目錄

網頁測試自動化 for Python

講師：劉艾霖 (A-Lin)

此課程面向哪些人

- 剛畢業學生想從事軟體產業
- 手動軟體測試人員想轉為自動化測試人員
- 自動化軟體測試人員想加強專業技能

合作洽談

授課 / 出版需求，請來信至 alin.code@gmail.com

授權

教材使用授權

此教材為 18 小時授課的內容

- 德明財經科技大學 - 資管系 (2021 年)

勘誤通知

如發現內容勘誤，歡迎利用這些管道和我聯繫：

- 使用 GitHub Issues 回報錯誤
- 發 pull request
- 寄信到 alin.code@gmail.com

最新更新時間：2021/03/14

職缺範例

QA / QC

- iKala 愛卡拉: https://www.cakeresume.com/companies/iKala/jobs/7d09af
- 透視數據有限公司: https://www.cakeresume.com/companies/PicSee/jobs/qa-engineer-edb4ea
- MaiCoin: https://www.cakeresume.com/companies/maicoin/jobs/quality-assurance-engineer-fe043a
- OneDegree: https://www.cakeresume.com/companies/onedegree/jobs/software-qa-engineer-software-test-engineer-61a274

SDET

- https://www.cakeresume.com/companies/iKala/jobs/ikala-cloud-software-engineer-in-test-sdet-straas
- https://www.cakeresume.com/companies/maicoin/jobs/software-engineer-in-test-f73363
- https://www.cakeresume.com/companies/tomofun/jobs/8960d0

測試工讀生 / 實習生

- https://www.cakeresume.com/companies/maicoin/jobs/qa-work-student
- https://www.cakeresume.com/companies/gogoout/jobs/test-engineer-intern

延伸閱讀

- 測試工程師和想像中的不一樣：在趨勢科技的暑期實習心得 - 傑瑞窩在這

準備開發環境

- Python
- PyCharm IDE
- 安裝瀏覽器驅動
- 安裝 Selenium Webdriver 模組

安裝

安裝 Python

- 下載安裝檔：https://www.python.org/downloads/
- 目前範例使用 Python 3.8

PyCharm IDE

- 下載安裝檔：https://www.jetbrains.com/pycharm/download/
- 安裝 Community 版本

安裝 Driver Binaries

- Chrome: https://sites.google.com/a/chromium.org/chromedriver/downloads
- Edge: https://developer.microsoft.com/en-us/microsoft-edge/tools/webdriver/
- Internet Explorer: https://github.com/SeleniumHQ/selenium/wiki/InternetExplorerDriver
- Firefox: https://github.com/mozilla/geckodriver/releases
- Safari: https://webkit.org/blog/6900/webdriver-support-in-safari-10/

或從

https://www.selenium.dev/downloads/ 找到 Platforms Supported by Selenium 的位址，並下載對應的驅動程式。

或從

https://github.com/christian-bromann/awesome-selenium#driver

測試環境

1. 下載 Chrome 驅動程式
2. 新增一個測試專案

3. 新增一個檔案叫 test.py

```
# test.py
from selenium import webdriver

# driver = webdriver.Edge("./msedgedriver.exe")
driver = webdriver.Chrome("./chromedriver.exe")

# for Mac
# driver = webdriver.Chrome("./chromedriver")

driver.get("http://www.python.org")
```

執行測試程式

```
pip install selenium
python test.py
```

練習題 😎

1. 試著照上面的步驟，完成 Chrome 瀏覽器的測試環境。
2. 試著改用 Edge 瀏覽器來跑測試

補充

環境常見問題

- 瀏覽器版本跟驅動程式不一致

```
selenium.common.exceptions.SessionNotCreatedException: Mess
Current browser version is 89.0.774.50 with binary path C:\
```

- 驅動程式的版本一直更新

參考文獻

- https://selenium-python.readthedocs.io/installation.html

快速入門

第一個最簡單的範例

建立一個檔案，檔名叫 `python_org_search.py` 。

```
# python_org_search.py
from selenium import webdriver
from selenium.webdriver.common.keys import Keys

driver = webdriver.Chrome("./chromedriver")
driver.get("http://www.python.org")
assert "Python" in driver.title
elem = driver.find_element_by_name("q")
elem.clear()
elem.send_keys("pycon")
elem.send_keys(Keys.RETURN)
assert "No results found." not in driver.page_source
driver.close()
```

執行測試程式

```
pip install selenium
python python_org_search.py
```

詳細解說

Step1: 匯入模組

- webdriver 可模擬人操作瀏覽器，支援 Firefox, Chrome, IE 等各類型的瀏覽器。
- 而 `Keys` 類別，提供了模擬鍵盤操作行為 (e.g. RETURN, F1, ALT)

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

Step2: 建立 webdriver 的 instance

```
driver = webdriver.Chrome("./chromedriver")
```

Step3: 前往給定的 URL 網頁

driver.get 方法的用途是前往給定的 URL 網頁

```
driver.get("http://www.python.org")
```

Step4: 斷言 - 確認標題中包含 Python

> 在程式設計中，斷言（assertion）是一種放在程式中的一階邏輯（如一個結果為真或是假的邏輯判斷式），目的是為了標示與驗證程式開發者預期的結果－當程式執行到斷言的位置時，對應的斷言應該為真。若斷言不為真時，程式會中止執行，並給出錯誤訊息。
> ***資料來源: Wiki***

```
assert "Python" in driver.title
```

Step5: 查找元素

```
elem = driver.find_element_by_name("q")
```

Step6: 執行模擬鍵盤的送出行為 + 提交頁面

```
elem.clear()
elem.send_keys("pycon")
elem.send_keys(Keys.RETURN)
```

