Blacklane Case Study Documentation

Overview

This document provides an overview and documentation for the web scraping script developed using Selenium to retrieve information from the Blacklane website.

Requirements

- Python
- Selenium
- Pandas
- ChromeDriver

Setups

- Install required libraries: '*pip install selenium pandas*'
- Download ChromeDriver and extract: *https://sites.google.com/chromium.org/driver/*

Script Overview

This script automates the process of retrieving price data from the car transfers from the Blacklane website using Selenium. The following steps are as follows:

- Launch the Chrome browser using Selenium.
- Accept website cookies.
- Input the pickup and drop-off locations.
- Click the search button to display the different services offered by Blacklane.

Script Components

- **ChromeDriver Initial Setup**
  The *'webdriver.chrome'* instance is set up with a path to the ChromeDriver application located on the local machine.

```
PATH = 'C:\\Program Files (x86)\\chromedriver.exe'

service = Service(executable_path = PATH)

driver = webdriver.Chrome(service = service)

driver.get('https://www.blacklane.com/en/')
```

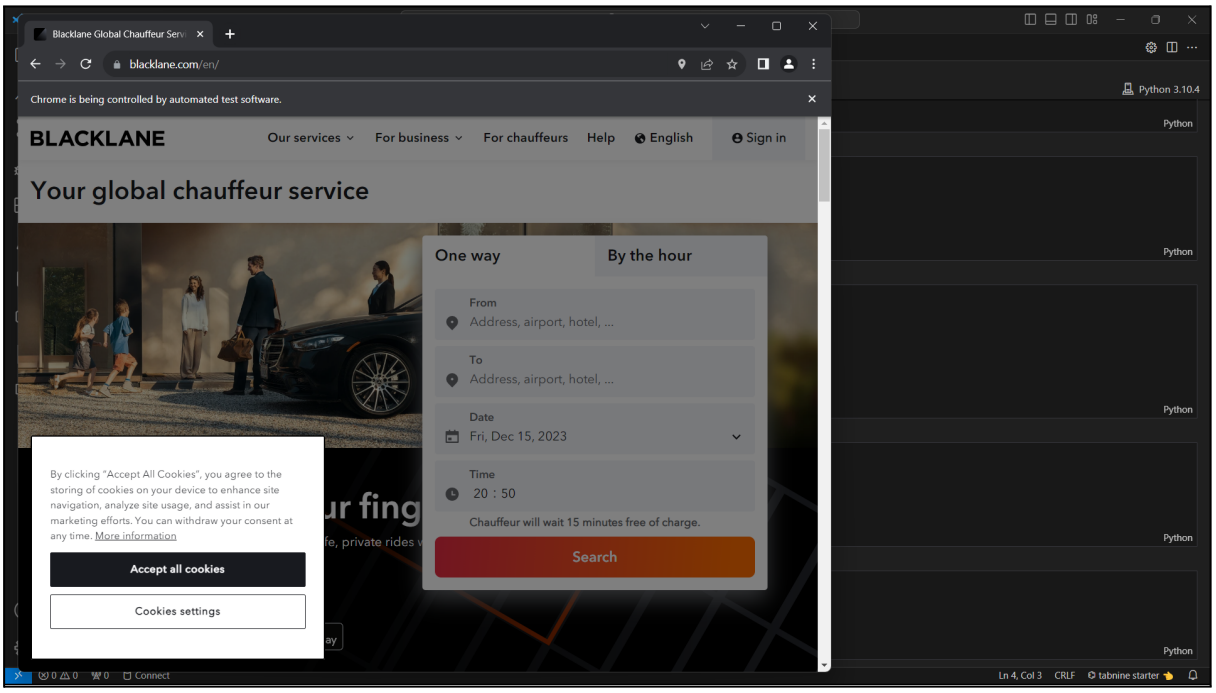
  This code initialises a Selenium WebDrier from the Chrome browser. It launches a new Chrome browser window and navigates to the Blacklane website. The *'webdriver.Chrome'* and *'Service'* classes are to facilitate browser automation for interactions and later data retrieval.

