Python Programming

Lesson 5 – Web Scraping

Lesson 5 - Outline

- HTTP request in Python (Recap)
- Identify elements on a Website
- Use of selenium
- Simulate key-in and mouse click
- Web scraping examples

HTTP request in Python

What is HTTP request

- <u>**Definition**</u>

  “An HTTP request is **made by a client**, to a named host, which is located on a server. The aim of the request is to access a **resource on the server**.”

- <u>**Examples of resource**</u>
  - Webpage
  - Media – Image, Video, Audio, etc.
  - Data from Database

HTTP request in Python

- **Usage of requests module**
  - **Request:** The `requests` module enables to to send HTTP requests using Python
  - **Response:** The HTTP request returns a Response Object with all the response data *(content, encoding, status, etc.)*

- **Install of requests module**

```
pip install requests
```

HTTP request in Python

- **How to install module to another path (e.g. without admin rights)**

1. Open the command prompt (cmd) at your windows
2. Type “pip install --user <module name>” to install the module at user folder
   e.g. `pip install --user requests`
3. Type “pip show <module name>” to locate the exact location of module
   e.g. `pip show requests`
4. Open the Python IDLE
5. Type the following command to append the location of module to the path
   ```
   import sys
   sys.path.append(r'<location of module>')
   ```

HTTP request in Python

- <u>**Syntax of using requests**</u>

```
requests.methodname(params)
```

- <u>**Methods of requests**</u>

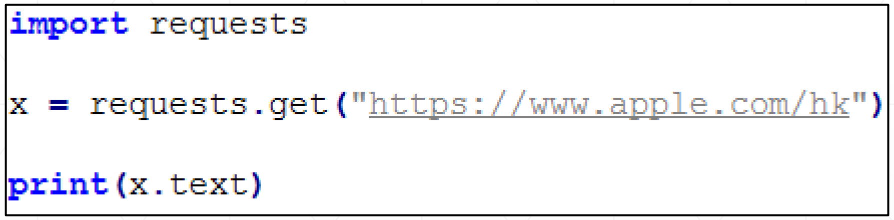
Method	**Description**
delete(*url*, *args*)	Sends a DELETE request to the specified url
get(*url, params, args*)	**Sends a GET request to the specified url**
head(*url*, *args*)	Sends a HEAD request to the specified url
patch(*url*, *data, args*)	Sends a PATCH request to the specified url
post(*url, data, json, args*)	**Sends a POST request to the specified url**
put(*url*, *data, args*)	Sends a PUT request to the specified url
request(*method, url, args*)	**Sends a request of the specified method to the specified url**

HTTP request in Python

- **Import requests module**

```
import requests
```

- **Example – simple HTTP request**



```
import requests

x = requests.get("https://www.apple.com/hk")

print(x.text)
```

- **Explanation**

Create HTTP GET request to a web site

HTTP request in Python

- **Example – request with parameters**

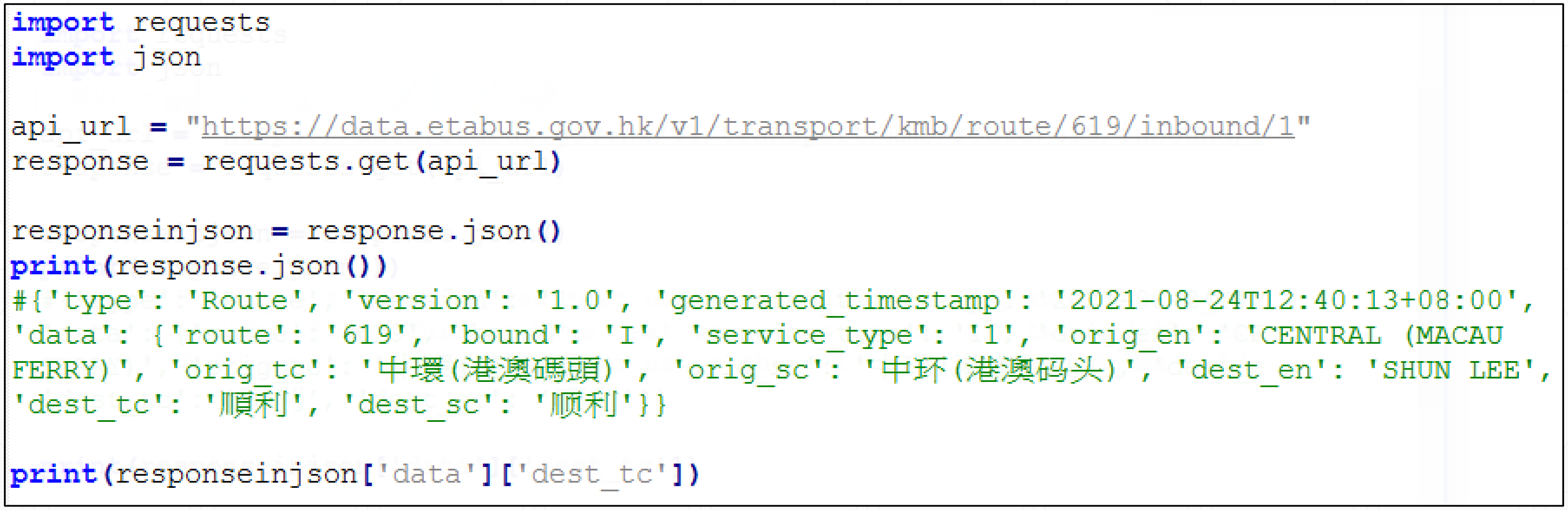
```
import requests

#payload = {'key1': 'value1', 'key2': 'value2'}
payload = {"keywordForQuickSearch": "programmer"}

x = requests.get("https://www.ctgoodjobs.hk/ctjob/listing/joblist.asp", params=payload)
#https://www.ctgoodjobs.hk/ctjob/listing/joblist.asp?keywordForQuickSearch=programmer

print(x.url)
print(x.text)
```