Step7: 斷言

提交頁面後，如果有的話應該得到結果。為了確保找到一些結果，請聲明:

```
assert "No results found." not in driver.page_source
```

Step8: 關閉瀏覽器視窗

你也可以使用 quit() 替代 close()

```
driver.close()
```

定位元素 (Locating Elements)

有多種策略可以找到元素，你可以跟自己的情況選擇使用。

單一元素

即使有多個也只返回一個

```
find_element_by_id
find_element_by_name
find_element_by_xpath
find_element_by_link_text
find_element_by_partial_link_text
find_element_by_tag_name
find_element_by_class_name
find_element_by_css_selector
```

多個元素

如果是要一次取多個元素，則要加 `s` 。

```
find_elements_by_name
find_elements_by_xpath
find_elements_by_link_text
find_elements_by_partial_link_text
find_elements_by_tag_name
find_elements_by_class_name
find_elements_by_css_selector
```

另一種寫法

```
find_element
find_elements
```

使用範例

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

driver.find_element(By.XPATH, '//button[text()="Some text"]
driver.find_elements(By.XPATH, '//button')
```

```
ID = "id"
XPATH = "xpath"
LINK_TEXT = "link text"
PARTIAL_LINK_TEXT = "partial link text"
NAME = "name"
TAG_NAME = "tag name"
CLASS_NAME = "class name"
CSS_SELECTOR = "css selector"
```

透過 ID 來選取

常見於選取唯一的元素

```
<html>
  <body>
    <form id="loginForm">
      <input name="username" type="text" />
      <input name="password" type="password" />
      <input name="continue" type="submit" value="Login" />
    </form>
  </body>
</html>
```

```
login_form = driver.find_element_by_id('loginForm')
```

透過 Class Name 來選取

比較常見於一次選取多個元素

```
<html>
  <body>
    <p class="content">Site content goes here.</p>
  </body>
</html>
```

```
content = driver.find_element_by_class_name('content')
```

透過 Name 來選取

常見於選取表單元素

```
<html>
  <body>
    <form id="loginForm">
      <input name="username" type="text" />
      <input name="password" type="password" />
      <input name="continue" type="submit" value="Login" />
      <input name="continue" type="button" value="Clear" />
    </form>
  </body>
</html>
```

```
username = driver.find_element_by_name('username')
password = driver.find_element_by_name('password')
```

透過 CSS_SELECTOR 來選取

```
<html>
  <body>
    <p class="content">Site content goes here.</p>
  </body>
</html>
```

```
content = driver.find_element_by_css_selector('p.content')
```

- CSS Selectors Reference
- CSS Diner 小遊戲

總結

- 有這麼多種，到底該用哪一種?
- 如果 selector 規則很複雜的時候，有沒有其他變通方式?
- 其他不常使用的，下週再說明。

REPL

Read-Eval-Print Loop 簡稱 REPL，是一個簡單的交互式的開發環境，常使用於一次性的操作，例如語法測試或 debug，你可以透過輸入 `python` 開始它。

```
% python

Python 3.8.5 (default, Jul 21 2020, 10:42:08)
[Clang 11.0.0 (clang-1100.0.33.17)] on darwin
Type "help", "copyright", "credits" or "license" for more i
>>>
```

練習題：測試計算機

解答

```
# 1. import 要使用的類別
from selenium import webdriver
from selenium.webdriver.common.keys import Keys

# 2. 建立模擬人的行為的瀏覽器的物件
driver = webdriver.Chrome("./chromedriver.exe")

# 3. 前往特定的網址
driver.get("http://juliemr.github.io/protractor-demo/")

# 4. 驗證是不是在對的頁面
assert "Super Calculator" in driver.title

# 5. 選取元素
input1 = driver.find_element_by_css_selector("[ng-model='f
input2 = driver.find_element_by_css_selector("[ng-model='s
button = driver.find_element_by_id("gobutton")

# 6. 控制元素行為
input1.send_keys("1")
input2.send_keys("2")
button.send_keys(Keys.ENTER)

# 7. 關閉視窗
driver.quit()
```

鍵盤操作 (Keyboard)

模擬鍵盤類型的行為

sendKeys

模擬輸入值

```
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.common.keys import Keys
driver = webdriver.Firefox()

driver.get("http://www.google.com")

# Enter "webdriver" text and perform "ENTER" keyboard actio
driver.find_element(By.NAME, "q").send_keys("webdriver" + K
```

keyDown

keyDown 用於模擬按下修改鍵（CONTROL，SHIFT，ALT）的動作

```
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.common.keys import Keys
driver = webdriver.Chrome()

driver.get("http://www.google.com")

# Enter "webdriver" text and perform "ENTER" keyboard actio
driver.find_element(By.NAME, "q").send_keys("webdriver" + K

# Perform action ctrl + A (modifier CONTROL + Alphabet A) t
webdriver.ActionChains(driver).key_down(Keys.CONTROL).send_
```

keyUp

keyUp 用於模擬修改鍵（CONTROL，SHIFT，ALT）的向上（或）釋放操作

```
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.common.keys import Keys
driver = webdriver.Chrome()

driver.get("http://www.google.com")

# Store google search box WebElement
search = driver.find_element(By.NAME, "q")

action = webdriver.ActionChains(driver)

# Enters text "qwerty" with keyDown SHIFT key and after key
action.key_down(Keys.SHIFT).send_keys_to_element(search, "
```

clear

清除可編輯元素的內容。這僅適用於可編輯和可交互的元素，否則 Selenium 返回錯誤（無效的元素狀態（或）不可交互的元素）