- **Cookie Acceptance**

  The script clicks the cookie acceptance button using Selenium *'WebDriverWait'* and *'element_to_be_clickable'* methods.

```
cookie_bttn = WebDriverWait(driver, 10).until(
    EC.element_to_be_clickable((By.ID, 'onetrust-accept-btn-handler'))
)

cookie_bttn.click()

time.sleep(5)
```

  This code uses Selenium's WebDriverWait to locate and wait for a clickable element with the HTML identification (ID) '*onetrust-accept-btn-handler*' on the webpage. Once the button is located by the bot, it is clickable. A wait time of seconds (*'time.sleep(5)'*) helps to allow for any associated actions to be completed before the bot clicks the button. This ensures the proper execution of subsequent actions in the script.



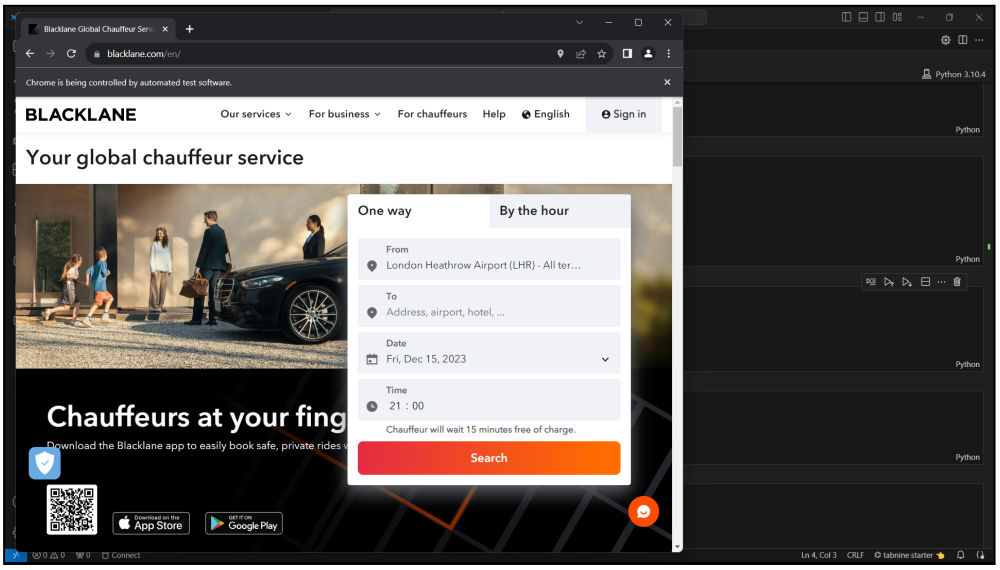
- **Pickup and Drop off Inputs**

  The script uses *'execute_script'* to set pickup and drop-off input fields' values.

```
# Find the 'fromTransfer' input field
from_transfer_input = driver.find_element('id', 'fromTransfer')
driver.execute_script("arguments[0].value = 'London Heathrow Airport (LHR) - All terminals';", from_transfer_input)

time.sleep(3)
```

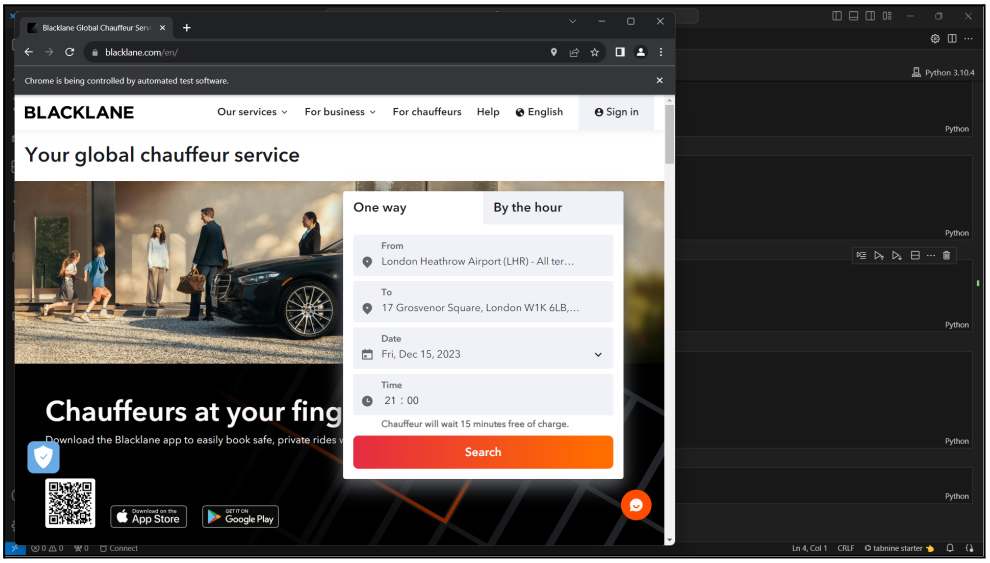
  This code located the input field for the pickup location on the Blacklane website using its unique HTML identification (ID) on the webpage (*'fromTransfer'*). It then uses the Selenium *'executie_script'* method to set the input fields value to *'London*

