Programa de Automação Web com Python

Introdução
Este programa utiliza Python para automatizar o preenchimento de um formulário web com dados de uma planilha Excel. É um exemplo prático de RPA (*Robotic Process Automation*), que simula ações humanas em um navegador. Vamos entender cada parte do código em detalhes.

Passo 1: Importação de Bibliotecas

```
1   from selenium import webdriver
2   from selenium.webdriver.common.by import By
3   from selenium.webdriver.edge.service import Service
4   from selenium.webdriver.support.ui import WebDriverWait
5   from selenium.webdriver.support import expected_conditions as EC
6   import openpyxl
7   import time
```

- **O que faz?**
  - selenium: Biblioteca para controlar o navegador (ex: Edge, Chrome).
  - openpyxl: Lê e escreve arquivos do Excel.
  - time: Cria pausas no código (ex: esperar 1 segundo após um clique).

Passo 2: Configurações Iniciais

```
10  excel_path = r"D:\OneDrive\RPA\challenge.xlsx"
11  edge_driver_path = r"D:\OneDrive\RPA\edgedriver_win64\msedgedriver.exe"
12  start_button_xpath = '//button[text()="Start"]'  # XPath do botão Start
```

- **O que faz?**
  - Define os caminhos do arquivo Excel (excel_path) e do driver do Edge (edge_driver_path).
  - start_button_xpath: Localiza o botão "Start" no site usando XPath (linguagem para encontrar elementos em páginas web).

Passo 3: Inicializar o Navegador

```
15    service = Service(edge_driver_path)
16    driver = webdriver.Edge(service=service)
```

- **O que faz?**
  - Cria um serviço para o Edge usando o caminho do driver.
  - Inicia o navegador Edge (webdriver.Edge).

Passo 4: Abrir o Site e Clicar em "Start"

```
18  try:
19      driver.get("https://www.rpachallenge.com/")
20      wait = WebDriverWait(driver, 10)
21      start_button = wait.until(EC.presence_of_element_located((By.XPATH, start_button_xpath)))
22      start_button.click()
```

- **O que faz?**
  - driver.get(...): Abre o site do desafio.
  - WebDriverWait: Espera até 10 segundos para o botão "Start" aparecer.
  - start_button.click(): Clica no botão quando ele estiver visível.

Passo 5: Ler Dados do Excel

```
28      workbook = openpyxl.load_workbook(excel_path)
29      sheet = workbook.active
30      excel_data = []
31
32      for row in sheet.iter_rows(min_row=2, values_only=True):
33          excel_data.append({
34              "First Name": row[0],
35              "Last Name": row[1],
36              "Company Name": row[2],
37              "Role in Company": row[3],
38              "Address": row[4],
39              "Email": row[5],
40              "Phone Number": row[6]
41          })
```

- **O que faz?**
  - Carrega o arquivo Excel e seleciona a planilha ativa.
  - Lê cada linha a partir da segunda (ignora o cabeçalho) e armazena os dados em uma lista de dicionários.
  - Exemplo: excel_data guarda informações como nome, sobrenome, empresa, etc.

Passo 6: Preencher o Formulário

```
    for item in excel_data:
        # Preencher os campos do formulário
        driver.find_element(By.CSS_SELECTOR, 'input[ng-reflect-name="labelFirstName"]').send_keys(item["First Name"])
        driver.find_element(By.CSS_SELECTOR, 'input[ng-reflect-name="labelLastName"]').send_keys(item["Last Name"])
        driver.find_element(By.CSS_SELECTOR, 'input[ng-reflect-name="labelCompanyName"]').send_keys(item["Company Name"])
        driver.find_element(By.CSS_SELECTOR, 'input[ng-reflect-name="labelRole"]').send_keys(item["Role in Company"])
        driver.find_element(By.CSS_SELECTOR, 'input[ng-reflect-name="labelAddress"]').send_keys(item["Address"])
        driver.find_element(By.CSS_SELECTOR, 'input[ng-reflect-name="labelEmail"]').send_keys(item["Email"])
        driver.find_element(By.CSS_SELECTOR, 'input[ng-reflect-name="labelPhone"]').send_keys(item["Phone Number"])

        # Clicar no botão Submit
        driver.find_element(By.CSS_SELECTOR, 'input[value="Submit"]').click()

        # Esperar um pouco para o próximo envio
        time.sleep(1)
```

- **O que faz?**
    - Para cada item em excel_data, preenche os campos do formulário usando seletores CSS (ex: input[ng-reflect-name="labelFirstName"]).
    - send_keys(...): Insere texto nos campos (ex: nome, telefone).
    - click(): Envia o formulário clicando em "Submit".
    - time.sleep(1): Espera 1 segundo antes do próximo envio.

Passo 7: Mensagem de Sucesso e Finalização

```
    success_message = wait.until(EC.presence_of_element_located((By.CLASS_NAME, 'congratulations')))
    print("Desafio concluído com sucesso!")
    print(success_message.text)

    # Manter o navegador aberto após a execução
    print("O navegador permanecerá aberto. Você pode fechá-lo manualmente.")
    input("Pressione Enter para fechar o script...")  # Aguarda entrada do usuário para encerrar o script
```

- **O que faz?**
    - Aguarda a mensagem de sucesso com a classe congratulations.
    - Exibe a mensagem no console e mantém o navegador aberto até o usuário pressionar Enter.

Passo 8: Tratamento de Erros

```
except Exception as e:
    print(f"Ocorreu um erro: {e}")
    input("Pressione Enter para fechar o script...")  # Aguarda entrada do usuário em caso de erro

finally:
    # Removido o driver.quit() para manter o navegador aberto
    pass
```

- **O que faz?**
    - Captura erros durante a execução (ex: arquivo Excel não encontrado, elemento web não existente).
    - Exibe a mensagem de erro e aguarda ação do usuário.

Conceitos-Chave

1. **Automação Web**: Simula interações humanas em navegadores (ex: cliques, preenchimento de campos).
2. **Seletores (XPath/CSS)**: Localizam elementos na página (ex: botões, campos de texto).
3. **Wait Explícito**: Espera condições específicas antes de prosseguir (ex: elemento carregar).

Dicas para Personalização

- Altere `excel_path` e `edge_driver_path` para seus próprios arquivos.
- Atualize os seletores CSS se a estrutura do site mudar.

Por Que Isso é Útil?
Automatizar tarefas repetitivas como preenchimento de formulários aumenta a eficiência e reduz erros. Este exemplo é aplicável em processos de negócios, testes de software e coleta de dados.