```
from selenium import webdriver
from selenium.webdriver.common.by import By
driver = webdriver.Chrome()


driver.get("http://www.google.com")

# Store 'SearchInput' element
SearchInput = driver.find_element(By.NAME, "q")
SearchInput.send_keys("selenium")

# Clears the entered text
SearchInput.clear()
```

使用情境

- 預防頁面有預設的文字

設定視窗位置

我們只能測試我們看得到的區域，所以最後在一開始測試時，把視窗展開到最大。

Maximize window

最大化視窗 (工具列還會存在)

```
driver.maximize_window()
```

Minimize window

最小化視窗

```
driver.minimize_window()
```

> Note: This feature works with Selenium 4 and later versions.

Fullscreen window

全螢幕

```
driver.fullscreen_window()
```

TakeScreenshot

擷取螢幕畫面

```
from selenium import webdriver

driver = webdriver.Chrome("./chromedriver")
driver.get("http://www.google.com")

driver.save_screenshot('./image.png')
driver.quit()
```

使用情境

- 檢查是否有畫面破版的情況

練習題：登入練習

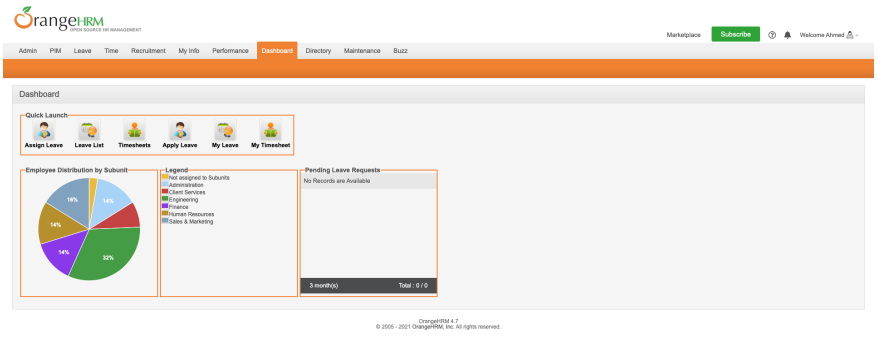
1. 透過自動化執行登入動作

複習 find_element 跟 send_keys 語法，並練習到新教的 submit 語法。

https://opensource-demo.orangehrmlive.com/

解答

2. 擷取一張登入後最大化視窗的截圖



練習目標

- 最大化視窗
- 截圖語法

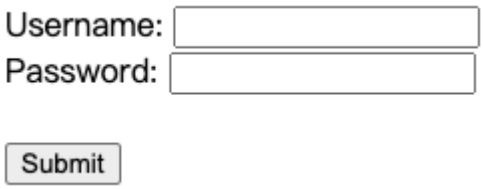
解答

```
from selenium import webdriver

driver = webdriver.Chrome("./chromedriver")
driver.get("https://opensource-demo.orangehrmlive.com/")
assert "OrangeHRM" in driver.title

# 最大化視窗
driver.maximize_window()

try:
    user_name = driver.find_element_by_css_selector("#txtUs
    user_name.send_keys("Admin")

    user_email = driver.find_element_by_css_selector("#txtP
    user_email.send_keys("admin123")

    login_button = driver.find_element_by_css_selector("#bt
    login_button.submit()

    # 截圖存檔
    driver.save_screenshot('screenshot.png')
finally:
    driver.quit()
```

表單操作

text

模擬輸入值到元素中




```
<form action="/action_page.php">
  <label for="username">Username:</label>
  <input type="text" id="username" name="username" /><br />
  <label for="pwd">Password:</label>
  <input type="password" id="pwd" name="pwd" /><br /><br />
  <input type="submit" value="Submit" />
</form>
```

```
form_textfield = driver.find_element_by_name('username')
form_textfield.send_keys("admin")
```

radio

```
<input type="radio" id="male" name="gender" value="male" />
<label for="male">Male</label><br />
<input type="radio" id="female" name="gender" value="female
<label for="female">Female</label><br />
<input type="radio" id="other" name="gender" value="other"
<label for="other">Other</label>
```

```
male_radio = driver.find_element_by_css_selector("[for='ma
male_radio.click()
```

checkbox

```
<input type="checkbox" id="vehicle1" name="vehicle1" value=
<label for="vehicle1"> I have a bike</label><br />
<input type="checkbox" id="vehicle2" name="vehicle2" value=
<label for="vehicle2"> I have a car</label><br />
<input type="checkbox" id="vehicle3" name="vehicle3" value=
<label for="vehicle3"> I have a boat</label><br />
```

```
male_radio = driver.find_element_by_css_selector("[for='veh
male_radio.click()
```

textarea

```
<label for="review">Review:</label>

<textarea id="review" name="review" rows="4" cols="50">
123
</textarea>
```

- https://www.w3schools.com/tags/tag_textarea.asp

```
textarea = driver.find_element_by_name('review')
textarea.send_keys("demo")
```

submit

模擬 submit 行為

```
driver.find_element_by_id("submit").click()
element.submit()
```

上傳檔案

```
picture = driver.find_element_by_css_selector("[type='file
picture.send_keys('/demo/screenshot.png')
```

select

```
<select class="month-select">
  <option value="0">January</option>
  <option value="1">February</option>
  <option value="2">March</option>
  <option value="3">April</option>
  <option value="4">May</option>
  <option value="5">June</option>
  <option value="6">July</option>
  <option value="7">August</option>
  <option value="8">September</option>
  <option value="9">October</option>
  <option value="10">November</option>
  <option value="11">December</option>
</select>
```