Heathrow Airport (LHR) - All terminals'. 'time.sleep(3)' adds a 3-second delay to allow for any updates or redirection on the webpage before proceeding with it next instruction.




```
drop_off_transfer_input = driver.find_element('id', 'to')
driver.execute_script("arguments[0].value = '17 Grosvenor Square, London W1K 6LB, UK';", drop_off_transfer_input)

time.sleep(3)
```

This code locates the drop-off location input field on the Blacklane website using its HTML ID attribute (*'to'*). It then sets the value of the input field to *'17 Grosvenor Square, London W1K 6LB, UK'.* The '*time.sleep(3)*' delay ensures a brief pause, allowing the page to reflect the updated input before proceeding with further instructions.

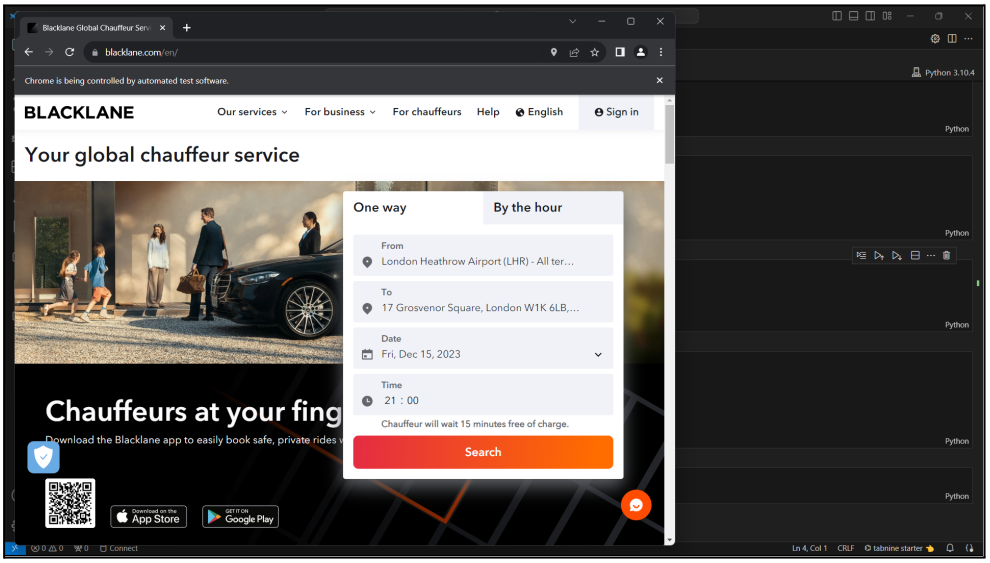


- **Search button Click**

  The search button is then clicked on the Blacklane website using *'element_to_be_clickable'* & *'click()'* functions.

```
search_button = WebDriverWait(driver, 10).until(
    EC.element_to_be_clickable((By.CSS_SELECTOR, 'button[data-tracking="submitButtonTransfer"]'))
)
search_button.click()

time.sleep(3)
```

  This code cell uses Selenium WebDriverWait to wait up to 10 seconds for the '*Search*' button on Blacklane's website to become clickable when both input fields have been populated. It then locates the button using a CSS selector with the HTML attribute '*data-tracking*' set to '*submitButtonTransfer*'. Once the button has been found it is clicked, initiating a search for a car transfer on the next page.



