento de dados desnecessários.***

Etapa 8: Cria a tabela Soma quantidade por cliente e me traz os valores correspondentes:

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente

***Explicação:******O comando CREATE TABLE ... AS SELECT é usado para criar uma nova tabela, Soma\_quantidade\_por\_cliente, baseada nos resultados de uma consulta SELECT. Neste caso, estamos selecionando o cliente\_id e a soma da coluna quantidade para cada cliente da tabela ROTAS. A cláusula GROUP BY cliente\_id agrupa os resultados por cliente\_id, garantindo que cada linha na nova tabela represente a soma total da quantidade para um cliente específico. Isso é útil para análises que requerem informações agregadas por cliente, facilitando a manipulação e visualização dos dados.***

Parte superior do formulário

Parte inferior do formulário