```
from selenium.webdriver.support.ui import Select
select_year = Select(driver.find_element_by_css_selector('.
select_year.select_by_value('1')
```

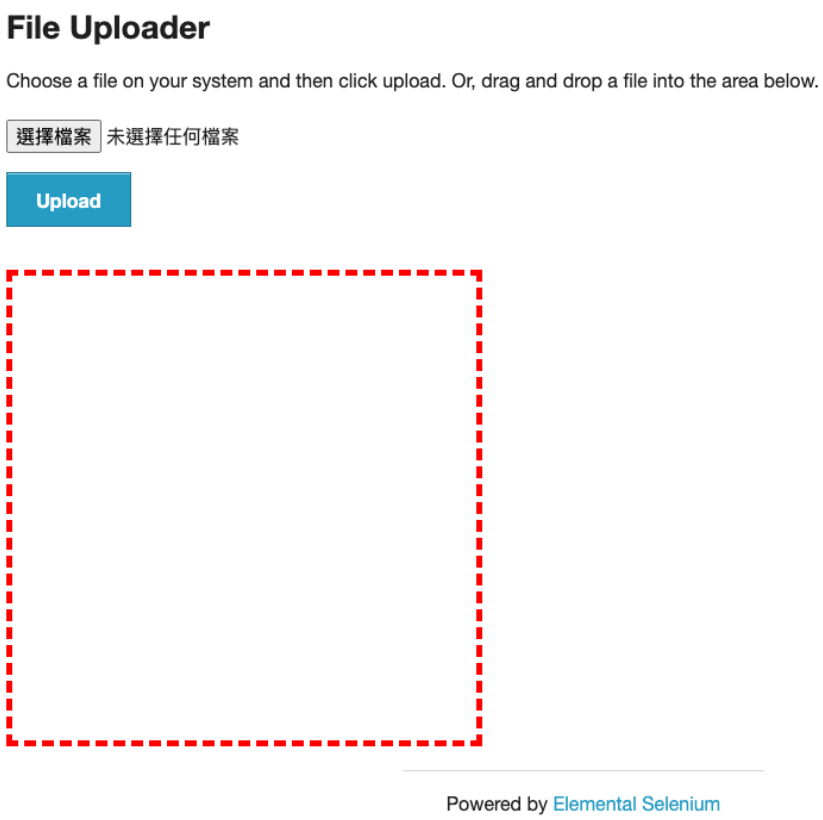
```
select = Select(driver.find_element_by_id('id'))

select.select_by_index(index)
select.select_by_visible_text("text")
select.select_by_value(value)

options = select.options

select.deselect_all()
```

參考文獻

- Selenium File Upload

練習題：上傳檔案

https://the-internet.herokuapp.com/upload



練習目標

- 上傳檔案
- 取得元素文字
- 驗證上傳成功

解答

```
from selenium import webdriver

driver = webdriver.Chrome("./chromedriver")
driver.get("https://the-internet.herokuapp.com/upload")
assert "The Internet" in driver.title

try:

    file = driver.find_element_by_css_selector("[type='file
    file.send_keys('/Users/alin/PycharmProjects/demo/screen
    submit_button = driver.find_element_by_css_selector("#
    submit_button.submit()

    result = driver.find_element_by_css_selector("h3")
    assert "File Uploaded!" in result.text

finally:
    driver.quit()
```

導航 (Navigate to)

一些基本的常用瀏覽器操作

get

模擬瀏覽器載入指定網頁

```
driver.get("https://selenium.dev")
```

```
def get(self, url):
    """
    Loads a web page in the current browser session.
    """
    self.execute(Command.GET, {'url': url})
```

current_url

取得當前的 URL

```
driver.current_url
```

使用情境

- 考慮到驗證 title 會遇到多國語系的問題

back

返回上一頁

```
driver.back()
```

forward

模擬按下瀏覽器下一頁的按鈕

```
driver.forward()
```

refresh

模擬按下瀏覽器重新整理的按鈕

```
driver.refresh()
```

使用情境

- 想要一次重新整理整頁，而不是局部。

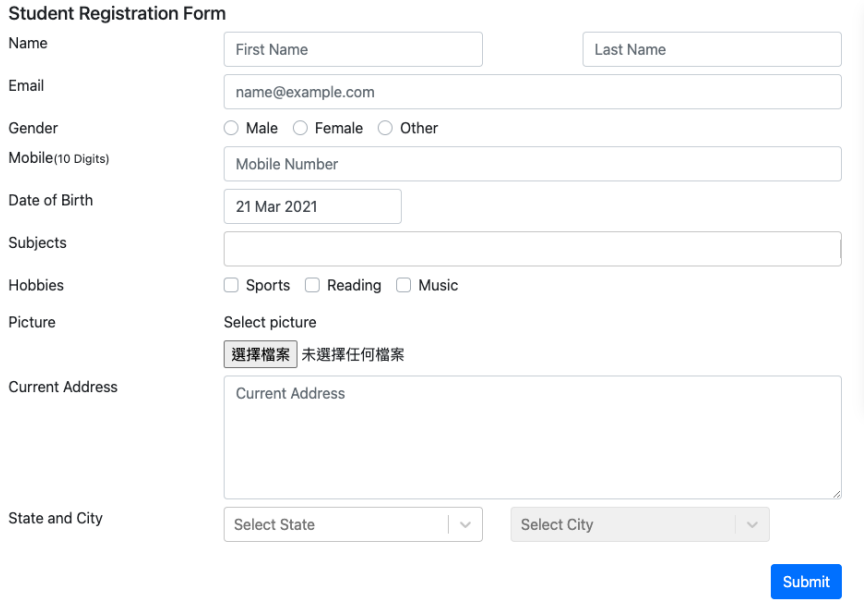
title

取得 title 值

