

## EXTRAÇÃO DE DADOS USANDO PROGRAMAÇÃO língua utilizada Python

"Destaco a aplicação de métodos computacionais na consecução desse feito, utilizando tecnologias contemporâneas. Python, uma língua proeminente no mercado, sobressai-se na manipulação e tratamento eficaz de dados. O projeto em foco visa extrair dados do software adotado pelo Grupo Gruner para atender às demandas do consultório. A extração é realizada por meio de um processo guiado por tutoriais conduzidos pelo Dr. José Barbosa, nosso supervisor no setor de inteligência, responsável pelo BackOffice, cujo propósito é controlar e diligenciar contas médicas.

O sistema em questão, denominado Amigo, é um ecossistema de soluções destinado a aprimorar a gestão, permitindo mais tempo dedicado ao cuidado dos pacientes. Embora o sistema e suas funcionalidades sejam notáveis, é crucial destacar a importância de sua configuração e parametrização para garantir seu desempenho ótimo. Isso se deve à sua adaptabilidade a diversos cenários, os quais podem sofrer alterações frequentes, demandando ajustes e correções.

O propósito do código é automatizar o processamento de um arquivo em formato xlsx gerado pelo sistema. Utilizando programação em Python, a importação, manipulação e ajustes são realizados de maneira direta e eficiente, entregando um arquivo seguro em conformidade com as normas de tratamento de dados da LGPD. Este processo minimiza erros de digitação, esquecimentos de lançamentos futuros e possibilita a geração de relatórios futuros de dados, promovendo uma resposta ágil por meio de métodos computacionais.

Ao centralizar a lógica no código, reduzimos a necessidade de ajustes constantes, ao mesmo tempo em que oferecemos a flexibilidade necessária para lidar com mudanças nos requisitos. O resultado é uma solução que aproveita o poder da programação e da IDE para efetuar tratamentos complexos e exportar dados de maneira eficiente, inclusive em formatos como xlsx (Excel)."

### Partindo para o Código

vou explicar o código ponto a ponto. Antes de começar, este código em Python realiza várias operações em um conjunto de dados em formato de planilha Excel (.xlsx) usando a biblioteca **pandas**. Ele também utiliza a biblioteca **openpyxl** para lidar com a leitura e escrita de arquivos Excel. Vamos analisar cada parte do código:

**# Importação das bibliotecas necessárias**

```
import openpyxl
```

```
import pandas as pd
```

```
import codecs # Módulo para realizar a transformação ROT13
```

Importação das bibliotecas necessárias: **openpyxl** para lidar com arquivos Excel, **pandas** para manipulação de dados e **codecs** para realizar a transformação **ROT13**.

### # Caminho do arquivo (arquivo local se atentar a essa configuração)

```
caminho_arquivo = "/home/henrique/Downloads/Relatório_Completo.xlsx"
```

O relatório amigo configurado para nossa realidade faz uso de 55 colunas onde diretamente no código definimos quais são de nossa necessidade como mostra o código abaixo faremos a extração para nossa base apenas 18 colunas que serão tratadas pelo pandas em seu dataframe.

### # Colunas desejadas

```
colunas_desejadas = [  
    'Data Atendimento', 'Código Atendimento', 'Paciente', 'Idade', 'ID amigo', 'Profissional',  
    'Forma de Pagamento', 'Item', 'Quantidade', 'Valor total do item R$', 'Número lote',  
    'Data envio', 'Data pagamento', 'Valor pago R$', 'Valor glosado R$', 'Mês pagamento',  
    'Recurso'  
]
```

O código mostrado Define as colunas que serão utilizadas no DataFrame.

### # Leitura do arquivo Excel com o uso explícito do engine 'openpyxl'

```
df = pd.read_excel(caminho_arquivo, usecols=colunas_desejadas, engine='openpyxl')
```

Lê o arquivo Excel especificado e carrega as colunas desejadas em um DataFrame usando pandas.

