TESTE – JUAN MATHEUS

[22]:
```python
import requests
import pandas as pd
import time
```

Primeiro importo a biblioteca requests, que é um biblioteca http para python

Importo a biblioteca pandas, que é para maninupalação

Importo a biblioteca time, que é para funções relacionadas a tempo

[23]:
```python
def get_data_for_page(url, page):
    # Parâmetros da solicitação GET
    params = {'page': page, 'pageSize': 50}

    # Fazendo a solicitação GET
    response = requests.get(url, params=params)

    # Verificando se a solicitação foi bem sucedida
    if response.status_code == 200:
        # Convertendo a resposta para JSON e retornando os itens
        data = response.json()
        return data.get('items', [])
    else:
        # Se não for bem sucedida, exibimos uma mensagem de erro
        print(f"Erro na página {page}: {response.status_code}")
        return None
```

def get_data_for_page(url, page): Essa função é projetada para buscar dados de uma página de URL específica.

params = {'page': page, 'pageSize': 50}: Aqui, são definidos os parâmetros da solicitação GET. A página que desejamos obter e o tamanho da página (neste caso, 50 itens por página) são especificados.

response = requests.get(url, params=params): Esta linha faz uma solicitação GET para a URL especificada, passando os parâmetros definidos anteriormente.

if response.status_code == 200: Aqui, verifico se a solicitação foi bem-sucedida. O status de resposta 200 indica uma resposta bem-sucedida.

data = response.json(): Se a solicitação foi bem-sucedida, é convertido e a resposta para JSON e a armazenada na variável data.

return data.get('items', []): São retornados os itens da resposta JSON. Se não houver itens, retornamos uma lista vazia [].

else

print(f"Erro na página {page}: Se a solicitação não foi bem-sucedida, aparece a mensagem de erro e o código de status da resposta.

return None: É retornado None para indicar que não foi possível obter os dados da página especificada.

```
# URL do endpoint
url = 'https://api.fbi.gov/wanted/v1/list'
```

Define uma variável “url” que armazena o endpoint da API. Esta url será usada na função ‘get_data_for_page’ para fazer solicitações GET para obter os dados.

```
# Lista para armazenar os dados de todas as páginas
data_list = []
```

Aqui foi criado uma lista vazia, será usada para armazenar os dados das páginas que serão buscadas.

```
# Número total de páginas
total_pages = 20
```

Esse código indica o total de número de páginas que serão buscadas.

```
# Tamanho do lote de páginas a serem processadas de uma vez
batch_size = 5
```

Tamanho do lote de páginas a serem buscadas se serem processadas por vez

```
# Tempo de espera entre os lotes de solicitações (em segundos)
wait_time = 40
```

Indica que o tempo de espera entre os lotes de solicitações é de 40s

```
# Fator de aumento do tempo de espera entre tentativas
wait_time_multiplier = 1.5
```

Indica um fator de aumento do tempo de espera entre uma tentativa e outra.

```
# Loop através das páginas em lotes
for batch_start in range(1, total_pages + 1, batch_size):
    # Obter páginas neste lote
    batch_pages = range(batch_start, min(batch_start + batch_size, total_pages + 1))

    # Loop através das páginas neste lote
    for page in batch_pages:
        # Obter dados para esta página
        page_data = get_data_for_page(url, page)

        # Se os dados foram recebidos com sucesso, adicione-os à lista de dados
        if page_data is not None:
            data_list.extend(page_data)

    # Aguardar um intervalo de tempo após o processamento de cada lote
    time.sleep(wait_time)
```

for batch_start in range(1, total_pages + 1, batch_size): Este loop está através do intervalo de 1 até o número total de páginas, com um passo igual ao tamanho do lote. Isso significa que ele avança de batch_size em batch_size, começando em 1.

batch_pages = range(batch_start, min(batch_start + batch_size, total_pages + 1)): Aqui, batch_pages é definido como um intervalo de páginas para este lote. O intervalo começa em batch_start e vai até o mínimo entre batch_start + batch_size e total_pages + 1. Isso garante que o último lote possa ser menor do que batch_size se o número total de páginas não for um múltiplo exato do tamanho do lote.

for page in batch_pages: Este é um loop interno que itera sobre as páginas neste lote.

page_data = get_data_for_page(url, page): Aqui, é chamado a função get_data_for_page para obter os dados da página atual.

if page_data is not None: Aqui é verificado se os dados da página foram recebidos com sucesso.

data_list.extend(page_data): Se os dados foram recebidos com sucesso, eles são adicionados à lista data_list.

time.sleep(wait_time): Após processar cada lote de páginas, o programa espera um intervalo de tempo definido por wait_time.

```
# Convertendo a lista de dicionários para um DataFrame do pandas
df = pd.DataFrame(data_list)
```

Aqui eu converto a lista para um DF do Pandas.

```
# Selecionando apenas as colunas desejadas
df = df[['title', 'sex', 'eyes_raw', 'nationality', 'hair', 'weight', 'race', 'dates_of_birth_used', 'subjects']]
```

