**1. 目的**

本脚本的目的是从 [中国货币网](https://www.chinamoney.com.cn/english/bdInfo/) 中爬取“2023年国债”信息，并将获取到的债券数据保存到本地 CSV 文件中。通过 Selenium 与 BeautifulSoup 的结合，实现了数据的自动化抓取与处理。

**2. 依赖库**

此脚本依赖以下 Python 库：

* selenium：用于自动化浏览器操作（例如：模拟点击、输入、等待元素加载等）。
* pandas：用于将抓取的数据转换为 DataFrame 格式并保存为 CSV 文件。
* beautifulsoup4：用于解析网页源代码并提取需要的表格数据。

**3. 安装依赖**

在运行脚本之前，请确保你已经安装了以下依赖包：

pip install selenium pandas beautifulsoup4

此外，您还需要下载并配置 **Edge 浏览器的 WebDriver**。请根据您的浏览器版本下载对应版本的 [Microsoft Edge WebDriver](https://developer.microsoft.com/en-us/microsoft-edge/tools/webdriver/) 并确保 msedgedriver.exe 在脚本的可执行路径中。

**4. 代码功能概述**

**1) 初始化浏览器驱动**

* 使用 selenium 库创建了一个 **Microsoft Edge 浏览器** 的无头（headless）实例，通过配置 WebDriver 和浏览器选项来启动。
* msedgedriver.exe 必须与当前浏览器版本兼容，并确保在脚本运行时可用。

**2) 打开目标网页**

* 访问 **中国货币网** 上的债券信息页面（https://www.chinamoney.com.cn/english/bdInfo/）。
* 页面加载后，脚本自动选择债券类型和发行年份。此时，我们选择了“Treasury Bond”（国债）和“2023”年。

**3) 自动填充选择条件**

* 在页面的下拉框中，选择债券类型（Treasury Bond）和年份（2023），模拟用户输入。

**4) 点击搜索按钮**

* 模拟点击 **Search** 按钮，触发搜索操作。

**5) 数据抓取**

* 使用 BeautifulSoup 解析当前页面的 HTML，并查找包含债券信息的表格。
* 提取表格的列标题和数据行，存储在列表 all\_data 中。

**6) 处理分页**

* 检查是否有“下一页”（Next）按钮。
* 如果有下一页，点击“下一页”按钮并继续抓取数据，直到所有页面的内容被抓取完毕。
* 如果没有“下一页”按钮或“下一页”按钮不可点击，则爬取停止。

**7) 保存数据**

* 使用 pandas 将抓取的所有数据存储为一个 CSV 文件（bond\_data\_all.csv）。

**8) 关闭浏览器**

* 爬取完成后，关闭浏览器。

**5. 代码工作流程**

1. **初始化浏览器**：通过设置 Service 和 Options 对象，启动一个无头模式的 Edge 浏览器实例。
2. **打开网页并等待元素加载**：在网页加载完成后，使用 WebDriverWait 等待页面上特定元素（如 Bond\_Type\_select）的加载。
3. **模拟用户操作**：自动选择债券类型和年份，并模拟点击搜索按钮进行查询。
4. **数据抓取和解析**：使用 BeautifulSoup 解析页面源代码，提取表格中的数据行并保存。每次解析一个页面的数据后，检查是否有下一页，若有则点击并继续抓取。
5. **保存为 CSV 文件**：所有页面的数据都存储在 all\_data 列表中，最终将其转换为 pandas.DataFrame，并保存为 CSV 文件。

**6. 代码的关键部分**

* **WebDriver 初始化和配置**

driver\_path = './msedgedriver.exe' # 设置浏览器驱动路径

service = Service(driver\_path)

options = Options()

options.add\_argument("--headless") # 启用无头模式

driver = webdriver.Edge(service=service, options=options)

* **模拟用户输入和点击操作**

# 输入选择条件：选择 "Treasury Bond"

bond\_type\_select = driver.find\_element(By.ID, 'Bond\_Type\_select')

bond\_type\_select.send\_keys('Treasury Bond')

# 输入选择条件：选择 "2023" 年

issue\_year\_select = driver.find\_element(By.ID, 'Issue\_Year\_select')

issue\_year\_select.send\_keys('2023')

# 点击搜索按钮

search\_button = driver.find\_element(By.XPATH, "//a[text()='Search']")

search\_button.click()

* **数据抓取和解析**

soup = BeautifulSoup(driver.page\_source, 'html.parser')

table = soup.find('table', class\_='san-sheet-alternating')

rows = table.find\_all('tr')

# 提取数据

data = []

for row in rows[1:]:

cols = row.find\_all('td')

data\_row = [col.text.strip() for col in cols]

data.append(data\_row)

* **分页处理**

# 等待“Next”按钮加载并点击

next\_button = WebDriverWait(driver, 2).until(

EC.element\_to\_be\_clickable((By.XPATH, "//li[@class='page-btn page-next']/a"))

)

# 判断是否可以点击“Next”按钮，若可以则继续爬取下一页

if "disabled" in next\_button.get\_attribute("class"):

break

else:

next\_button.click()

time.sleep(2)

* **保存数据到 CSV**

df = pd.DataFrame(all\_data, columns=columns)

df.to\_csv('bond\_data\_all.csv', index=False)

**7. 常见问题和排查**

* **WebDriver 不兼容**：确保所使用的 msedgedriver.exe 版本与 Microsoft Edge 浏览器版本一致。
* **元素无法定位**：可能是页面加载慢，考虑增加等待时间（WebDriverWait 或 time.sleep）。
* **分页问题**：确保“下一页”按钮的 XPath 路径正确，若页面结构发生变化，更新 XPath。

**8. 总结**

本脚本提供了一个简单的框架，用于从网页中抓取债券市场数据，并将其保存为 CSV 文件。通过使用 Selenium 模拟用户操作和 BeautifulSoup 解析网页内容，可以高效地抓取并处理多页数据。