Browser Opening Algorithm

This code creates a script that uses Selenium and the Chrome webdriver to open a Google Chrome tab. It also uses the 'fake_useragent' library to randomly generate a user agent, which is then passed as an option to the Chrome webdriver. The script also includes several options passed to the webdriver to enhance privacy such as disabling dev shm usage, disabling blink features, incognito mode and disabling the infobars. Additionally, the script attempts to remove the webdriver property from the navigator object to prevent detection of the automation. The script calls the main() function which creates an instance of the WebDriver class and uses its get() method to open the website. Before running this script, you will need to have the Chrome webdriver installed on your machine. You can download the appropriate version of the Chrome webdriver for your system from the Selenium website (http://chromedriver.chromium.org/home). Once you have downloaded the webdriver, you will need to add it to your system's PATH. On Linux and macOS systems, the recommended location to put the webdriver is in /usr/local/bin. Once you have placed the webdriver in this location, you should be able to run the script without any issues. Keep in mind that the version of the Chrome browser and the Chrome webdriver must match in order for the script to work.

Packages:

In [ ]:

```
!pip install fake_useragent
!pip install selenium
!pip install bs4
```

Code:

In [10]:

```
import sys
import os

from fake_useragent import UserAgent
from selenium import webdriver
from selenium.webdriver.chrome.options import Options
from selenium.webdriver.common.action_chains import ActionChains
from selenium.webdriver.support import expected_conditions as EC
from selenium.webdriver.support.ui import WebDriverWait
from selenium.common.exceptions import TimeoutException

print('All modules are loaded')


class Spoofer(object):
    def __init__(self):
        self.userAgent = UserAgent().random

    def get(self):
        return self.userAgent

class DriverOptions(object):
    def __init__(self):
        self.options = Options()
        self.options.add_argument('--no-sandbox')
        self.options.add_argument('--start-maximized')
        self.options.add_argument('--start-fullscreen')
        self.options.add_argument('--single-process')
        self.options.add_argument('--disable-dev-shm-usage')
        self.options.add_argument("--incognito")
        self.options.add_argument('--disable-blink-features=AutomationControlled')
        self.options.add_argument('--disable-blink-features=AutomationControlled')
        self.options.add_experimental_option('useAutomationExtension', False)
        self.options.add_experimental_option("excludeSwitches", ["enable-automation"])
        self.options.add_argument("disable-infobars")

        self.helperSpoofer = Spoofer()
        self.options.add_argument('user-agent={}'.format(self.helperSpoofer.userAgent))

class WebDriver(DriverOptions):
    def __init__(self):
        DriverOptions.__init__(self)
        self.driver_instance = self.get_driver()

    def get_driver(self):
        print("""
        UserAgent: {}
        """.format(self.helperSpoofer.userAgent))

        driver = webdriver.Chrome(options=self.options)

        driver.execute_script("Object.defineProperty(navigator, 'webdriver', {get: () => undefined})")
        driver.execute_cdp_cmd("Page.addScriptToEvaluateOnNewDocument", {
```

```
            "source":
                "const newProto = navigator.__proto__;"
                "delete newProto.webdriver;"
                "navigator.__proto__ = newProto;"
        })

        return driver

def main():
    driver = WebDriver()
    driver.driver_instance.get('https://www.whatismybrowser.com/')
    # Do your stuff here


if __name__ == "__main__":
    main()
```

```
All modules are loaded

        UserAgent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.9.0.16) Gecko/2009120208 Firefox/3.0.16 F
BSMTWB
```

created by: Santiago Amoretti