- **Explanation**

  Create HTTP GET request with parameters to a web site

HTTP request in Python

- **Example – request according API**



```
import requests
import json

api_url = "https://data.etabus.gov.hk/v1/transport/kmb/route/619/inbound/1"
response = requests.get(api_url)

responseinjson = response.json()
print(response.json())
#{'type': 'Route', 'version': '1.0', 'generated_timestamp': '2021-08-24T12:40:13+08:00',
'data': {'route': '619', 'bound': 'I', 'service_type': '1', 'orig_en': 'CENTRAL (MACAU
FERRY)', 'orig_tc': '中環(港澳碼頭)', 'orig_sc': '中环(港澳码头)', 'dest_en': 'SHUN LEE',
'dest_tc': '順利', 'dest_sc': '顺利'}}

print(responseinjson['data']['dest_tc'])
```

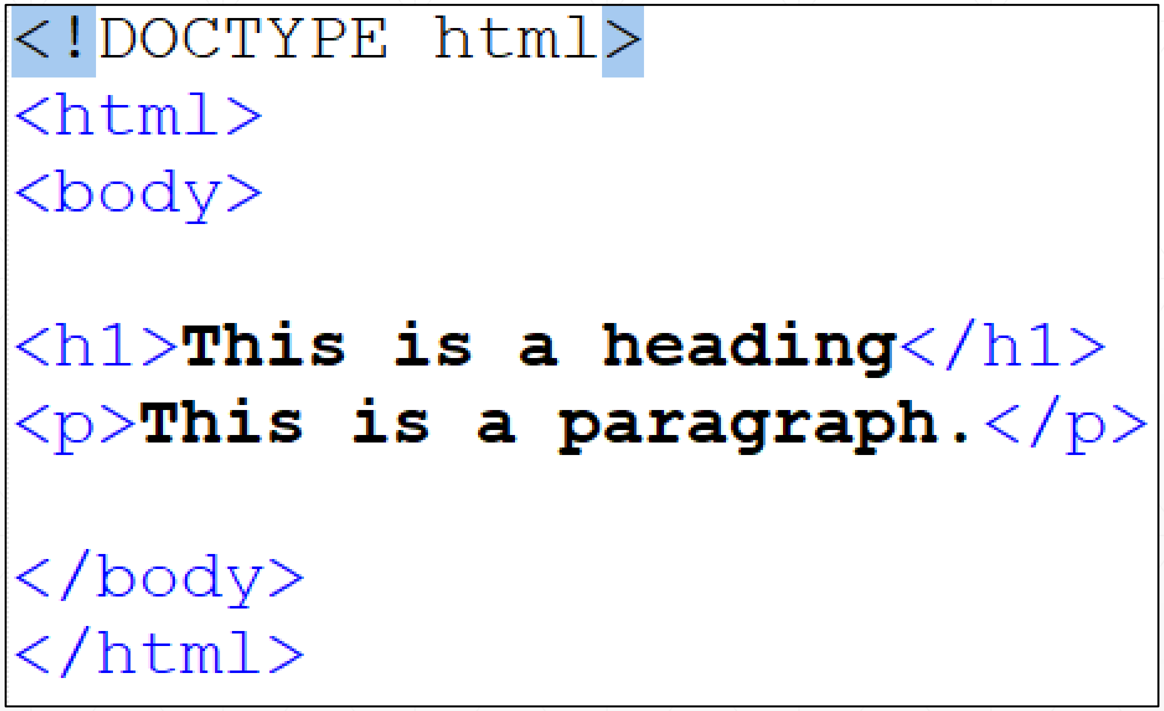
- **Explanation**

Create HTTP GET request according the API of a web site

Identify elements on a Website

Identify elements on a Website

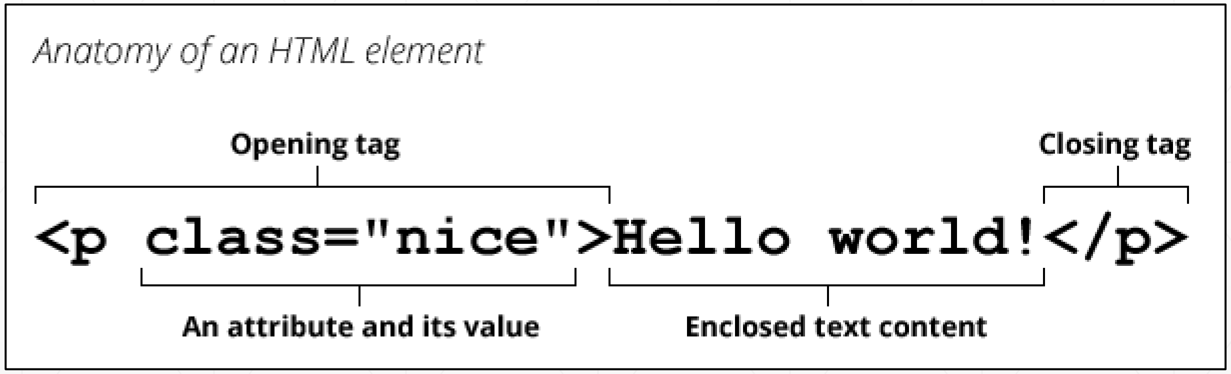
- **Example of HTML code**



```
<!DOCTYPE html>
<html>
<body>

<h1>This is a heading</h1>
<p>This is a paragraph.</p>

</body>
</html>
```

Identify elements on a Website

- **<u>What are elements on a Website</u>**
  - A part of a webpage.
  - May contain a data e.g. a text, an image or a link.
  - A typical element includes **an opening tag** with **some attributes**, **enclosed text content**, and **a closing tag**.
- **<u>Example of HTML code</u>**



```
<p class="nice">Hello world!</p>
```

Identify elements on a Website

- **What are the common tags on a Website**
