BEE2041 Empirical Project Blog

In [59]:
```
pip install selenium --upgrade
```

```
Requirement already up-to-date: selenium in c:\users\socor\anaconda3\lib\site-pa
ckages (4.19.0)
Requirement already satisfied, skipping upgrade: typing_extensions>=4.9.0 in
c:\users\socor\anaconda3\lib\site-packages (from selenium) (4.10.0)
Collecting certifi>=2021.10.8
  Downloading certifi-2024.2.2-py3-none-any.whl (163 kB)
Requirement already satisfied, skipping upgrade: trio-websocket~=0.9 in c:\users
\socor\anaconda3\lib\site-packages (from selenium) (0.11.1)
Requirement already satisfied, skipping upgrade: trio~=0.17 in c:\users\socor\an
aconda3\lib\site-packages (from selenium) (0.25.0)
Collecting urllib3[socks]<3,>=1.26
  Downloading urllib3-2.2.1-py3-none-any.whl (121 kB)
Requirement already satisfied, skipping upgrade: wsproto>=0.14 in c:\users\socor
\anaconda3\lib\site-packages (from trio-websocket~=0.9->selenium) (1.2.0)
Requirement already satisfied, skipping upgrade: exceptiongroup; python_version
< "3.11" in c:\users\socor\anaconda3\lib\site-packages (from trio-websocket~=0.9
->selenium) (1.2.0)
Requirement already satisfied, skipping upgrade: outcome in c:\users\socor\anaco
nda3\lib\site-packages (from trio~=0.17->selenium) (1.3.0.post0)
Requirement already satisfied, skipping upgrade: attrs>=23.2.0 in c:\users\socor
\anaconda3\lib\site-packages (from trio~=0.17->selenium) (23.2.0)
Requirement already satisfied, skipping upgrade: sortedcontainers in c:\users\so
cor\anaconda3\lib\site-packages (from trio~=0.17->selenium) (2.2.2)
Requirement already satisfied, skipping upgrade: sniffio>=1.3.0 in c:\users\soco
r\anaconda3\lib\site-packages (from trio~=0.17->selenium) (1.3.1)
Requirement already satisfied, skipping upgrade: idna in c:\users\socor\anaconda
3\lib\site-packages (from trio~=0.17->selenium) (2.10)
Requirement already satisfied, skipping upgrade: cffi>=1.14; os_name == "nt" and
implementation_name != "pypy" in c:\users\socor\anaconda3\lib\site-packages (fro
m trio~=0.17->selenium) (1.14.0)
Requirement already satisfied, skipping upgrade: pysocks!=1.5.7,<2.0,>=1.5.6; ex
tra == "socks" in c:\users\socor\anaconda3\lib\site-packages (from urllib3[sock
s]<3,>=1.26->selenium) (1.7.1)
Requirement already satisfied, skipping upgrade: h11<1,>=0.9.0 in c:\users\socor
\anaconda3\lib\site-packages (from wsproto>=0.14->trio-websocket~=0.9->selenium)
(0.14.0)
Requirement already satisfied, skipping upgrade: pycparser in c:\users\socor\ana
conda3\lib\site-packages (from cffi>=1.14; os_name == "nt" and implementation_na
me != "pypy"->trio~=0.17->selenium) (2.20)
Installing collected packages: certifi, urllib3
  Attempting uninstall: certifi
    Found existing installation: certifi 2020.6.20
    Uninstalling certifi-2020.6.20:
      Successfully uninstalled certifi-2020.6.20
  Attempting uninstall: urllib3
    Found existing installation: urllib3 1.25.9
    Uninstalling urllib3-1.25.9:
      Successfully uninstalled urllib3-1.25.9
Successfully installed certifi-2024.2.2 urllib3-2.2.1
Note: you may need to restart the kernel to use updated packages.

ERROR: requests 2.24.0 has requirement urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1,
but you'll have urllib3 2.2.1 which is incompatible.
```

In [37]:

```python
import requests

url = 'https://news.ycombinator.com/'
response = requests.get(url)
html_content = response.text
html_headers = response.headers
html_headers
```