Given the absence of a unique identifier for the '*Search*' button, I resorted to using the '*data-tracking*' attribute to locate the button. However, despite many attempts with other HTML attributes such as *'class', 'type*' and *'data-testid',* none of them produced the desired result of interacting with the search button and proceeding with the search, which is to display the different services and prices offered by Blacklane.

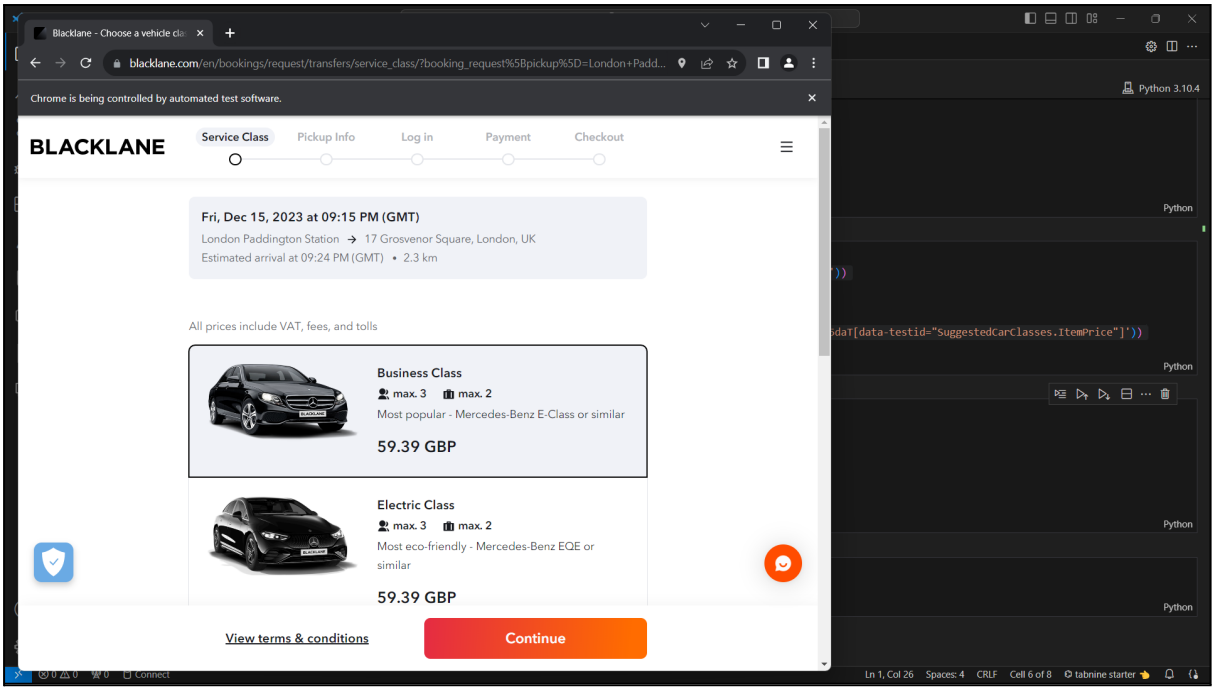
As a result of this, I was forced to skip this step and move on to the next which was to identify the different services offered by Blacklane (business class, electric class, SUV and first class) and their respective prices and store them in a Pandas data frame. This was done using the following code:

```
service_elements = WebDriverWait(driver, 10).until(
    EC.presence_of_all_elements_located((By.CLASS_NAME, 'SuggestedCarClasses_class__kH7h5'))
)

price_elements = WebDriverWait(driver, 10).until(
    EC.presence_of_all_elements_located((By.CSS_SELECTOR, 'p.SuggestedCarClasses_price__95daT[data-testid="SuggestedCarClasses.ItemPrice"]'))
)
```
✓ 0.0s