  => how web browser will format and display the content

Tag	Description
<!doctype>	Defines a document type
<html>	Define a document is a HTML markup language
<body>	Defines a main section(body) part in HTML document
<a>	Use for link in internal/external web documents
<h1> to <h6>	Defines a Headings level from 1 to 6 different sizes
<p>	Used to represents a paragraph text
<style>	Used to add CSS style to an HTML document
<ol>	Defines an ordered list of items
<ul>	Defines an unordered list of items
<li>	Define a list item either ordered list or unordered list
<table>	Used to defines a table in an HTML document
<tr>	Defines a row of cells in a table
<td>	Used for creates standard data cell in HTML table

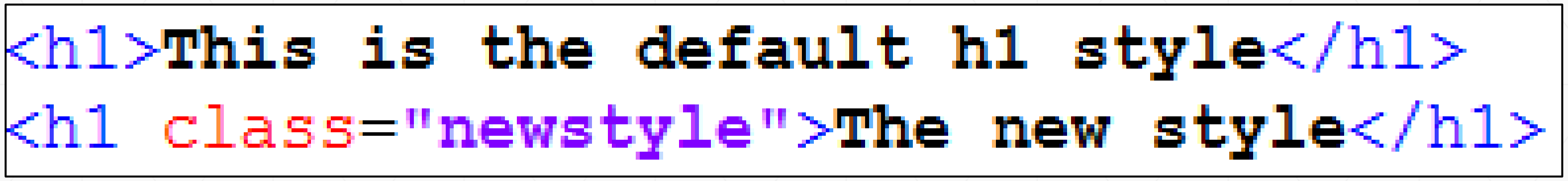
Identify elements on a Website

- **Attributes on a Website**
  - Inside the opening tag to control the element's behaviour
- **Example of Attributes**

```
<p style="color:blue">A blue paragraph.</p>

<a href="https://www.google.com/">Google!</a>
```

Identify elements on a Website

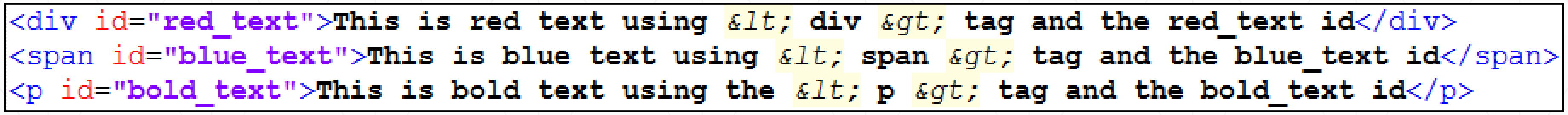
- **<u>Class Attribute on a Website</u>**
  - The class is an attribute which specifies one or more class names for an HTML element
  - The class attribute can be used on any HTML element
  - The class name can be used by CSS and JavaScript to perform certain tasks for elements with the specified class name
- **<u>Example of class attribute</u>**



```html
<h1>This is the default h1 style</h1>
<h1 class="newstyle">The new style</h1>
```