Out[37]:

```
<html lang="en" op="news"><head><meta content="origin" name="referrer"/><meta
content="width=device-width, initial-scale=1.0" name="viewport"/><link href
="news.css?i3TCm9mQaQeIHjM4t2Io" rel="stylesheet" type="text/css"/>
<link href="y18.svg" rel="icon"/>
<link href="rss" rel="alternate" title="RSS" type="application/rss+xml"/>
<title>Hacker News</title></head><body><center><table bgcolor="#f6f6ef" borde
r="0" cellpadding="0" cellspacing="0" id="hnmain" width="85%">
<tr><td bgcolor="#ff6600"><table border="0" cellpadding="0" cellspacing="0" s
tyle="padding:2px" width="100%"><tr><td style="width:18px;padding-right:4px">
<a href="https://news.ycombinator.com"><img height="18" src="y18.svg" style
="border:1px white solid; display:block" width="18"/></a></td>
<td style="line-height:12pt; height:10px;"><span class="pagetop"><b class="hn
name"><a href="news">Hacker News</a></b>
<a href="newest">new</a> | <a href="front">past</a> | <a href="newcomments">c
omments</a> | <a href="ask">ask</a> | <a href="show">show</a> | <a href="job
s">jobs</a> | <a href="submit" rel="nofollow">submit</a> </span></td><td styl
e="text-align:right;padding-right:4px;"><span class="pagetop">
<a href="login?goto=news">login</a>
</span></td>
```

In [20]:

```python
soup = BeautifulSoup(html_content, 'html.parser')
soup
```

Out[20]:

```
<html lang="en" op="news"><head><meta content="origin" name="referrer"/><meta
content="width=device-width, initial-scale=1.0" name="viewport"/><link href
="news.css?i3TCm9mQaQeIHjM4t2Io" rel="stylesheet" type="text/css"/>
<link href="y18.svg" rel="icon"/>
<link href="rss" rel="alternate" title="RSS" type="application/rss+xml"/>
<title>Hacker News</title></head><body><center><table bgcolor="#f6f6ef" borde
r="0" cellpadding="0" cellspacing="0" id="hnmain" width="85%">
<tr><td bgcolor="#ff6600"><table border="0" cellpadding="0" cellspacing="0" s
tyle="padding:2px" width="100%"><tr><td style="width:18px;padding-right:4px">
<a href="https://news.ycombinator.com"><img height="18" src="y18.svg" style
="border:1px white solid; display:block" width="18"/></a></td>
<td style="line-height:12pt; height:10px;"><span class="pagetop"><b class="hn
name"><a href="news">Hacker News</a></b>
<a href="newest">new</a> | <a href="front">past</a> | <a href="newcomments">c
omments</a> | <a href="ask">ask</a> | <a href="show">show</a> | <a href="job
s">jobs</a> | <a href="submit" rel="nofollow">submit</a> </span></td><td styl
e="text-align:right;padding-right:4px;"><span class="pagetop">
<a href="login?goto=news">login</a>
</span></td>
```