### # Mapeamento de nomes de itens para 'Consulta Particular'

```
consulta_mapping = {  
    'consulta a1': 'Consulta Particular', 'consulta 01': 'Consulta Particular',  
    'consulta 02': 'Consulta Particular', 'consulta 03': 'Consulta Particular'  
}
```

### # Substituição dos nomes de itens

```
df['Item'] = df['Item'].str.lower().map(consulta_mapping).fillna(df['Item'])
```

A configuração acima faz parte de uma particularidade proposta em resolver diferentes valores variados de um mesmo produto temos n/Consultas e diferentes valores logo o código analisa a coluna em questão e Substitui os nomes de alguns itens na coluna '**Item**' por '**Consulta Particular**' usando um mapeamento.

### # Mapeamento de nomes de profissionais para CRMs

```
crm_mapping = {  
    'NOME DO MÉDICO X': 'CRM: XXXX', 'NOME DO MÉDICO Y': 'CRM: YYYY',  
    'NOME DO MÉDICO WWW': 'CRM: WWW', 'NOME DO MÉDICO @@@@': 'CRM:  
    @@@@'  
}
```

### # Substituição dos nomes pelo CRM

```
df['Profissional'] = df['Profissional'].map(crm_mapping)
```

O relatório exibe no campo profissional a extensão de todo o nome do profissional executante afim de amenizar a exposição de seu nome tratamos junto ao código a transferência de seu nome pelo seu CRM passando a ser mostrado apenas a numeração usando um mapeamento.

### # Ajuste do formato da coluna 'Idade' para exibir apenas a parte inteira

```
df['Idade'] = df['Idade'].astype(float).astype(pd.Int32Dtype(), errors='ignore')
```

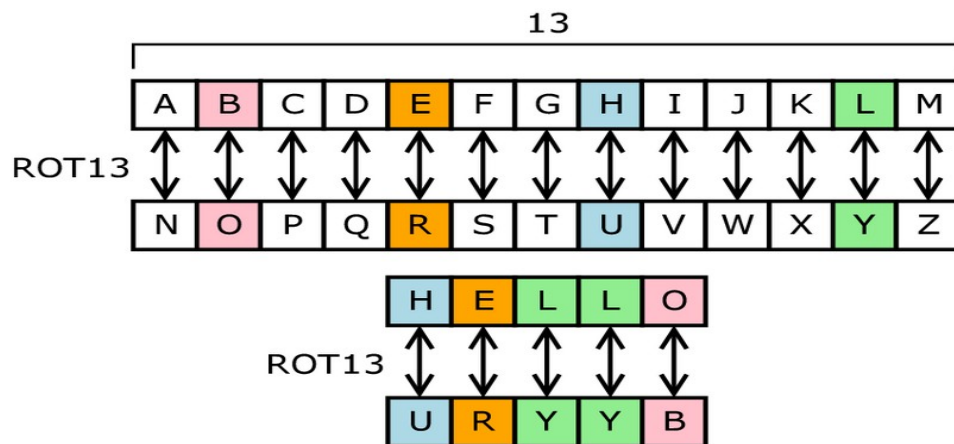
Ajusta o formato da coluna 'Idade' para exibir apenas a parte inteira.

O valor exibido no relatório é ex: 46.0 assim a tratativa passa a usar apenas a parte inteira como foi mencionado exibindo 46.

### # Descaracterização da coluna 'Paciente' usando ROT13

```
df['Paciente'] = df['Paciente'].apply(lambda x: codecs.encode(x, 'rot_13'))
```

Usa o algoritmo ROT13 para "descaracterizar" a coluna 'Paciente' (criptografia simples).



O código usa o algoritmo **ROT13** para "descaracterizar" a coluna 'Paciente' uma forma de realizar uma (criptografia simples).

### # Criação da coluna 'Forma de Cobrança'

```
df['Forma de Cobrança'] = df['Forma de Pagamento'].apply(lambda x: 'Gruner' if any(keyword in x for keyword in [...]) else [...])
```

Cria uma nova coluna '**Forma de Cobrança**' baseada nas informações da coluna '**Forma de Pagamento**'.