Identify elements on a Website

- **ID Attribute on a Website**
  - The id global attribute defines an identifier (ID)
  - Must be unique in the whole document
  - Identify the element when linking to e.g. JavaScript / CSS
- **Example of id attribute**



```
<div id="red_text">This is red text using &lt; div &gt; tag and the red_text id</div>
<span id="blue_text">This is blue text using &lt; span &gt; tag and the blue_text id</span>
<p id="bold_text">This is bold text using the &lt; p &gt; tag and the bold_text id</p>
```

Use of selenium

Use of selenium

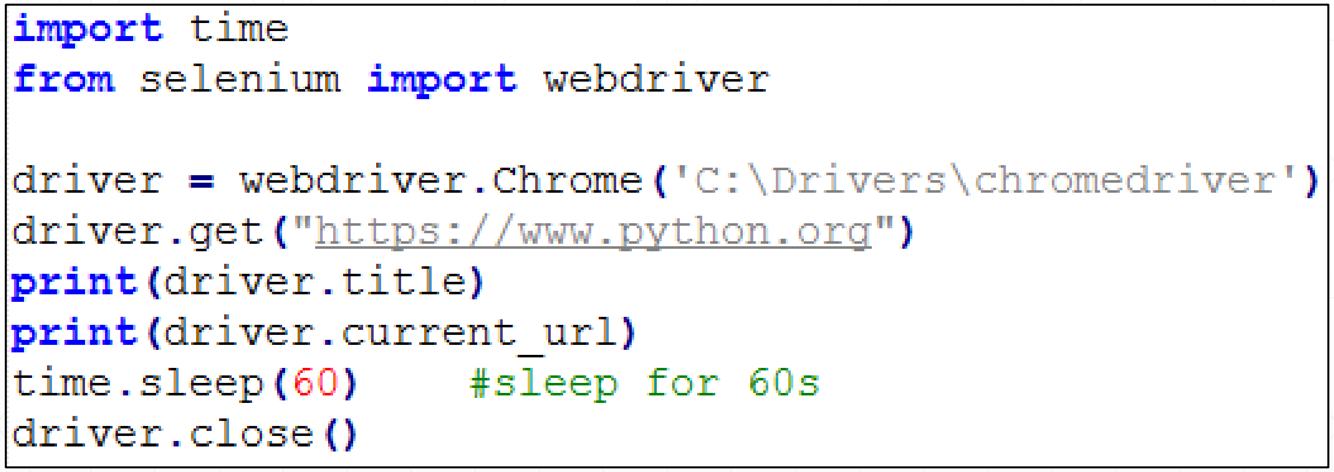
- **Preparation before Web Scraping**
  1. Install selenium module
  2. Download and make use of Chrome Driver

Use of selenium

- **Install selenium module**
  1. Open the command prompt (cmd) at your windows
  2. Type `pip install selenium`
     (refer to previous slides for steps if you don't have the admin right)
- **Download and prepare the Chrome driver**
  1. Browse https://chromedriver.chromium.org/downloads
  2. Find open your Chrome Browser and check the version
  3. Download an appropriate version of chromedriver matches your Chrome Browser
  4. Place the `chromedriver.exe` at a path (let's say `C:\Drivers`)

Use of selenium

- **Try the first example of selenium**

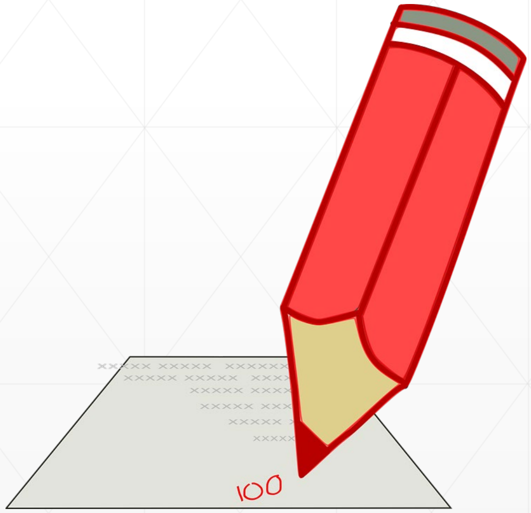


```python
import time
from selenium import webdriver

driver = webdriver.Chrome('C:\Drivers\chromedriver')
driver.get("https://www.python.org")
print(driver.title)
print(driver.current_url)
time.sleep(60)     #sleep for 60s
driver.close()
```

- **Explanation**
  - Python using the selenium to browse a webpage using chromedriver *(the Chrome will be opened)*