In [35]:

```
headline_rows = soup.find_all('span', class_='titleline')
headlines = []
for row in headline_rows:
    headline_link = row.find('a')
    if headline_link:
    # Extract the text (headline) and the 'href' attribute (URL)
        headline_text = headline_link.text
        headline_url = headline_link['href']

    # Append a tuple of the headline text and URL to the headlines list
        headlines.append((headline_text, headline_url))
headlines
```

```
Out[35]: [('Structuralism as a Philosophy of Mathematics',
           'https://www.infinitelymore.xyz/p/structuralism'),
          ('Did any processor implement an integer square root instruction?',
           'https://retrocomputing.stackexchange.com/questions/29787/did-any-processor-im
         plement-an-integer-square-root-instruction'),
          ('ElephantSQL Is Shutting Down',
           'https://www.elephantsql.com/blog/end-of-life-announcement.html'),
          ('Is the frequency domain a real place?',
           'https://lcamtuf.substack.com/p/is-the-frequency-domain-a-real-place'),
          ('WinBtrfs – an open-source btrfs driver for Windows',
           'https://github.com/maharmstone/btrfs'),
          ('Sophia: Scalable Stochastic 2nd-Order Optimizer for Language Model Pre-Traini
         ng',
           'https://arxiv.org/abs/2305.14342'),
          ('PM2: Production Process Manager with a Built-In Load Balancer',
           'https://github.com/Unitech/pm2'),
          ('Show HN: Online database diagram editor',
           'https://github.com/drawdb-io/drawdb'),
          ('Cache is King: A guide for Docker layer caching in GitHub Actions',
           'https://blacksmith.sh/blog/cache-is-king-a-guide-for-docker-layer-caching-in-
         github-actions'),
          ('Dot: use of local LLMs and RAG in particular to interact with documents',
           'https://github.com/alexpinel/Dot'),
          ('Faces.js, a JavaScript library for generating vector-based cartoon faces',
           'https://zengm.com/facesjs/'),
          ('Gakken Ex-System', 'https://en.wikipedia.org/wiki/Gakken_EX-System'),
          ('A memory model for Rust code in the kernel',
           'https://lwn.net/SubscriberLink/967049/0ffb9b9ed8940013/'),
          ('A canonical Hamiltonian formulation of the Navier–Stokes problem',
           'https://www.cambridge.org/core/journals/journal-of-fluid-mechanics/article/ca
         nonical-hamiltonian-formulation-of-the-navierstokes-problem/B3EB9389AE700867A6A3
         EA63A45E69C6'),
          ('Lago, Open-Source Stripe Alternative, banks $22M in funding',
           'https://techcrunch.com/2024/03/14/lago-a-paris-based-open-source-billing-plat
         form-banks-22m/'),
          ('Anti-crime humps in medieval Venice',
           'https://www.visitvenezia.eu/en/venetianity/discover-venice/the-venetian-antib
         andito-humps-or-pissotte-what-exactly-are-they'),
          ('A Theory of Composing Protocols (2023)',
           'https://programming-journal.org/2023/7/6/'),
          ('More Agents Is All You Need: LLMs performance scales with the number of agent
         s',
           'https://arxiv.org/abs/2402.05120'),
          ('Chisel: A fast TCP/UDP tunnel over HTTP',
           'https://github.com/jpillora/chisel'),
          ('The xz sshd backdoor rabbithole goes quite a bit deeper',
           'https://twitter.com/bl4sty/status/1776691497506623562'),
          ('Exposure therapy for arachnophobia can benefit unrelated fears, study finds',
           'https://www.psypost.org/exposure-therapy-for-arachnophobia-can-benefit-unrela
         ted-fears-study-finds/'),
          ('Show HN: Brutalist Hacker News – A HN reader inspired by brutalist web desig
         n',
           'https://brutalisthackernews.com'),
          ('ChrysaLisp GUI Demo [video]',
           'https://www.youtube.com/watch?v=ADvyZOxlBu4'),
          ('Zep AI (YC W24) is hiring a founding Go engineer',
           'https://jobs.gem.com/zep/am9icG9zdDre4RbzEeB4wYY7s9TjXwhp'),
          ('Tokens, n-grams, and bag-of-words models (2023)',
           'https://zilliz.com/learn/introduction-to-natural-language-processing-tokens-n
         grams-bag-of-words-models'),
          ('Home insurers are dropping customers based on aerial images',
           'https://www.wsj.com/real-estate/home-insurance-aerial-images-37a18b16'),
          ('System/360 – CHM Revolution',
           'https://www.computerhistory.org/revolution/mainframe-computers/7/164'),
          ('Language models are Super Mario: Absorbing abilities from homologous models',
           'https://arxiv.org/abs/2311.03099'),
          ('What I think about when I edit (2019)',
           'https://evaparish.com/blog/how-i-edit'),
          ('New sunflower family tree reveals multiple origins of flower symmetry',
           'https://phys.org/news/2024-04-sunflower-family-tree-reveals-multiple.html')]
```

In [ ]:

```python
# Find all 'a' tags within 'td' tags with the class 'title'
headline_tags = soup.find_all('a')

# Extract headlines and URLs
headlines = [(tag.text, tag['href']) for tag in headline_tags]
type(headline_rows)
```

In [9]:

```python
for i, (headline, url) in enumerate(headlines, 1):
    print(f"{i}. {headline}\n   {url}\n")
```

We need a list of all PCCs/force areas. let us scrape that list:

In [46]:

```python
url = 'https://www.police.uk/'
response = requests.get(url)
soup = BeautifulSoup(response.text, 'html.parser')
soup
```