### # Adicionando regra para 'dinheiro' na coluna 'Forma de Cobrança'

```
df.loc[df['Forma de Pagamento'].str.contains('dinheiro', case=False), 'Forma de Cobrança']  
= 'Gruner'
```

Ao longo de mais algumas linhas você irá ver mencionar diferentes formas de cobrança, a clínica executa consultas particulares logo a fonte pagadora é local e seu recebimento é direto como mostra Gruner, esse é o objetivo. Adiciona uma regra específica para 'dinheiro' na coluna 'Forma de Cobrança'.

### # Criação da coluna 'Forma de Cobrança' com a regra adicional

```
df['Forma de Cobrança'] = df.apply(lambda row: 'CFH' if row['Profissional'] == 'CRM:  
@@@@' else row['Forma de Cobrança'], axis=1)
```

O profissional em questão mencionado junto a regra tem todas as suas cobranças destinadas a um faturamento terceirizado que logo é diferente dos demais então a regra consiste em manter a leitura visual de forma destacada logo ao usar um alto filtro em uma planilha ao mostrar CFH logo representará todas as contas referentes ao CRM @@@@.

### # Cabeçalho extra

```
cabecalho_extra = "Gruner - Grupo de Neurocirurgia do Recife"
```

Define um cabeçalho extra.

### # Configurações de formatação para melhorar a visualização

```
pd.set_option('display.max_columns', None)  
pd.set_option('display.expand_frame_repr', False)
```

Configura opções de formatação para melhorar a visualização do DataFrame.

### # Exibição do cabeçalho extra

```
print(cabecalho_extra.center(df.shape[1] * 20))
```

Exibe o cabeçalho extra centralizado na largura total do DataFrame.

### # Exibição da planilha com as colunas desejadas

```
print(df.to_string(index=False, justify='center'))
```

Exibe o DataFrame formatado para uma melhor visualização.

### # Exibição da contagem de linhas

```
print(f'\nTotal de Linhas: {len(df)}')
```

Ao ler a planilha em excel para afirmar a totalidade de informação captada essa ideia consiste em informar quantas linhas foi extraída assim você poderá fazer o comparativo. Assim é exibida a informação total de linhas do Dataframe.

### # Pergunta ao usuário sobre a criação do arquivo CSV

```
resposta_csv = input("\nDeseja criar um arquivo CSV? (Digite 'sim' ou 'não'): ").lower()
```

Solicita ao usuário se deseja criar um arquivo CSV.

```
if resposta_csv == 'sim':
```

### # Pergunta ao usuário sobre a criação do arquivo descriptografado

```
resposta_descriptografado = input("\nDeseja criar um arquivo descriptografado? (Digite 'sim' ou 'não'): ").lower()
```

Se o usuário optar por criar um arquivo CSV, pergunta se deseja criar um arquivo descriptografado.

### # Caminho para o arquivo CSV

```
caminho_csv = f"/home/henrique/Downloads/Relatorio_Completo_{'Descriptografado' if resposta_descriptografado == 'sim' else 'Criptografado'}.csv"
```

Define o caminho para o arquivo CSV com base na escolha do usuário.

### # Salvando o DataFrame como arquivo CSV

```
df.to_csv(caminho_csv, index=False)  
print(f"Arquivo CSV criado em: {caminho_csv}")
```

Salva o DataFrame como um arquivo CSV e exibe a mensagem de confirmação.

```
if resposta_descriptografado == 'sim':
```

### # Descaracterização da coluna 'Paciente' usando ROT13 reverso (descriptografando)

```
df['Paciente'] = df['Paciente'].apply(lambda x: codecs.decode(x, 'rot_13'))
```

Se o usuário optar por criar um arquivo descriptografado, reverte a descaracterização feita anteriormente.