Here Selenium is used to wait for the elements on the website. It specifically targets the elements class 'SuggestedCarClasses_class__kH7h5' to retrieve the service information and elements with the CSS selector *'p.SuggestedCarClasses_price__95daT[data-testid="SuggestedCarClasses.ItemPrice"]'* to extract the price information. The '*WebDriverWait*' ensures that the elements are fully loaded before acquiring them.

```
services_prices_data = []

for service_element, price_element in zip(service_elements, price_elements):
    service = service_element.text
    price = price_element.text
    services_prices_data.append({'Service': service, 'Price': price})
```
✓ 2.2s

Here a For Loop is used to iterate through the different services and price elements using zip. It will then extract the text from each of the elements adn append a dictionary containing the service and price into a list. The results of this will be a list of dictionaries, where each dictionary represents a different service and its corresponding price according to the Blacklane website.



```
df = pd.DataFrame(services_prices_data)
print('Heathrow Airport to 17 Grosvenor Square', df)
```

✓ 0.0s

```
Heathrow Airport to 17 Grosvenor Square            Service       Price
0    Business Class  104.06 GBP
1    Electric Class  104.06 GBP
2  Business Van/SUV  190.59 GBP
3       First Class  137.28 GBP
```

Finally, a Pandas DataFrame (‘df’) is created from the data ‘*services_prices_data*’ and prints it with the header ‘*Heathrow Airport to 17 Grosvenor Square*’ the data frame contains information about the various services and their corresponding prices for the specialised routes

```
df = pd.DataFrame(services_prices_data)
print('Gatwick Airport to 17 Grosvenor Square', df)
```

✓ 0.0s

```
Gatwick Airport to 17 Grosvenor Square            Service       Price
0    Business Class  185.14 GBP
1    Electric Class  185.14 GBP
2  Business Van/SUV  342.74 GBP
3       First Class  227.95 GBP
```

```
df = pd.DataFrame(services_prices_data)
print('Paddington Station to 17 Grosvenor Square', df)
```

✓ 0.0s

```
Paddington Station to 17 Grosvenor Square            Service       Price
0    Business Class   58.48 GBP
1    Electric Class   58.48 GBP
2  Business Van/SUV  114.82 GBP
3       First Class   99.91 GBP
```

Tableau Superstore

Product with the best GMP for 2022 and 2023:
Recycled Interoffice Envelopes with String and Button Closure, 10 x 3
Formula used:

=INDEX(Orders!Q2:Q2652, MATCH(MAX(IF(YEAR(Orders!C2:C2652) = 2022, Orders!W2:W2652)), Orders!W2:W2652, 0))

=INDEX(Orders!Q2:Q6014, MATCH(MAX(IF(YEAR(Orders!C2:C6014) = 2023, Orders!W2:W6014)), Orders!W2:W6014, 0))

- This formula retirees the product that is associated with the maximum gross profit in the years 2022 & 2023. The INDEX and MATCH functions are used to retire the product name by pairing the maximum gross profit value.

The product that made the most profit in 2022 and 2023:
Recycled Interoffice Envelopes with String and Button Closure, 10 x 3
Formula used:

=INDEX(Orders!Q2:Q2652, MATCH(MAX(IF(YEAR(Orders!C2:C2652) = 2022, Orders!W2:W2652)), Orders!W2:W2652, 0))

- This formula retrieves the product name that is associated with the maximum gross profit in 2022. The INDEX and MATCH functions work together to find the corresponding product name by matching the maximum gross profit value.

The region with the most sales in 2022:
Central
Formula used:
=INDEX(Orders!M2:M2652, MATCH(MAX(IF(YEAR(Orders!C2:C2652) = 2022, Orders!R2:R2652)), Orders!R2:R2652, 0))

- This formula uses the INDEX and MATCH functions to find the region that is associated with the maximum sales in the year 2022.

The region with the best profit in 2023:
West
Formula used:
=INDEX(Orders!M2:M2652, MATCH(MAX(IF(YEAR(Orders!C2:C2652) = 2023, Orders!X2:X2652)), Orders!X2:X2652, 0))

- This formula identifies the region that is linked with the maximum profit in the year 2023.