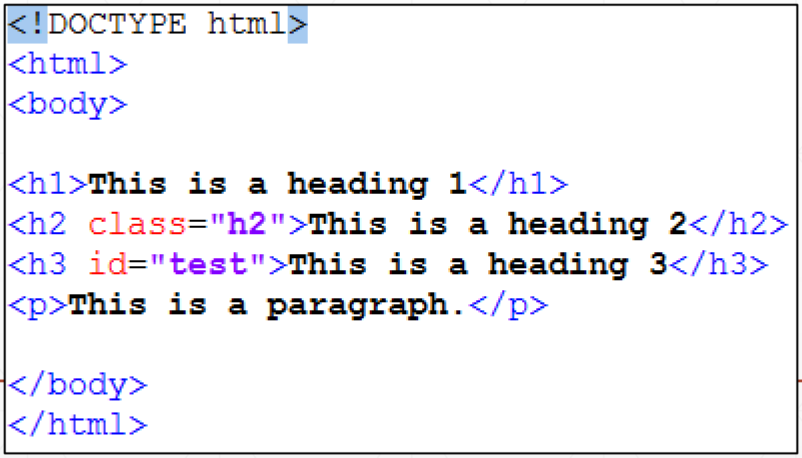
Use of selenium

- Try to browse another website!



Use of selenium

- **How to identify an HTML element**
  1. Tag name
  2. Class name
  3. IDs
  4. XPath
- **Try with the following example**



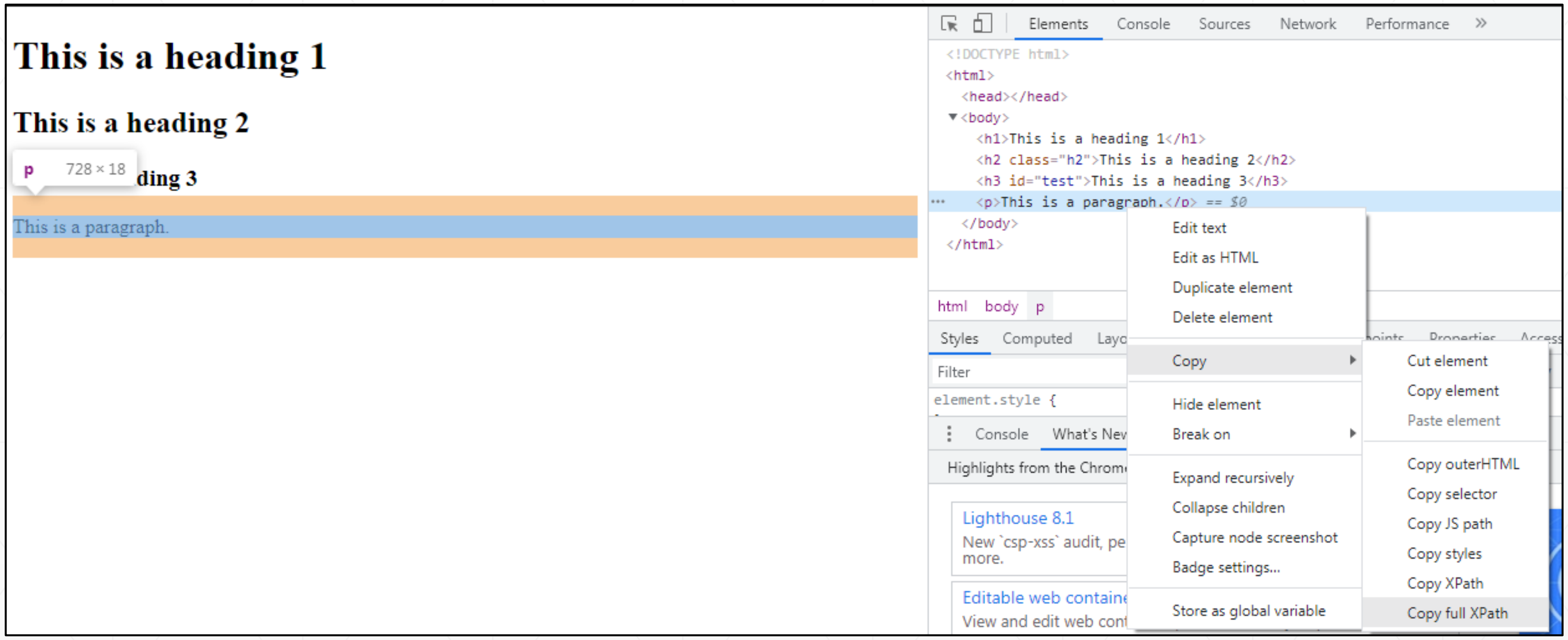
```html
<!DOCTYPE html>
<html>
<body>

<h1>This is a heading 1</h1>
<h2 class="h2">This is a heading 2</h2>
<h3 id="test">This is a heading 3</h3>
<p>This is a paragraph.</p>

</body>
</html>
```

Use of selenium

- Try `<element>.text` after located the element
  e.g. `h1.text`
- **Example - Tag name**
  - `Selenium 3: h1 = driver.find_element_by_tag_name('h1')`
  - `Selenium 4: h1 = driver.find_element(By.TAG_NAME, 'h1')`
- **Example - Class name**
  - `Selenium 3: h2 = driver.find_element_by_class_name('h2')`
  - `Selenium 4: h2 = driver.find_element(By.CLASS_NAME, 'h2')`
- **Example - Id name**
  - `Selenium 3: h3 = driver.find_element_by_id('test')`
  - `Selenium 4: h3 = driver.find_element(By.ID, 'test')`
- **Example - XPath**
  - `Selenium 3: p = driver.find_element_by_xpath('/html/body/p')`
  - `Selenium 4: p = driver.find_element(By.XPATH, '/html/body/p')`