### # Caminho para o arquivo Excel descriptografado

```
caminho_excel_descriptografado =  
"/home/henrique/Downloads/Relatorio_Descriptografado.xlsx"
```

### # Salvando o DataFrame descriptografado como arquivo Excel

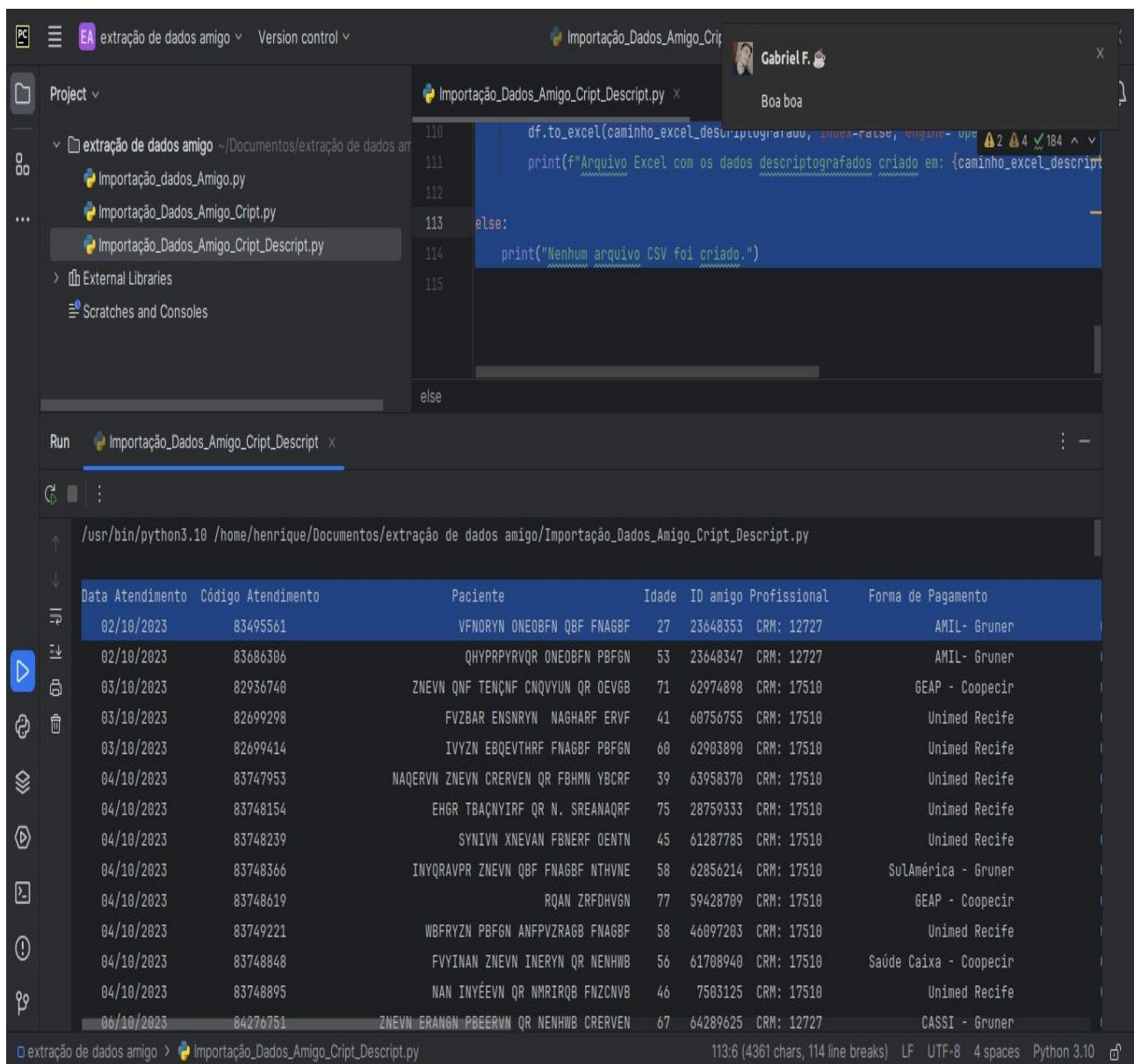
```
df.to_excel(caminho_excel_descriptografado, index=False, engine='openpyxl')  
print(f"Arquivo Excel com os dados descriptografados criado em:  
{caminho_excel_descriptografado}")
```