Out[46]:

```
<!DOCTYPE html>
<html lang="en-US"><head><title>Just a moment...</title><meta content="text/h
tml; charset=utf-8" http-equiv="Content-Type"/><meta content="IE=Edge" http-e
quiv="X-UA-Compatible"/><meta content="noindex,nofollow" name="robots"/><meta
content="width=device-width,initial-scale=1" name="viewport"/><style>*{box-si
zing:border-box;margin:0;padding:0}html{line-height:1.15;-webkit-text-size-ad
just:100%;color:#313131}button,html{font-family:system-ui,-apple-system,Blink
MacSystemFont,Segoe UI,Roboto,Helvetica Neue,Arial,Noto Sans,sans-serif,Apple
Color Emoji,Segoe UI Emoji,Segoe UI Symbol,Noto Color Emoji}@media (prefers-c
olor-scheme:dark){body{background-color:#222;color:#d9d9d9}body a{color:#fff}
body a:hover{color:#ee730a;text-decoration:underline}body .lds-ring div{borde
r-color:#999 transparent transparent}body .font-red{color:#b20f03}body .big-b
utton,body .pow-button{background-color:#4693ff;color:#1d1d1d}body #challenge
-success-text{background-image:url(data:image/svg+xml;base64,PHN2ZyB4bWxucz0i
aHR0cDovL3d3dy53My5vcmcvMjAwMC9zdmciIHdpZHRoPSIzMiIgaGVpZ2h0PSIzMiIgZmlsbD0ib
m9uZSIgdmlld0JveD0iMCAwIDI2IDI2Ij48cGF0aCBmaWxsPSIjZDlkOWQ5IiBkPSJNMTMgMGExMy
AxMyAwIDEgMCAwIDI2IDEzIDEzIDAgMCAwIDAtMjZtMCAyNGExMSAxMSAwIDEgMSAwLTIyIDExIDE
xIDAgMCAxIDAgMjIiLz48cGF0aCBmaWxsPSIjZDlkOWQ5IiBkPSJtMTAuOTU1IDE2LjA1NS0zLjk1
LTQuMTI1LTEuNDQ1IDEuMzg1IDUuMzcgNS42MSA5LjQ5NS05LjYtMS40MzYtMi42bC04LjMzNS01
```

```
In [58]:  from selenium import webdriver
          from bs4 import BeautifulSoup

          driver = webdriver.Chrome(r'C:\Users\socor\Downloads\chromedriver-win64\chromedri
          driver.get('https://www.police.uk/')

          # Let the page load. Consider using WebDriverWait for better practice.
          import time
          time.sleep(5)  # Adjust sleep time as needed.

          soup = BeautifulSoup(driver.page_source, 'html.parser')
          print(soup)

          driver.quit()
```

```
  File "<ipython-input-58-c29cab7ae0dd>", line 1
    pip install selenium --upgrade
        ^
SyntaxError: invalid syntax
```

```
In [60]:  pip install cloudscraper
```

```
Collecting cloudscraper
  Downloading cloudscraper-1.2.71-py2.py3-none-any.whl (99 kB)
Collecting requests-toolbelt>=0.9.1
  Downloading requests_toolbelt-1.0.0-py2.py3-none-any.whl (54 kB)
Requirement already satisfied: requests>=2.9.2 in c:\users\socor\anaconda3\lib\s
ite-packages (from cloudscraper) (2.24.0)
Requirement already satisfied: pyparsing>=2.4.7 in c:\users\socor\anaconda3\lib
\site-packages (from cloudscraper) (2.4.7)
Collecting urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1
  Downloading urllib3-1.25.11-py2.py3-none-any.whl (127 kB)
Requirement already satisfied: chardet<4,>=3.0.2 in c:\users\socor\anaconda3\lib
\site-packages (from requests>=2.9.2->cloudscraper) (3.0.4)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\socor\anaconda3\li
b\site-packages (from requests>=2.9.2->cloudscraper) (2024.2.2)
Requirement already satisfied: idna<3,>=2.5 in c:\users\socor\anaconda3\lib\site
-packages (from requests>=2.9.2->cloudscraper) (2.10)
Installing collected packages: requests-toolbelt, cloudscraper, urllib3
  Attempting uninstall: urllib3
    Found existing installation: urllib3 2.2.1
    Uninstalling urllib3-2.2.1:
      Successfully uninstalled urllib3-2.2.1
Successfully installed cloudscraper-1.2.71 requests-toolbelt-1.0.0 urllib3-1.25.
11
Note: you may need to restart the kernel to use updated packages.

ERROR: selenium 4.19.0 has requirement urllib3[socks]<3,>=1.26, but you'll have
urllib3 1.25.11 which is incompatible.
```

In [62]:
```python
import cloudscraper
url = 'https://www.police.uk/'
scraper = cloudscraper.create_scraper()
res= scraper.get(url)
print(res.status_code)
res.text
```

```
403
```

Out[62]:
```
'<!DOCTYPE html><html lang="en-US"><head><title>Just a moment...</title><meta
http-equiv="Content-Type" content="text/html; charset=UTF-8"><meta http-equiv
="X-UA-Compatible" content="IE=Edge"><meta name="robots" content="noindex,nof
ollow"><meta name="viewport" content="width=device-width,initial-scale=1"><st
yle>*{box-sizing:border-box;margin:0;padding:0}html{line-height:1.15;-webkit-
text-size-adjust:100%;color:#313131}button,html{font-family:system-ui,-apple-
system,BlinkMacSystemFont,Segoe UI,Roboto,Helvetica Neue,Arial,Noto Sans,sans
-serif,Apple Color Emoji,Segoe UI Emoji,Segoe UI Symbol,Noto Color Emoji}@med
ia (prefers-color-scheme:dark){body{background-color:#222;color:#d9d9d9}body
a{color:#fff}body a:hover{color:#ee730a;text-decoration:underline}body .lds-r
ing div{border-color:#999 transparent transparent}body .font-red{color:#b20f0
3}body .big-button,body .pow-button{background-color:#4693ff;color:#1d1d1d}bo
dy #challenge-success-text{background-image:url(data:image/svg+xml;base64,PHN
2ZyB4bWxucz0iaHR0cDovL3d3dy53My5vcmcvMjAwMC9zdmciIHdpZHRoPSIzMiIgaGVpZ2h0PSIz
MiIgZmlsbD0ibm9uZSIgdmlld0JveD0iMCAwIDI2IDI2Ij48cGF0aCBmaWxsPSIjZDlkOWQ5IiBkP
SJNMTMgMGExMyAxMyAwIDEgMCAwIDI2IDEzIDEzIDAgMCAwIDAtMjZtMCAyNGExMSAxMSAwIDEgMS
AwLTIyIDExIDExIDAgMCAxIDAgMjIiLz48cGF0aCBmaWxsPSIjZDlkOWQ5IiBkPSJtMTAuOTU1IDE
```