Use of selenium

- **Remark of XPath**
  - XPath can be found in **Chrome Developer Tools** (i.e. F12)



This is a heading 1

This is a heading 2

This is a heading 3

This is a paragraph.

```
<!DOCTYPE html>
<html>
  <head></head>
  ▼<body>
    <h1>This is a heading 1</h1>
    <h2 class="h2">This is a heading 2</h2>
    <h3 id="test">This is a heading 3</h3>
    <p>This is a paragraph.</p> == $0
  </body>
</html>
```

Simulate key-in and mouse click

Simulate key-in and mouse click

- **Simulate key-in of keyboard**
- **Before use - Import the module of webdriver and Keys**

Selenium 3:

```
from selenium import webdriver
from selenium.webdriver.common.keys import Keys
```

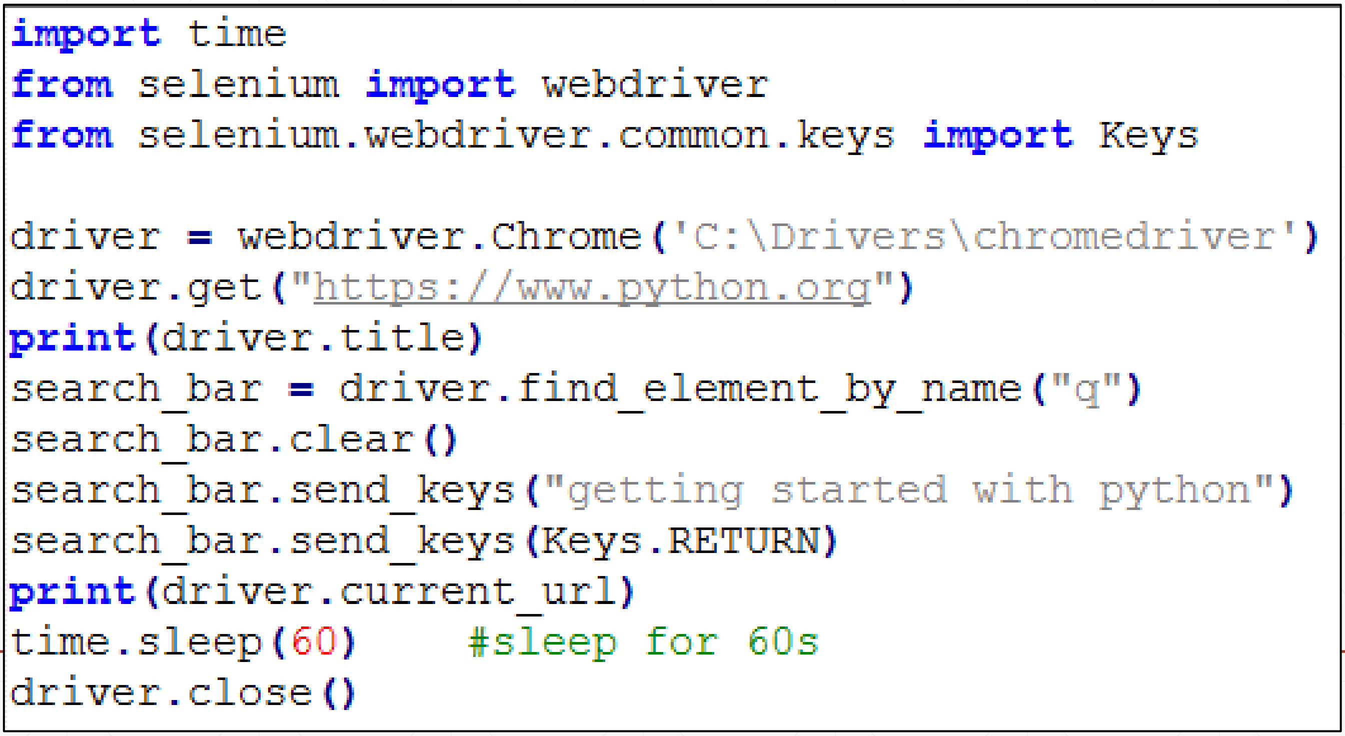
Selenium 4:

```
from selenium import webdriver
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.common.by import By
```

  - **Syntax**
  - `<element>.send_keys("TEXT TO BE SENT!")`
  - **Examples**
  - `search_bar.send_keys("getting started with python")`
  - `search_bar.send_keys(Keys.RETURN)`
  - **Explanation**
  - We can simulate to send a sequence of text or ‘Enter’ (i.e. `Keys.RETURN`) to the designated element