Seleciono as colunas necessárias.

```
# Renomeando as colunas
df = df.rename(columns={'title': 'nome', 'sex': 'sexo', 'eyes_raw': 'cor_olhos',
                        'nationality': 'nacionalidade', 'hair': 'cor_cabelo',
                        'weight': 'peso', 'race': 'raca',
                        'dates_of_birth_used': 'data_nascimento', 'subjects': 'delitos'})
```

Utilizo “rename” para renomear as colunas.

```
# Exibindo o DataFrame resultante
print(df)
```

Exibo o Df na Tela.

```
print(df[df['nome'] == 'AARON PAUL VICTORY'])
```

```
                 nome  sexo cor_olhos nacionalidade cor_cabelo  \
8  AARON PAUL VICTORY  Male     Brown          None      brown

                peso   raca  \
8  196 to 201 pounds  white

                                    data_nascimento  \
8  [January 28, 1979, July 28, 1979, January 28, ...

                                       delitos
8  [Additional Violent Crimes, Case of the Week]
```

Aqui filtro o nome “AARON PAUL VICTORY” como exemplo, no df sem alterações.

```
# Salvando o DataFrame alterado em um arquivo Excel
df.to_excel('df.xlsx', index=False)
```

Utilizo a função “to_excel” para transformar o df em um arquivo excel.

```
# Separar os delitos em linhas individuais
new_data = []
for index, row in df.iterrows():
    for delito in row['delitos']:
        new_row = row.copy()
        new_row['delito'] = delito
        new_data.append(new_row)
```

new_data = []: Inicializa uma nova lista vazia que será usada para armazenar os dados processados.

for index, row in df.iterrows(): Este loop itera sobre cada linha do DataFrame df. index é o índice da linha e row é uma série que contém os dados da linha atual.

for delito in row['delitos']: Para cada linha, este loop itera sobre os delitos associados a essa linha. A coluna 'delitos' parece ser uma lista de delitos para cada suspeito.

new_row = row.copy(): Cria uma cópia da linha atual. Isso garante que as alterações feitas na nova linha não afetem a linha original.

new_row['delito'] = delito: Adiciona uma nova coluna chamada 'delito' à nova linha e atribui o valor do delito atual a essa coluna.

new_data.append(new_row): Adiciona a nova linha à lista new_data.

```
# Criar o DataFrame alterado
df_alterado = pd.DataFrame(new_data)
```

Crio o novo df usando a lista ‘new_data’

```
# Exibindo o DataFrame resultante
print(df_alterado)
```

Exibo o novo Df na Tela.

```
for index, row in df_alterado.iterrows():
    if isinstance(row['data_nascimento'], list):  # Verificar se é uma lista
        df_alterado.at[index, 'data_nascimento'] = ', '.join(row['data_nascimento'])  # Juntar os elementos da lista em uma str
    else:
        print(value)  # Se não for uma lista, imprimir o valor diretamente
```

for index, row in df_alterado.iterrows():

Este loop for itera sobre todas as linhas do DataFrame df_alterado. A cada iteração, index contém o índice da linha e row contém os dados daquela linha.

if isinstance(row['data_nascimento'], list):

Esta linha verifica se o valor na coluna 'data_nascimento' da linha atual (row) é uma lista. A função isinstance() é usada para fazer essa verificação. Se o valor for uma lista, o bloco de código dentro do if será executado.

df_alterado.at[index, 'data_nascimento'] = ', '.join(row['data_nascimento']) :

Se o valor em 'data_nascimento' for uma lista, esta linha junta os elementos dessa lista em uma única string, onde os elementos são separados por vírgula e espaço. O método join() é usado para essa concatenação, e o resultado é atribuído de volta à mesma posição no DataFrame df_alterado.

else:

    print(value)

Se o valor em 'data_nascimento' não for uma lista, este bloco será executado. Ele simplesmente imprime o valor diretamente.

```
# Remover a coluna 'delitos'
df_alterado.drop(columns=['delitos'], inplace=True)
```

Aqui eu removo a coluna ‘delitos’

```
# Exibir o DataFrame resultante
print(df_alterado)
```

Exibo o novo Df na Tela.

```
print(df_alterado[df_alterado['nome'] == 'AARON PAUL VICTORY'])
```

```
                nome  sexo cor_olhos nacionalidade cor_cabelo  \
8  AARON PAUL VICTORY  Male     Brown          None      brown
8  AARON PAUL VICTORY  Male     Brown          None      brown

               peso   raca  \
8  196 to 201 pounds  white
8  196 to 201 pounds  white

                                    data_nascimento  \
8  January 28, 1979, July 28, 1979, January 28, 1969
8  January 28, 1979, July 28, 1979, January 28, 1969

                     delito
8  Additional Violent Crimes
8           Case of the Week
```

Aqui filtro o nome “AARON PAUL VICTORY” como exemplo, no df com as alterações.

```
# Salvando o DataFrame alterado em um arquivo Excel
df_alterado.to_excel('df_alterado.xlsx', index=False)
```

Utilizo a função “to_excel” para transformar o novo df em um arquivo excel.