In [66]:
```python
import cfscrape
url = 'https://www.police.uk/'
scrape = cfscrape.create_scraper()
res = scrape.get(url)
print(res.status_code)
```

```
403
```

In [64]:
```
pip install cfscrape
```

```
Collecting cfscrape
  Downloading cfscrape-2.1.1-py3-none-any.whl (12 kB)
Requirement already satisfied: requests>=2.6.1 in c:\users\socor\anaconda3\lib\s
ite-packages (from cfscrape) (2.24.0)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\socor\anaconda3\li
b\site-packages (from requests>=2.6.1->cfscrape) (2024.2.2)
Requirement already satisfied: idna<3,>=2.5 in c:\users\socor\anaconda3\lib\site
-packages (from requests>=2.6.1->cfscrape) (2.10)
Requirement already satisfied: chardet<4,>=3.0.2 in c:\users\socor\anaconda3\lib
\site-packages (from requests>=2.6.1->cfscrape) (3.0.4)
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in c:\use
rs\socor\anaconda3\lib\site-packages (from requests>=2.6.1->cfscrape) (1.25.11)
Installing collected packages: cfscrape
Successfully installed cfscrape-2.1.1
Note: you may need to restart the kernel to use updated packages.
```

```
In [1]:  from selenium import webdriver
         from selenium.webdriver.chrome.service import Service
         from selenium.webdriver.common.keys import Keys
         import time

         # Specify the path to chromedriver if it's not in your PATH
         chromedriver_path = r"C:\Users\socor\Downloads\chromedriver-win64\chromedriver-wi

         # Initialize the WebDriver (assuming Chrome)
         service = Service(executable_path=chromedriver_path)
         driver = webdriver.Chrome(service=service)

         try:
             # Navigate to a website
             driver.get("http://police.uk")

             # Wait for 5 seconds to see the page
             time.sleep(5)

             # Optionally, interact with the website
             # For example, search for 'Selenium' in Wikipedia
             # search_box = driver.find_element_by_name('q')
             # search_box.send_keys('Selenium')
             # search_box.send_keys(Keys.RETURN)
             # time.sleep(5)

             print("Selenium is working fine!")
         except Exception as e:
             print(f"An error occurred: {e}")
         finally:
             # Close the browser
             driver.quit()
```

```
Selenium is working fine!
```

```
In [ ]:
```

Having established that Selenium is capable of accessing the police.uk website, let's start building an ethical bot! Firstly, we accessed the https://police.uk/robots.txt (https://police.uk/robots.txt) page and found certain URLs needed to be disallowed. I decided to start by caching the robots.txt file so that my bot could refer to it without sending repeated requests to the site. My bot would then check URLs against those contained in the robot.txt file and would return a "robot.txt error" rather than crawl the forbidden URL:

In [19]:
```python
from urllib.robotparser import RobotFileParser
from urllib.parse import urlparse

def can_crawl(url):
    """
    Check if the crawler can crawl a given URL based on the site's robots.txt.
    """
    parsed_url = urlparse(url)
    robots_url = f"{parsed_url.scheme}://{parsed_url.netloc}/robots.txt"

    rp = RobotFileParser()
    rp.set_url(robots_url)
    rp.read()

    return rp.can_fetch("FriendlyUniStudentResearcher", url)

def crawl(url):
    """
    Attempt to crawl a URL, respecting robots.txt rules.
    """
    if can_crawl(url):
        try:
            response = requests.get(url)
            # Process the response here (e.g., parse HTML, follow links, etc.)
            print(f"Successfully crawled: {url}")
        except Exception as e:
            print(f"An error occurred while crawling {url}: {e}")
    else:
        print(f"robots.txt error: Crawling not allowed for {url}")

# Example usage
urls_to_crawl = [
    "http://police.uk/mediacentre",
    "http://police.uk/?u=media",
    "http://police.uk"
    # Add other URLs you're interested in
]

for url in urls_to_crawl:
    crawl(url)
```

```
robots.txt error: Crawling not allowed for http://police.uk/mediacentre (http://
police.uk/mediacentre)
robots.txt error: Crawling not allowed for http://police.uk/?u=media (http://pol
ice.uk/?u=media)
robots.txt error: Crawling not allowed for http://police.uk (http://police.uk)
```