```
driver.title
```

練習題：填寫表單

https://demoqa.com/automation-practice-form



練習目標

- 控制 radio, checkbox 元素
- 控制 modal

解答

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

driver = webdriver.Chrome("./chromedriver")
driver.get("https://demoqa.com/automation-practice-form")
print(driver.title)
assert "ToolsQA" in driver.title

driver.maximize_window()

try:

    first_name = driver.find_element_by_css_selector("#firs
    first_name.send_keys("AILIN")
    last_name = driver.find_element_by_css_selector("#lastN
    last_name.send_keys("LIOU")
    user_email = driver.find_element_by_css_selector("#user
    user_email.send_keys("aaa@aaa.com")
    user_number = driver.find_element_by_css_selector("#use
    user_number.send_keys("0123456789")

    # radio
    male_radio = driver.find_element_by_css_selector("[for=
    male_radio.click()

    # checkbox
    sports_checkbox = driver.find_element_by_css_selector('
    sports_checkbox.click()

    reading_checkbox = driver.find_element_by_css_selector(
    reading_checkbox.click()

    current_address = driver.find_element_by_css_selector('
    current_address.send_keys("TAIPEI")

    # file
    picture = driver.find_element_by_css_selector("[type='f
    picture.send_keys('/Users/alin/PycharmProjects/demo/scr

    submit_button = driver.find_element_by_css_selector("#s
    submit_button.submit()

    # modal
    close_large_modal = driver.find_element_by_css_selector
    close_large_modal.click()
finally:
    driver.quit()
```

元素的狀態與屬性

get_attribute("attr_name")

取得元素指定的屬性值

範例

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

driver = webdriver.Chrome()
driver.get("https://www.google.com")
driver.find_element(By.CSS_SELECTOR, '[name="q"]').send_key

# Get attribute of current active element
attr = driver.switch_to.active_element.get_attribute("title
print(attr)
```

isEnabled()

元素是否是 Enable 的狀態，回傳值為 Boolean。

範例

```
is_enable = driver.find_element_by_css_selector("[type='tex
```

is_selected()

元素是否被選取，回傳值為 Boolean。

範例

```
driver.get("https://the-internet.herokuapp.com/checkboxes")
value = driver.find_element(By.CSS_SELECTOR, "input[type='c
```

tag_name

元素的 tag 名稱

範例

```
attr = driver.find_element(By.CSS_SELECTOR, "h1").tag_name
# 結果回傳：h1
```

rect

取得元素的 height, width, x 和 y 值。

範例

```
rest = driver.find_element(By.CSS_SELECTOR, "h1").rect
```

value_of_css_property('css_attr_name')

取得元素的 CSS 的屬性值

範例

```
cssValue = driver.findElement(By.LINK_TEXT, "More informati
```

text

取得元素的文字

範例

```
text = driver.find_element(By.CSS_SELECTOR, "h1").text
```

補充：判斷元素是否存在

> 注意這裡是 `elements`，如果是呼叫 `element` 會丟出 error。

```
elements = driver.find_elements_by_css_selector("h4")
elements_size = len(elements)

if(elements_size > 0):
  print("元素存在 " + elements.text)
else:
   print("元素不存在")
```

參考文獻

- https://www.selenium.dev/documentation/en/webdriver/web_element/

斷言檢查 (assert)

定義與使用方式

assert 關鍵字可以測試，判斷條件回傳的是否為 `True`，如果回傳 `False` 的話，就會拋出一個 `AssertionError`。

常見使用情境

- 前置條件：在每個容易出錯的行為前面一個步驟做事前檢查，並提供出錯時的友善錯誤提示。
- 後置條件：檢查下一步測試行為的必要條件

語法

```
assert_stmt ::=  "assert" expression ["," expression]
```

語法範例

元素存不存在

元素的數量對不對

檢查數值布林值

```
assert False == True, '不相等'
```

檢查數值條件

```
amount = -1
assert amount > 0, '必須是大於 0 的正數'
```

文字比對

```
assert "your website tile" in driver.title

assert "我是內文" in element.text
```

列表比對

```
class_names = element.get_attribute("class")
assert "active" in class_names.split()
```

- https://mdbootstrap.com/docs/standard/navigation/tabs/

練習題：檢查 Tab 是否可以正常切換

- https://demoqa.com/tabs
- 會用到的語法
  - element.get_attribute("attr_name")
  - string.split()
  - element.text

解答

```
from selenium import webdriver

driver = webdriver.Chrome("./chromedriver")
driver.get("https://demoqa.com/tabs")
assert "ToolsQA" in driver.title

try:
    driver.find_element_by_id("demo-tab-origin").click()
    origin_tab_content = driver.find_element_by_id("demo-ta
    class_names = origin_tab_content.get_attribute("class")
    assert "active" in class_names.split()
    assert "Contrary to popular belief" in origin_tab_conte
finally:
    driver.quit()
```

練習題：檢查欄位是否能被輸入

- https://the-internet.herokuapp.com/dynamic_controls
- 會用到的語法
  - is_enabled()

解答

```
from selenium import webdriver

driver = webdriver.Chrome("./chromedriver")
driver.get("https://the-internet.herokuapp.com/dynamic_cont
assert "The Internet" in driver.title

try:
    is_enable = driver.find_element_by_css_selector("[type=
    print(is_enable)
    assert False == is_enable
finally:
    driver.quit()
```

等待 (Waits)

因為現在大多數網站都使用大量的 Ajax，當頁面載入時，透過非同步動態載入的元素，可能尚未存在而引發 `ElementNotDisableException`，可使用 Waits 語法來解決這個問題。

Selenium Webdriver 提供兩種類型的 Wait：

- 顯示等待
- 隱式等待

模擬情境說明

https://demoqa.com/dynamic-properties

因為第一個按鈕一開始是被 disable，需要等到五秒後，才會被 enable，所以如果我們要點擊第一個按鈕，必需要等到五秒之後。

WebDriverWait 語法