Salva o DataFrame descriptografado como um arquivo Excel e exibe a mensagem de confirmação.

else:

```
print("Nenhum arquivo CSV foi criado.")
```

Se o usuário optar por não criar um arquivo CSV, exibe uma mensagem informando que nenhum arquivo foi criado. Esse código realiza diversas operações, incluindo leitura de dados de um arquivo Excel, manipulação de colunas, criação de novas colunas com base em regras, exibição formatada de dados e opções para salvar os resultados em arquivos CSV e Excel.



The screenshot shows a Python IDE with a project named "extração de dados amigo". The file explorer on the left shows the project structure, including files like "Importação\_dados\_Amigo.py", "Importação\_Dados\_Amigo\_Cript.py", and "Importação\_Dados\_Amigo\_Cript\_Descript.py". The main editor displays the code for "Importação\_Dados\_Amigo\_Cript\_Descript.py", which includes a function to read an Excel file and print the results. The console output shows the execution of the script, displaying a table of data with columns: Data Atendimento, Código Atendimento, Paciente, Idade, ID amigo Profissional, and Forma de Pagamento. The table contains 15 rows of data, including patient names, IDs, ages, and payment methods.

```
110 df.to_excel(caminho_excel_descritografado, index=False, engine='openpyxl')
111 print(f"Arquivo Excel com os dados descritografados criado em: {caminho_excel_descritografado}")
112
113 else:
114     print("Nenhum arquivo CSV foi criado.")
115
```

Run Importação\_Dados\_Amigo\_Cript\_Descript

```
/usr/bin/python3.10 /home/henrique/Documents/extração de dados amigo/Importação_Dados_Amigo_Cript_Descript.py
```

Data Atendimento	Código Atendimento	Paciente	Idade	ID amigo Profissional	Forma de Pagamento
02/10/2023	83495561	VFNORYN ONEBFN QBF FNAGBF	27	23648353 CRM: 12727	AMIL- Gruner
02/10/2023	83686386	QHYPVPYRQR ONEBFN PBFGN	53	23648347 CRM: 12727	AMIL- Gruner
03/10/2023	82936740	ZNEVN QNF TENCNF CNQVYUN QR OEVBG	71	62974898 CRM: 17510	GEAP - Coopecir
03/10/2023	82699298	FVZBAR ENSRNYN NAGHARF ERVF	41	60756755 CRM: 17510	Unimed Recife
03/10/2023	82699414	IVYZN EBQEVTHRF FNAGBF PBFGN	60	62903890 CRM: 17510	Unimed Recife
04/10/2023	83747953	NAQERNV ZNEVN CRERVEN QR FBHMN YBCRF	39	63958370 CRM: 17510	Unimed Recife
04/10/2023	83748154	EHGR TBAQNYIRF QR N. SREANAQRF	75	28759333 CRM: 17510	Unimed Recife
04/10/2023	83748239	SYNIVN XNEVAN FBNERF OENTN	45	61287785 CRM: 17510	Unimed Recife
04/10/2023	83748366	INVQRAVPR ZNEVN QBF FNAGBF NTHVNE	58	62856214 CRM: 17510	SuLAmérica - Gruner
04/10/2023	83748619	RQAN ZRFDHVG	77	59428709 CRM: 17510	GEAP - Coopecir
04/10/2023	83749221	WBFYZN PBFGN ANFPVZRAGB FNAGBF	58	60897203 CRM: 17510	Unimed Recife
04/10/2023	83748848	FVYINAN ZNEVN INERYN QR NENHNB	56	61708940 CRM: 17510	Saúde Caixa - Coopecir
04/10/2023	83748895	NAN INVÉEVN QR NMRIRQB FNZCNVB	46	7503125 CRM: 17510	Unimed Recife
06/10/2023	84276751	ZNEVN ERANGN PBEERVN QR NENHNB CRERVEN	67	64289625 CRM: 12727	CASSI - Gruner

extração de dados amigo > Importação\_Dados\_Amigo\_Cript\_Descript.py 113:6 (4361 chars, 114 line breaks) LF UTF-8 4 spaces Python 3.10