Simulate key-in and mouse click

- **Simulate key-in of keyboard**
- **Try the example together!**
  Remark: Appropriate sleep may help if everything is too fast



```python
import time
from selenium import webdriver
from selenium.webdriver.common.keys import Keys

driver = webdriver.Chrome('C:\Drivers\chromedriver')
driver.get("https://www.python.org")
print(driver.title)
search_bar = driver.find_element_by_name("q")
search_bar.clear()
search_bar.send_keys("getting started with python")
search_bar.send_keys(Keys.RETURN)
print(driver.current_url)
time.sleep(60)    #sleep for 60s
driver.close()
```

Simulate key-in and mouse click

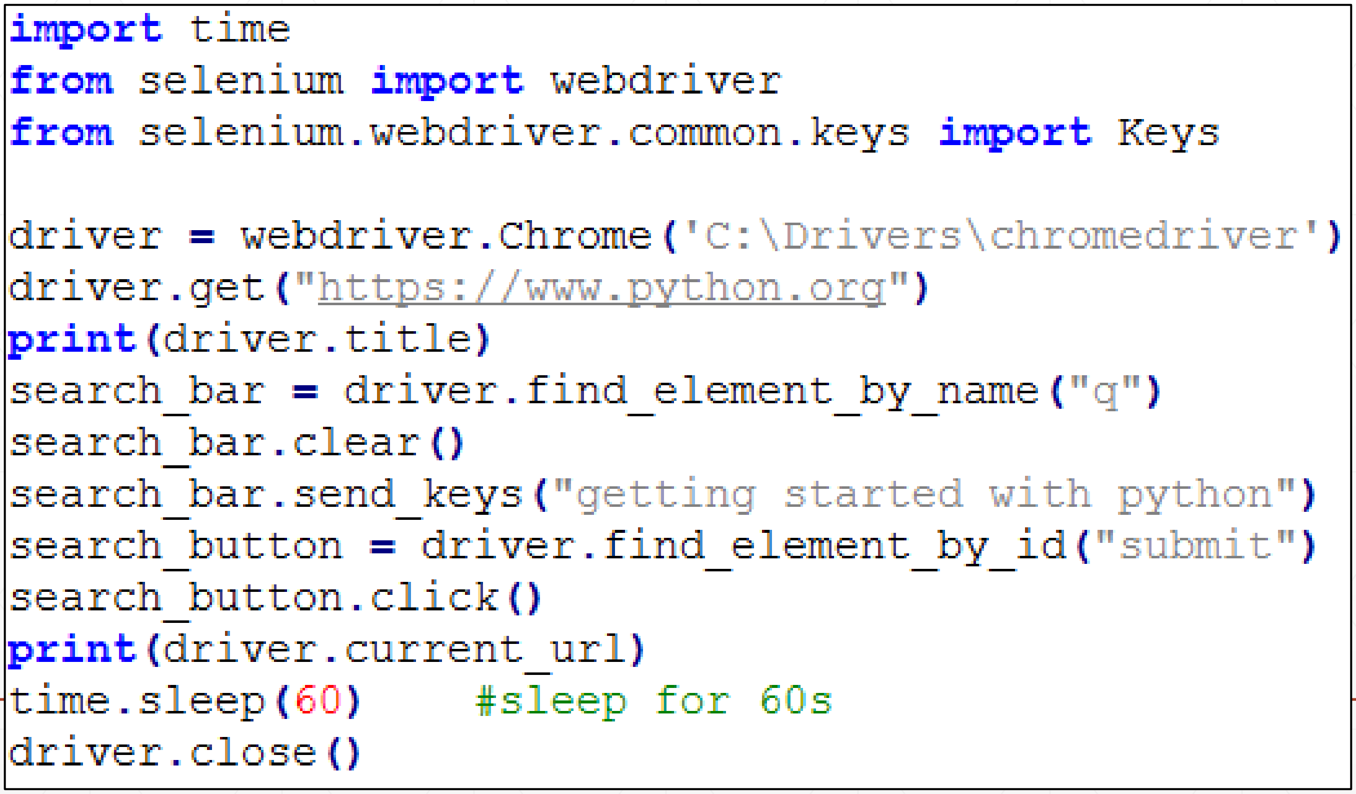
- **Simulate mouse click**
- **Before use - Import the module of webdriver**

```
from selenium import webdriver
```

  - **Syntax**
  - `<element>.click()`
  - **Examples**
  - `search_button.click()`
  - **Explanation**
  - We can simulate a mouse click on a designated element