```
WebDriverWait(driver, timeout, poll_frequency=0.5, ignored_
```

- driver：WebDriver 的驅動程式(Ie, Firefox, Chrome 或遠端)
- timeout：最長超時時間，預設以秒為單位。
- poll_frequency：休眠時間的間隔（步長）時間，預設為 0.5 秒。
- ignored_exceptions：超時後的異常資訊，預設情況下拋 NoSuchElementException 異常。

顯式等待 (Explicit Waits)

顯示等待是指明確的等到某個元素的出現，或者是某個元素的可點選等條件符合，等不到就一直等，直到規定的時間之內都沒找到，則跳出 Exception。顯式等待需等特定條件發生時，在執行後面的程序，常與 `until()` 和 `until_not()` 搭配使用。

until / until_not

直到調用的方法返回值為 True

until(method, message='')

method：expected_conditions 庫中定義的方法

範例

```
element = WebDriverWait(driver, 10).until(
    EC.element_to_be_clickable((By.ID, "enableAfter"))
)
```

在上面的範例 Selenium 最多等待 10 秒，若元素則拋出 `TimeoutException` 。在預設情況下，每 500 ms 會檢查一次條件，如果成功 `ExpectedCondition` 會返回 `true` ，找不到元素，則返回 `null` 。

自訂等待條件 (skip)

```
class element_has_css_class(object):
  """An expectation for checking that an element has a part

  locator - used to find the element
  returns the WebElement once it has the particular css cla
  """
  def __init__(self, locator, css_class):
    self.locator = locator
    self.css_class = css_class

  def __call__(self, driver):
    element = driver.find_element(*self.locator)   # Findi
    if self.css_class in element.get_attribute("class"):
        return element
    else:
        return False

# Wait until an element with id='myNewInput' has class 'my(
wait = WebDriverWait(driver, 10)
element = wait.until(element_has_css_class((By.ID, 'myNewI
```

隱式等待 (Implicit Waits)

- 設置全域的元素查找的超時時間
- 隱式等待是代表建立一個最長等待時間，若得不到某個元素就等待，直到拿到元素位置，但如果一直拿不到，就等到時間截止，再執行下一步。
- 隱式等待對整個 driver 週期都起作用，所以只要設定一次即可。

語法

```
# 單位是秒
driver.implicitly_wait(time_to_wait)
```

範例

```
driver.implicitly_wait(5)
element = driver.find_element_by_id("demo")
```

補充：強制等待

```
from time import sleep
sleep(3)
```

使用情境

- 希望可以讓測試慢一點，方便肉眼看到結果。
- 若可以用顯示等待或隱式等待，就不要用 sleep。(如果真的使用需小心謹慎)

參考文獻

- 5. Waits
- Python selenium — 一定要会用 selenium 的等待，三種等待方式解讀
- Web 自动化测试：WebDriverWait 元素等待和全局设置

預期條件 (Expected Conditions)

內建的檢查條件，有以下幾個內建的判斷，除了這些之外，也可以自訂預期條件。

title_is

判斷當前頁面標題是否為 title

```
EC.title_is("Google")
```

title_contains

判斷當前頁面標題是否包含 title

```
EC.title_contains("Google")
```

presence_of_element_located

判斷此定位的元素是否存在

presence_of_all_elements_located(locator)

```
EC.presence_of_element_located((By.ID, "enableAfter"))
```

url_contains

判斷頁面網址中是否包含 url

```
EC.url_contains(url)
```

url_to_be

判斷頁面網址是否為 url

```
EC.url_to_be(url)
```

url_changes(url)

判斷頁面網址不是 url

```
EC.url_changes(url)
```

visibility_of_element_located(locator)

判斷此定位的元素是否可見

visibility_of

判斷此元素是否可見

presence_of_all_elements_located(locator)

判斷此定位的一組元素至少有一個可見

visibility_of_all_elements_located(locator)

判斷此定位的一組元素全部可見

text_to_be_present_in_element(locator, text)

判斷此定位中是否包含 text 的內容

text_to_be_present_in_element_value(locator, text)

判斷此定位中的 value 屬性中是否包含 text 的內容

frame_to_be_available_and_switch_to_it(locator)

判斷定位的元素是否為 frame，並直接切換到這個 frame 中

invisibility_of_element_located(locator)

判斷定位的元素是否不可見

invisibility_of_element(element)

判斷此元素是否不可見

element_to_be_clickable(locator)

判斷所定位的元素是否可見且可點擊

element_to_be_selected(element)

判斷該元素是否被選中

element_located_to_be_selected(locator)

判斷該元素被選中狀態是否和期望狀態相同

element_located_selection_state_to_be(locator,Boolean)

判斷定位的元素被選中狀態是否和期望狀態相同

number_of_windows_to_be(num)

判斷當前瀏覽器頁簽數量是否為 num

new_window_is_opened(handles)

判斷此 handles 頁簽不是唯一打開的頁簽

alert_is_present()

判斷是否會出現 alert 窗口警報

- 官方 API 文件

練習題：非同步行為

```
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions
```

顯式等待

- https://demoqa.com/dynamic-properties
- 會用到的語法
  - WebDriverWait
    - until
  - expected_conditions element_to_be_clickable

解答