Project: extração de dados amigo

Importação\_Dados\_Amigo\_Cript\_Descript.py

```
110 df.to_excel(caminho_excel_descriptografado, index=False, engine='openpyxl')
111 print(f"Arquivo Excel com os dados descriptografados criado em: {caminho_excel_descriptografado}")
112
113 else:
114     print("Nenhum arquivo CSV foi criado.")
115
```

Run: Importação\_Dados\_Amigo\_Cript\_Descript

Gruner - Grupo de Neurocirurgia do Recife

Item	Quantidade	Valor total do item R\$	Número lote	Data envio	Data pagamento	Valor pago R\$	Valor glosado R\$	Mês pagamento	Recurso
Consulta   Convênio	1.0	93.00	NaN	NaN	NaN	0.0	0.0	NaN	Não
Consulta   Convênio	1.0	93.00	NaN	NaN	NaN	0.0	0.0	NaN	Não
Consulta   Convênio	1.0	100.00	NaN	NaN	NaN	0.0	0.0	NaN	Não
Consulta   Convênio	1.0	115.00	NaN	NaN	NaN	0.0	0.0	NaN	Não
Consulta   Convênio	1.0	115.00	NaN	NaN	NaN	0.0	0.0	NaN	Não
Consulta   Convênio	1.0	115.00	NaN	NaN	NaN	0.0	0.0	NaN	Não
Consulta   Convênio	1.0	115.00	NaN	NaN	NaN	0.0	0.0	NaN	Não
Consulta   Convênio	1.0	115.00	NaN	NaN	NaN	0.0	0.0	NaN	Não
Consulta   Convênio	1.0	173.46	NaN	NaN	NaN	0.0	0.0	NaN	Não
Consulta   Convênio	1.0	100.00	NaN	NaN	NaN	0.0	0.0	NaN	Não
Consulta   Convênio	1.0	115.00	NaN	NaN	NaN	0.0	0.0	NaN	Não
Consulta   Convênio	1.0	100.13	NaN	NaN	NaN	0.0	0.0	NaN	Não
Consulta   Convênio	1.0	115.00	NaN	NaN	NaN	0.0	0.0	NaN	Não
Consulta   Convênio	1.0	104.00	NaN	NaN	NaN	0.0	0.0	NaN	Não

Project: extração de dados amigo

Importação\_Dados\_Amigo\_Cript\_Descript.py

```
110 df.to_excel(caminho_excel_descriptografado, index=False, engine='openpyxl')
111 print(f"Arquivo Excel com os dados descriptografados criado em: {caminho_excel_descriptografado}")
112
113 else:
114     print("Nenhum arquivo CSV foi criado.")
115
```

Run: Importação\_Dados\_Amigo\_Cript\_Descript

Número lote	Data envio	Data pagamento	Valor pago R\$	Valor glosado R\$	Mês pagamento	Recurso	Forma de Cobrança
NaN	NaN	NaN	0.0	0.0	NaN	Não	Gesaúde
NaN	NaN	NaN	0.0	0.0	NaN	Não	Gesaúde
NaN	NaN	NaN	0.0	0.0	NaN	Não	Coopecir
NaN	NaN	NaN	0.0	0.0	NaN	Não	Unimed Recife
NaN	NaN	NaN	0.0	0.0	NaN	Não	Unimed Recife
NaN	NaN	NaN	0.0	0.0	NaN	Não	Unimed Recife
NaN	NaN	NaN	0.0	0.0	NaN	Não	Unimed Recife
NaN	NaN	NaN	0.0	0.0	NaN	Não	Gesaúde
NaN	NaN	NaN	0.0	0.0	NaN	Não	Coopecir
NaN	NaN	NaN	0.0	0.0	NaN	Não	Unimed Recife
NaN	NaN	NaN	0.0	0.0	NaN	Não	Coopecir
NaN	NaN	NaN	0.0	0.0	NaN	Não	Unimed Recife
NaN	NaN	NaN	0.0	0.0	NaN	Não	Gesaúde

