Crie um programa em python não usando o selenium, que baixe a cada 30 minutos um relatório no site amplo.eship.com.br, nesse site o usuario será usuario: dashboard3, senha: 12341234, ao logar na página precisar baixar os dados disponível, para isso tem um botão “exportar” com o seguinte elemento: <button type="button" onclick="javascript: setValorParaFormulario('botaoAcao', 'Exportar');setValorParaFormulario('coletarValores', 1);adicionarDadoRequisicao('modoRequisicao', 2);setValorParaFormulario('coletarValores', 1);setValorParaFormulario('elementoTroca', 'corpo');setValorParaFormulario('rota', 'HOME');ExecAjax();" id="cFuncaoListarRemessas\_Exportar" name="Exportar" class="field\_botao\_item display-flex s-v-align-center retangulo btn solido waves-effect waves-light " style="cursor: pointer; "><span>Exportar</span></button>, salvar o arquivo em excel como dashexp.xlsx. importante é que sempre o novo arquivo substitua o anterior.

Certifique-se de ter instalado a biblioteca msedge-selenium-tools usando o comando pip install msedge-selenium-tools.

Lembre-se de substituir as configurações usuario, senha, url\_login e arquivo\_saida pelo seu próprio nome de usuário, senha, URL de login e nome de arquivo desejado.

O código acima usará o navegador Edge para fazer o download do relatório a cada 30 minutos. Certifique-se de ter configurado corretamente o Microsoft Edge WebDriver no seu ambiente.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Nesse trecho de código, temos um loop principal que realiza a extração de elementos contendo palavras-chave e verifica se há uma próxima página para continuar o processo. Vou explicar cada linha em detalhes:

1. while True:: Inicia um loop infinito.
2. try:: Inicia um bloco try-except para lidar com possíveis exceções.
3. elementos = navegador.find\_elements(By.XPATH, '//\*[@id="FormListarRemessas"]/ul/li[2]/div/a[3]/div'): Encontra os elementos na página que correspondem ao XPath especificado e os armazena na variável elementos. Esse XPath é utilizado para localizar os elementos que contêm a soma das palavras-chave.
4. time.sleep(4): Aguarda por 4 segundos, permitindo que a página seja carregada completamente antes de prosseguir.
5. for elemento in elementos:: Itera sobre cada elemento encontrado na página.
6. conteudo\_elemento = elemento.text: Obtém o texto do elemento atual e armazena na variável conteudo\_elemento.
7. for palavra in palavras\_chave:: Itera sobre cada palavra-chave definida na lista palavras\_chave.
8. contador\_palavras\_chave[palavra] += conteudo\_elemento.count(palavra): Contabiliza o número de ocorrências da palavra-chave no conteúdo do elemento atual e incrementa o contador correspondente na estrutura de dados contador\_palavras\_chave.
9. except NoSuchElementException:: Captura a exceção NoSuchElementException, que é lançada quando o XPath não corresponde a nenhum elemento na página.
10. break: Sai do loop principal caso ocorra a exceção NoSuchElementException.
11. proxima\_pagina = navegador.find\_elements(By.XPATH, '//\*[@id="FormListarRemessas"]/ul/li[8]'): Encontra os elementos na página que correspondem ao XPath especificado e os armazena na variável proxima\_pagina. Esse XPath é utilizado para localizar o botão/link que leva à próxima página.
12. if len(proxima\_pagina) == 0 or "disabled" in proxima\_pagina[0].get\_attribute("class"):: Verifica se não há elementos na lista proxima\_pagina ou se o atributo class do primeiro elemento contém a palavra "disabled", indicando que não há próxima página disponível.
13. break: Sai do loop principal caso não haja próxima página disponível.
14. proxima\_pagina[0].click(): Clica no primeiro elemento da lista proxima\_pagina, que representa o botão/link para a próxima página.

Essas linhas de código permitem percorrer múltiplas páginas, extrair os elementos relevantes e contar as palavras-chave em cada página até que não haja mais páginas disponíveis.

09/07/2023 00:46:28