```
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions

driver = webdriver.Chrome("./chromedriver")
driver.get("https://demoqa.com/dynamic-properties")
assert "ToolsQA" in driver.title

try:
    element = WebDriverWait(driver, 10).until(
        EC.element_to_be_clickable((By.ID, "enableAfter"))
    )
finally:
    driver.quit()
```

隱式等待

- https://demoqa.com/dynamic-properties
- 會用到的語法
  - implicitly_wait(time_to_wait)

解答

```python
from selenium import webdriver

driver = webdriver.Chrome("./chromedriver")
driver.implicitly_wait(5) # 單位是秒
driver.get("https://demoqa.com/dynamic-properties")
assert "ToolsQA" in driver.title
try:
    element = driver.find_element_by_id("visibleAfter")
    print(element.text)
finally:
    driver.quit()
```

動作鏈 (action chains)

模擬滑鼠操作，例如點擊左鍵、連擊、點擊右鍵、拖曳等等行為。

```
from selenium.webdriver.common.action_chains import ActionC
```

基本用法

有時候我們的行為是有連續性的一連串的動作，希望將命令像鎖鏈一樣排列好，再依照排列的順序依序執行。所以當命令排入 queue 時，並不會立即執行，會等到呼叫 perform() 函式時，才會真的執行。

常見使用情境

- 拖拉 bar
- SortTable
- https://www.draw.io/

使用範例

```
from selenium import webdriver
from selenium.webdriver.common.action_chains import ActionC

driver = webdriver.Chrome("./chromedriver")
driver.get("https://example.com")

menu = driver.find_element(By.CSS_SELECTOR, ".nav")
hidden_submenu = driver.find_element(By.CSS_SELECTOR, ".nav
actions = ActionChains(driver)
actions.move_to_element(menu).click(hidden_submenu).perform
```

連點

```
actions.click(element)

for i for range(10):
  actions.perform()
```

API

click(on_element = None)

點擊一下滑鼠左鍵

> 解說 `on_element = None` 的意思

click_and_hold(on_element = None)

單擊滑鼠左鍵，然後不鬆開。

context_click(on_element = None)

點擊滑鼠右鍵

double_click(on_element = None）

點擊滑鼠左鍵兩下

drag_and_drop(source，target）

拖拽 A 元素到 B 個元素區域，然後鬆開。

drag_and_drop_by_offset(source, xoffset, yoffset)

拖拽元素位移到某個坐標，然後鬆開。

key_down(value, element = None)

點擊某個鍵盤上的鍵

key_up(value, element = None)

鬆開某個鍵

move_by_offset(xoffset, yoffset)

滑鼠從當前位置移動到某個坐標

move_to_element(to_element)

滑鼠移動到某個元素

move_to_element_with_offset（to_element, xoffset, yoffset）

移動到距離某個元素（左上角坐標）多少距離的位置

pause(seconds)

暫停所有輸入至 n 秒

Pause all inputs for the specified duration in seconds

perform()

執行鏈中的所有動作

release(on_element = None)

在某個元素位置鬆開滑鼠左鍵

reset_actions()

清除已經存儲的操作

send_keys(keys_to_send)

發送某個鍵到當前焦點的元素

send_keys_to_element(element, keys_to_send)

發送某個鍵到指定元素

參考文獻

- 7.2. Action Chains
- selenium.webdriver.common.action_chains

練習題：拖拉 Bar

- https://demoqa.com/slider
- 會用到的語法
  - ActionChains
    - click_and_hold
    - move_by_offset
    - release
    - perform

解答

```
from selenium import webdriver
from selenium.webdriver.common.action_chains import Action(

driver = webdriver.Chrome("./chromedriver")
driver.get("https://demoqa.com/slider")
assert "ToolsQA" in driver.title

try:
    input_range = driver.find_element_by_css_selector("[typ
    move = ActionChains(driver)
    move.click_and_hold(input_range).move_by_offset(10, 0).
finally:
    driver.quit()
```

IFrame

- `<iframe>` 標籤用來定義 inline frame，一個 inline frame 嵌入在當前的頁面裡。
- https://www.w3schools.com/tags/tag_iframe.ASP

常見使用情境

- 頁面內含外部套件，如下：
  - 社交外掛程式：
    https://developers.facebook.com/docs/plugins/embedded-posts/
  - editor
  - map

切換 Iframe

語法一：透過 ID

```
<iframe id="ifr" name="demo" src="demo.html" height="200" w
```

```
driver.switch_to_frame("ifr")
```

語法二：透過 Name

```
<iframe id="ifr" name="demo" src="demo.html" height="200" w
```

```
driver.switch_to_frame("demo")
```

語法三：透過 index

```
<iframe id="ifr" name="demo" src="demo.html" height="200" w
<iframe id="ifr" name="demo" class='second' src="width.htm
<iframe id="ifr" name="demo" src="width.html" height="200"
```

```
driver.switch_to_frame(1)
```

語法四：透過元素

```
iframe = driver.find_element_by_css_selector('iframe')
driver.switch_to.frame(iframe)
```

回到預設頁面區塊

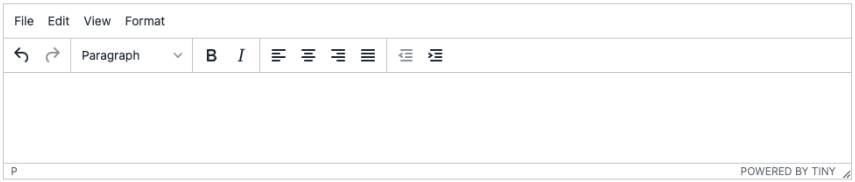
```
driver.switch_to.default_content()
```

延伸閱讀

- Select iframe using Python and Selenium

練習題：TinyMCE

An iFrame containing the TinyMCE WYSIWYG Editor



- http://the-internet.herokuapp.com/iframe
  - driver.switch_to.frame(iframe)
  - driver.switch_to.default_content()

答案

