== 工具 ==

python3 (3.10+), seleniumwire, undetected\_chromedriver

== 網站 ==

www.partslink24.com

== 說明 ==

1. 從入口網站左下方點選 bmw 徽標進入車型點選畫面

2. 從車型頁面源碼中找到相對應的車型 API 接口，進行呼叫。目的是要模擬點選

X1 -> F48 -> SAV -> X1 18d -> THA > Engine

3. 每一個點擊都有相對應的接口，和 json 返回內容。通過這些 json 找到下一層的 API 並獲得 json 內容。

=== 第一题 ===

在 partslink 下，一個汽車品牌下有多個車型年款，由不同的參數組裝而成.

掃描獲得 BMW 下所有車型，並保留每一個接口對應的 json 內容

e.g.

https://www.partslink24.com/p5bmw/extern/groups/func-mdl?hg=04&lang=zh-TW&mdl=X1&modelType=F48&res1=SAV&res2=X1+18d&res3=THA&serviceName=bmw\_parts

=== 第二题 ===

汽車品牌數據是樹狀結構。以BMW為例，

BMW - 1'

- E81

- 3 Doors

- 116d

- ECE

- 11 Engine

- 11\_4021 Short Engine

- Part 1

- Part 2

- 11\_4023 Engine block

- 11\_4099 Engine block mounting parts

- E82

- E87

由於車型眾多，為了加速數據獲取，一般都會以分散式框架，平行獲取。

在第一題 我們知道n+1層數據的API 都可以在 第n層 的API裡獲得。

假設獲得的API URL 會在 30分鐘後過期，且有五隻爬蟲實例，每個爬蟲每次可以發送一個 URL 請求，

請問該如何規劃請求來達到最少重複請求(最少請求次數)

1. 建立一個任務隊列，其中包含需要請求的URL。這些URL可能分層獲取，每層都可以從前一層的API中獲得。
2. 將所有爬蟲實例分為五組。每組中的爬蟲實例負責處理一部分URL。
3. 每個爬蟲實例從任務隊列中獲取一個URL，並發送請求獲取數據。如果數據中包含下一層的URL，則將這些URL添加到任務隊列中，以便後續處理。
4. 爬蟲實例在處理URL時，應該檢查該URL是否已經被其他爬蟲實例處理過。可以使用Redis分佈式鎖、Shared Database(會增加程式控制的複雜度與其他故障風險)等方式進行確認，以避免重複請求。
5. 當爬蟲實例處理完當前的URL後，可以繼續從任務隊列中獲取下一個URL進行處理。
6. 設定定時任務，每30分鐘重新檢查任務隊列中的URL是否已過期，如果已過期則從隊列中移除。

這種策略可以確保不會重複請求相同的URL，並且最大程度地平行處理多個URL，從而提高數據獲取效率。同時，定時清理過期的URL也可以確保你的爬蟲不會請求過期的數據。

題目補充：

1、完成後，請放 python code 以及結果放在 github 上，如果沒有 github 賬號的話，zip 後交給 HR；

2、不是透過 selenium + browser 點選連結，而是在進入主網頁後 使用 seleniumwire 自帶的 get/post 來進行 API 呼叫。