Simulate key-in and mouse click

- **Simulate mouse click**
- **Try the example together!**

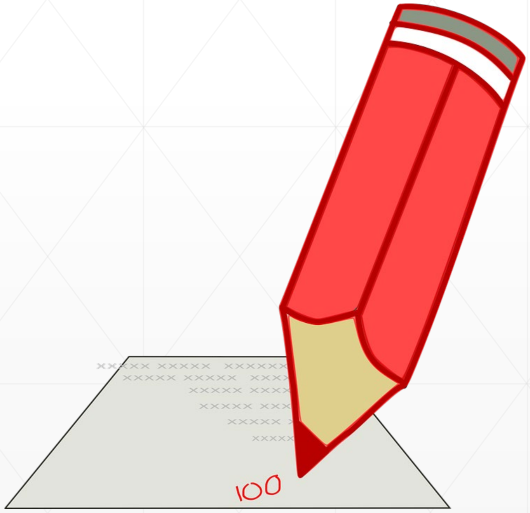


```python
import time
from selenium import webdriver
from selenium.webdriver.common.keys import Keys

driver = webdriver.Chrome('C:\Drivers\chromedriver')
driver.get("https://www.python.org")
print(driver.title)
search_bar = driver.find_element_by_name("q")
search_bar.clear()
search_bar.send_keys("getting started with python")
search_button = driver.find_element_by_id("submit")
search_button.click()
print(driver.current_url)
time.sleep(60)    #sleep for 60s
driver.close()
```

Web scraping examples

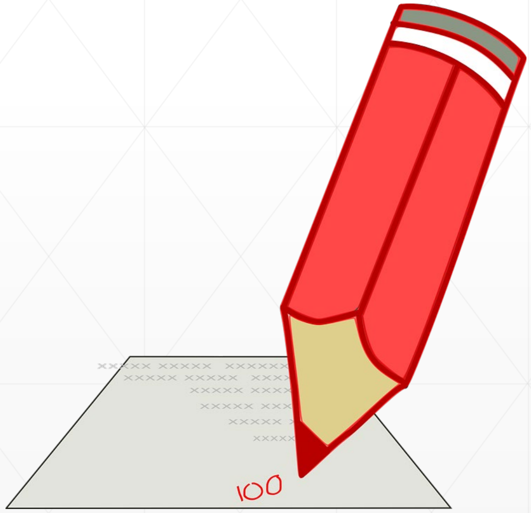
Web scraping examples

- **<u>Try to login your Moodle / any testing account</u>**
- Keep your password secretly!
  (*E.g. read a file for your password instead of putting it on the code*)



Web scraping examples

- Try to download the latest python release file



Web scraping examples

- **Introduce of Headless mode**
- Without opening the browser physically and getting the things done
- Before use - Import the module of webdriver

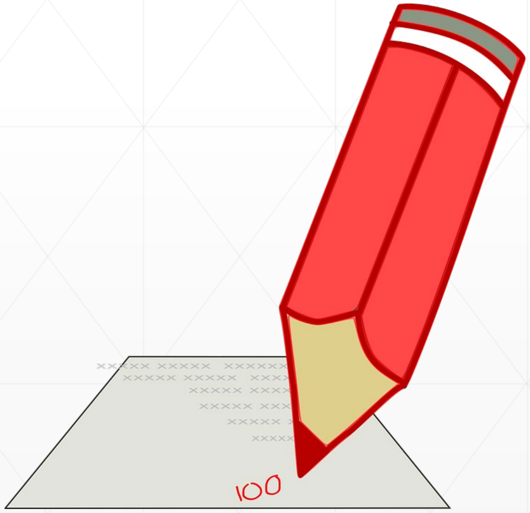
```
from selenium import webdriver
from selenium.webdriver.chrome.options import Options
```

- **Syntax**

```
options = Options()

options.headless = True

options.add_argument("--window-size=1920,1200")

driver = webdriver.Chrome(options=options,
executable_path="C:\Drivers\chromedriver")
```

Web scraping examples

- Headless Chrome
- Try the example together!



Web scraping examples

- **Introduce of Screenshot**
- Without opening the browser physically and getting the things done
- Before use - Import the module of webdriver

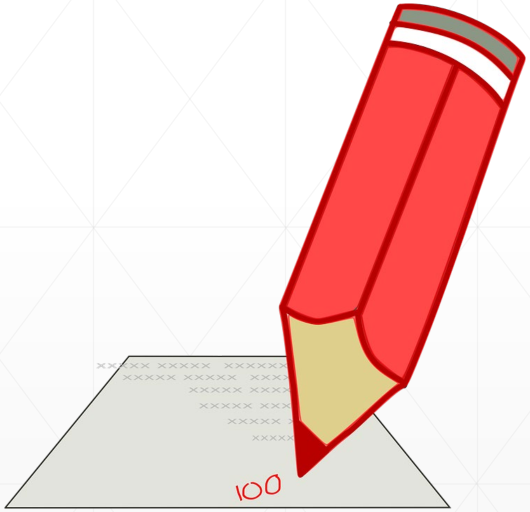
```
from selenium import webdriver
```

- **Syntax**

```
driver.save_screenshot('screenshot.png')
```

Web scraping examples

- Implementation of Image Web Scrapping using Selenium Python from Google Search



Reference:
https://www.analyticsvidhya.com/blog/2020/08/web-scraping-selenium-with-python/

Thank you