## Resumo do Código

O código em questão realiza várias operações em um conjunto de dados contido em um arquivo Excel. Aqui estão algumas das principais funcionalidades do código:

1. **Leitura de Dados:** Utiliza a biblioteca `pandas` para ler um arquivo Excel especificado, selecionando colunas específicas.
2. **Manipulação de Dados:** Realiza diversas transformações nos dados, como mapeamento de valores, ajuste de formatos, descaracterização de texto usando ROT13, criação de novas colunas com base em regras definidas, entre outras.
3. **Visualização:** Configura opções de formatação para melhorar a visualização do `DataFrame` e exibe os dados formatados no console, incluindo um cabeçalho extra.
4. **Exportação de Dados:** Oferece ao usuário a opção de criar um arquivo CSV, permitindo também a escolha de descriptografar certos dados antes da exportação.

### Qualidades:

1. **Clareza e Estrutura:** O código é relativamente claro e está estruturado em seções, facilitando a compreensão.
2. **Uso de Bibliotecas:** Utiliza bibliotecas populares como `pandas` e `openpyxl` para manipulação eficiente de dados e leitura/gravação de arquivos Excel.
3. **Interação com o Usuário:** Incorpora interação do usuário para decidir sobre a criação de arquivos CSV e a opção de descriptografar dados.

### Defeitos:

1. **Comentários Ausentes:** Embora o código seja em grande parte autoexplicativo, alguns comentários adicionais poderiam ser úteis para esclarecer a lógica por trás de certas operações.
2. **Mágica Números:** Existem alguns valores "mágicos" (como o número 20 no `print(cabecalho_extra.center(df.shape[1] * 20))`) que poderiam ser substituídos por constantes ou variáveis mais descritivas.
3. **Repetição de Código:** Há repetição de lógica para descriptografar a coluna 'Paciente' no caso de exportar o Excel descriptografado. Poderia ser consolidada em uma função para evitar duplicação de código.
- 4.

### Sugestões de Melhoria a Longo Prazo:

1. **Documentação:** Adicionar comentários explicativos para melhorar a compreensão do código.
2. **Refatoração:** Identificar oportunidades para refatorar partes do código, especialmente onde há repetição, visando melhorar a legibilidade e manutenção.
3. **Testes Unitários:** Implementar testes unitários para garantir a robustez do código, especialmente ao realizar transformações nos dados.



4. **Tratamento de Exceções:** Adicionar tratamento de exceções para lidar com possíveis erros durante a leitura/gravação de arquivos e outras operações.
5. **Modularização:** Considerar a possibilidade de dividir o código em funções/modular para tornar cada parte mais fácil de entender e testar.

Lembrando que essas sugestões são considerações gerais e a aplicabilidade delas pode depender do contexto específico em que o código está sendo utilizado.

### Arquivos

O PROJETO TEVE INICIO AOS DIAS 27 DE DEZEMBRO DE 2023 E DEVE CONTINUAR ...

PRODUZIDO POR EDSON HENRIQUE

GABRIEL FEITOSA

AMBOS, ESTUDANTES DE ANÁLISE E DESENVOLVIMENTO DE SISTEMAS.

AQUI TAMBÉM REGISTRO MEUS AGRADECIMENTOS AO GRUPO GRUNER E A COORDENAÇÃO DO EXCELENTE TRABALHO MINISTRADO PELO Dr. JOSÉ BARBOSA.