It is customary to include a specific "user-agent" to identify your bot and make it possible for website administrators to contact you with concerns:

In [ ]:
```python
session = requests.Session()

# Set the custom user-agent for all requests made with this session
session.headers.update({
    'User-Agent': "FriendlyUniStudentResearcher/1.0 (+mailto:soc204@exeter.ac.uk)
})
response = session.get()
```

In [7]:

```python
import requests
import json

# Define your custom user-agent string
user_agent = "FriendlyUniStudentResearcher/1.0 (+mailto:soc204@exeter.ac.uk)"

# Set the headers for your request to include your custom user-agent
headers = {
    'User-Agent': user_agent
}

# The URL for testing headers (httpbin.org is useful for HTTP requests testing)
test_url = "https://httpbin.org/headers"

# Make the request with your headers
response = requests.get(test_url, headers=headers)

# Parse the JSON response
response_json = response.json()

# Extract and print the User-Agent header from the response
print("User-Agent received by httpbin.org:")
print(response_json['headers']['User-Agent'])
```

```
User-Agent received by httpbin.org:
FriendlyUniStudentResearcher/1.0 (+mailto:soc204@exeter.ac.uk)
```

In [ ]:

```python
from selenium import webdriver
from selenium.webdriver.chrome.service import Service
from selenium.webdriver.common.keys import Keys
import time
from selenium.webdriver.chrome.options import Options

options = Options()
user_agent = "FriendlyUniStudentResearcher/1.0 (+mailto:soc204@exeter.ac.uk)"
options.add_argument(f'user-agent={user_agent}')
# Specify the path to chromedriver if it's not in your PATH
chromedriver_path = r"C:\Users\socor\Downloads\chromedriver-win64\chromedriver-wi

# Initialize the WebDriver (assuming Chrome)
service = Service(executable_path=chromedriver_path)
driver = webdriver.Chrome(service=service)

session = requests.Session()

# Set the custom user-agent for all requests made with this session
session.headers.update({
    'User-Agent': "FriendlyUniStudentResearcher/1.0 (+mailto:soc204@exeter.ac.uk)
})
response = session.get()

driver.quit()
```

In [22]:

```python
url = 'https://www.police.uk/'
parsed_url = urlparse(url)
robots_url = f"{parsed_url.scheme}://{parsed_url.netloc}/robots.txt"
rp = RobotFileParser()
rp.set_url(robots_url)
rp.read()
for line in rp.default_entry.rulelines:
    print(f"Allow: {line.allowance} Path: {line.path}")

# Check if the root URL is allowed
print(rp.can_fetch("*", "https://www.police.uk/"))
rp.can_fetch("*", url)
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-22-dd4376a6c90c> in <module>
      5 rp.set_url(robots_url)
      6 rp.read()
----> 7 for line in rp.default_entry.rulelines:
      8     print(f"Allow: {line.allowance} Path: {line.path}")
      9 

AttributeError: 'NoneType' object has no attribute 'rulelines'
```

In [18]:

```python
from urllib.parse import urlparse
from urllib.robotparser import RobotFileParser

# Initialize a cache dictionary
robots_cache = {}

def cache_robots_data(url):
    """
    Fetches and caches the robots.txt data for the given URL's domain.
    """
    # Parse the domain from the given URL
    parsed_url = urlparse(url)
    base_url = f"{parsed_url.scheme}://{parsed_url.netloc}"
    robots_url = f"{base_url}/robots.txt"

    # Check if we already have cached data for this domain
    if base_url in robots_cache:
        print("Using cached robots.txt data.")
        return robots_cache[base_url]
    else:
        print("Fetching new robots.txt data.")
        # Initialize a RobotFileParser instance
        rp = RobotFileParser()
        rp.set_url(robots_url)
        rp.read()  # Fetch and parse the robots.txt

        # Cache the RobotFileParser instance for future use
        robots_cache[base_url] = rp

        return rp

# Example usage
rp = cache_robots_data('https://www.police.uk')
rp
```

```
Fetching new robots.txt data.
```

Out[18]:

```
<urllib.robotparser.RobotFileParser at 0x22aa7399bb0>
```

In [ ]:

```python


from selenium.webdriver.common.keys import Keys








from selenium.webdriver.common.action_chains import ActionChains


```

In [1]:

```python
from selenium.webdriver.chrome.options import Options
def establish_user_agent(user_agent, chromedriver_path):
    chrome_options = Options()
    chrome_options.add_argument(f"user-agent={user_agent}")
    return chrome_options
```

In [2]:
```python
from selenium import webdriver
from selenium.webdriver.chrome.service import Service
def init_chrome_webdriver(chromedriver_path, chrome_options):
    service = Service(executable_path=chromedriver_path)
    driver = webdriver.Chrome(service=service, options=chrome_options)
    return driver
```

In [3]:
```python
import time
import json
from selenium.webdriver.common.by import By
def test_user_agent(driver, user_agent):
    driver.get("https://httpbin.org/user-agent")
    time.sleep(5)
    response_data = json.loads(driver.find_element(By.TAG_NAME, "body").text)
    echoed_user_agent = response_data["user-agent"]

    if echoed_user_agent != user_agent:
        print("User-Agent does not match the expected value. Quitting...")
        raise Exception("User-Agent does not match the expected value.")
```

In [7]:
```python
def is_target_disallowed(target, disallowed_url_patterns):
    """
    Check if the target path matches any of the disallowed paths.

    :param target_path: The target path to check
    :param disallowed_paths: A list of disallowed paths from robots.txt
    :return: True if the target path is disallowed, False otherwise
    """
    # Normalize target path
    target_pattern = f'{urlparse(target).path}?{urlparse(target).query}'
    target_path = target_pattern.rstrip("/")

    for disallowed in disallowed_url_patterns:
        # Normalize disallowed path
        disallowed = disallowed.rstrip("/")

        # Check if the target pattern starts with the disallowed pattern
        if target_path.startswith(disallowed):
            return True
        # Checking for file extension disallowance, e.g., '*.aspx$'
        if disallowed.endswith('$'):
            target_pattern = target_pattern.rstrip("?")
            target_path = target_pattern.rstrip("/")
            base_pattern = disallowed[1:-1]
            if target_path.endswith(base_pattern):
                return True

    return False
```

In [9]:
```python
from urllib.parse import urlparse
import re
def establish_bot_permissions(driver, target):
    parsed_url = urlparse(target)
    base_url = f"{parsed_url.scheme}://{parsed_url.netloc}"
    robots_url = f"{base_url}/robots.txt"
    driver.get(robots_url)
    time.sleep(1)
    robots_txt_content = driver.find_element(By.TAG_NAME, "body").text
    disallow_pattern = r"Disallow: ([^\n]+)"
    disallowed_paths = re.findall(disallow_pattern, robots_txt_content)

    if is_target_disallowed(target, disallowed_paths):
        print('This URL is not allowed to be crawled in line with robots.txt')
        raise Exception("Target path is disallowed.")
```

In [15]:
```python
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC

def get_force_areas(driver, target):
    driver.get(target)
    all_buttons = WebDriverWait(driver, 10).until(
        EC.presence_of_all_elements_located((By.CSS_SELECTOR, ".js-crime-stats-ta
    )

    if len(all_buttons) > 1:
        toggle_button = all_buttons[1]  # Select the second button
        driver.execute_script("arguments[0].scrollIntoView(true);", toggle_button
        toggle_button.click()
        time.sleep(2)
    else:
        print("Not enough buttons found.")

    tables = driver.find_elements(By.TAG_NAME, 'table')
    table = tables[-1]
    driver.execute_script("arguments[0].scrollIntoView(true);", table)
    rows = table.find_elements(By.TAG_NAME, 'tr')
    force_areas = []

    for row in rows:
        cells = row.find_elements(By.TAG_NAME, 'td')
        if cells:
            text = cells[0].text.strip()
            force_areas.append(text)

    return force_areas
```

```
In [16]:
# Setup User-Agent
user_agent = "FriendlyUniStudentResearcher/1.0 (+mailto:soc204@exeter.ac.uk)"
chromedriver_path = r"C:\Users\socor\Downloads\chromedriver-win64\chromedriver-wi

chrome_options = establish_user_agent(user_agent, chromedriver_path)

# Initialize the WebDriver (assuming Chrome)
driver = init_chrome_webdriver(chromedriver_path,chrome_options)

# Set target URL
target = 'https://www.police.uk/pu/your-area/south-wales-police/performance/finan

try:
    # Navigate to a website that echoes back the user-agent
    test_user_agent(driver, user_agent)

    # Navigate to target website robots.txt
    establish_bot_permissions(driver, target)

    # Collect the names of Force areas for which data is available
    Force_Areas = get_force_areas(driver, target)

    # This will contain the text from the first column of each row in your table
    print(Force_Areas)
    time.sleep(10)

    print("Selenium is working fine with the expected user-argument, and in line
except Exception as e:
    print(f"An error occurred: {e}")
finally:
    # Close the browser
    driver.quit()
```

```
['Avon and Somerset Constabulary', 'Bedfordshire Police', 'Cambridgeshire Consta
bulary', 'Cheshire Constabulary', 'Cleveland Police', 'Cumbria Constabulary', 'D
erbyshire Constabulary', 'Devon & Cornwall Police', 'Dorset Police', 'Durham Con
stabulary', 'Dyfed-Powys Police', 'Essex Police', 'Gloucestershire Constabular
y', 'Greater Manchester Police', 'Gwent Police', 'Hampshire Constabulary', 'Hert
fordshire Constabulary', 'Humberside Police', 'Kent Police', 'Lancashire Constab
ulary', 'Leicestershire Police', 'Lincolnshire Police', 'Merseyside Police', 'MO
PAC', 'Norfolk Constabulary', 'North Wales Police', 'North Yorkshire Police', 'N
orthamptonshire Police', 'Northumbria Police', 'Nottinghamshire Police', 'South
Wales Police', 'South Yorkshire Police', 'Staffordshire Police', 'Suffolk Consta
bulary', 'Surrey Police', 'Sussex Police', 'Thames Valley Police', 'Warwickshire
Police', 'West Mercia Police', 'West Midlands Police', 'West Yorkshire Police',
'Wiltshire Police', 'Total England & Wales']
Selenium is working fine with the expected user-argument, and in line with robot
s.txt!
```

In [44]:
```
import re
robots_txt_content[0:100]
# Regular expression to match 'Disallow' lines
disallow_pattern = r"Disallow: ([^\n]+)"

# Find all matches of the pattern
disallowed_url_patterns = re.findall(disallow_pattern, robots_txt_content)

# Print the list of disallowed paths
print(disallowed_url_patterns)
```

```
['/mediacentre', '/?u=media', '/DownloadEvent?', '/GetPdf/?', '/ExportPdf/?', '/
Complete?', '/GetPaginatedResults/?', '*.aspx$']
```

In [99]:
```
is_target_disallowed('http://police.uk/example',disallowed_paths)
#urlparse('https://police.uk/example/?u=media').query
```

```
/example?
/example?
.aspx
```

Out[99]:
```
False
```

In [117]:
```
len(Force_Areas)
```

Out[117]:
```
43
```

In [123]:

```python
from selenium import webdriver
from selenium.webdriver.chrome.service import Service
from selenium.webdriver.common.keys import Keys
import time
from selenium.webdriver.chrome.options import Options
from urllib.robotparser import RobotFileParser
from urllib.parse import urlparse
import json
from selenium.webdriver.common.by import By
import re

user_agent = "FriendlyUniStudentResearcher/1.0 (+mailto:soc204@exeter.ac.uk)"

# Specify the path to chromedriver if it's not in your PATH
chromedriver_path = r"C:\Users\socor\Downloads\chromedriver-win64\chromedriver-wi

chrome_options = Options()
chrome_options.add_argument(f"user-agent={user_agent}")

# Initialize the WebDriver (assuming Chrome)
service = Service(executable_path=chromedriver_path)
driver = webdriver.Chrome(service=service, options=chrome_options)
target = 'https://www.police.uk/pu/your-area/south-wales-police/performance/finan

try:
    # Navigate to a website that echoes back the user-agent
    driver.get("https://httpbin.org/user-agent")
    time.sleep(5)
    # Extract and check the user-agent from the page's response
    response_data = json.loads(driver.find_element(By.TAG_NAME, "body").text)
    echoed_user_agent = response_data["user-agent"]

    if echoed_user_agent != user_agent:
        print("User-Agent does not match the expected value. Quitting...")
        raise Exception("User-Agent does not match the expected value.")

    # Navigate to a website
    parsed_url = urlparse(target)
    base_url = f"{parsed_url.scheme}://{parsed_url.netloc}"
    robots_url = f"{base_url}/robots.txt"
    driver.get(robots_url)
    # Wait for 5 seconds to see the page
    time.sleep(1)
    # Extract text content from the <body> tag
    robots_txt_content = driver.find_element(by=By.TAG_NAME, value="body").text

    # Regular expression to match 'Disallow' lines
    disallow_pattern = r"Disallow: ([^\n]+)"

    # Find all matches of the pattern
    disallowed_paths = re.findall(disallow_pattern, robots_txt_content)
    if is_target_disallowed(target, disallowed_paths):
        print('This URL is not allowed to be crawled in line with robots.txt')
        raise Exception("Target path is disallowed.")
    time.sleep(2)
    driver.get(target)
    time.sleep(5)
    # Assuming you want to target only the last table's cells
    last_table_cells = driver.find_elements(By.CSS_SELECTOR, 'table:last-of-type

    # Extract text from each cell
    Force_Areas = [cell.text for cell in last_table_cells if cell.text.strip()]
    # Print extracted data
    print(Force_Areas)
    time.sleep(10)

    print("Selenium is working fine with the expected user-argument, and in line
except Exception as e:
    print(f"An error occurred: {e}")
finally:
    # Close the browser
    driver.quit()
```

```
[]
Selenium is working fine with the expected user-argument, and in line with robot
s.txt!
```

https://github.com/SOCStudentUoE/BEE2041-Empirical-Assignment
(https://github.com/SOCStudentUoE/BEE2041-Empirical-Assignment)