```
from selenium import webdriver
from time import sleep

driver = webdriver.Chrome("./chromedriver")
driver.get("http://the-internet.herokuapp.com/iframe")
assert "The Internet" in driver.title

try:
    iframe = driver.find_element_by_css_selector('iframe')
    driver.switch_to.frame(iframe)
    editor = driver.find_element_by_css_selector('body')
    editor.clear()
    editor.send_keys("123")
    # sleep(5)
    driver.switch_to.default_content()
    title = driver.find_element_by_css_selector('h3')
    assert "An iFrame containing the TinyMCE WYSIWYG Editor
finally:
    driver.quit()
```

訊息框

https://www.selenium.dev/documentation/en/webdriver/js_alerts_prompts_and_confirmations/

使用情境

- 跳離本站警告
- 動作前警告提示

警告訊息框 (Alert)

```
alert = driver.switch_to.alert
alert.text
alert.accept()
expected_conditions.alert_is_present()
```

確認訊息框 (Confirm)

```
alert = driver.switch_to.alert
alert.text
alert.accept()
alert.dismiss()
```

提示訊息對話 (Prompts)

```
alert = driver.switch_to.alert
alert.send_keys("AILIN LIOU")
alert.text
alert.accept()
```

練習題：訊息框

- https://demoqa.com/alerts

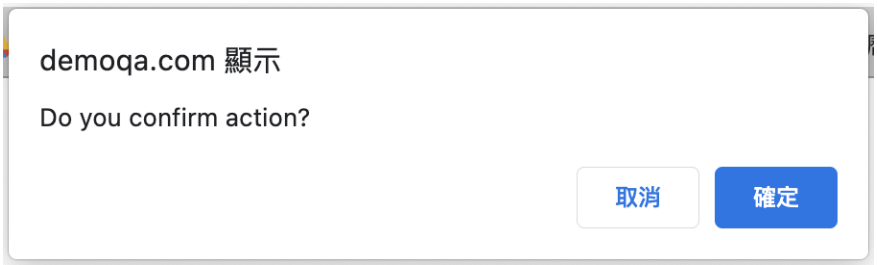
警告訊息框 (alert)

- 會用到的語法
  - switch_to.alert
  - expected_conditions.alert_is_present()
  - alert.text
  - alert.accept()

答案

```
from selenium import webdriver
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions

driver = webdriver.Chrome("./chromedriver")
driver.get("https://demoqa.com/alerts")
assert "ToolsQA" in driver.title

driver.maximize_window()

try:
    driver.find_element_by_id("alertButton").click()
    alert = driver.switch_to.alert
    text = alert.text
    print(text)
    alert.accept()

    driver.find_element_by_id("timerAlertButton").click()
    alert2 = WebDriverWait(driver, 6).until(expected_condi
    text2 = alert.text
    print(text2)
    alert2.accept()

finally:
    driver.quit()
```

確認訊息框 (confirm)



- 會用到的語法
  - driver.switch_to.alert
  - alert.accept()
  - alert.dismiss()

答案

```
from selenium import webdriver

driver = webdriver.Chrome("./chromedriver")
driver.get("https://demoqa.com/alerts")
assert "ToolsQA" in driver.title

try:
    driver.find_element_by_id('confirmButton').click()
    alert = driver.switch_to.alert
    print(alert.text)
    alert.accept()
    # alert.dismiss()
finally:
    driver.quit()
```

提示訊息對話 (prompt)

- 會用到的語法
  - driver.switch_to.alert
  - alert.send_keys("AILIN LIOU")

答案

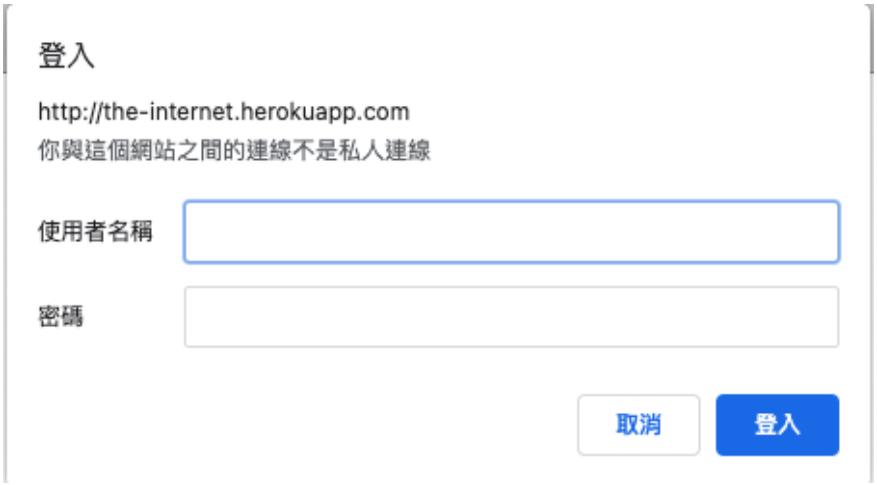
```
from selenium import webdriver

driver = webdriver.Chrome("./chromedriver")
driver.get("https://demoqa.com/alerts")
assert "ToolsQA" in driver.title

try:
    driver.find_element_by_id('promtButton').click()
    alert = driver.switch_to.alert
    alert.send_keys("AILIN LIOU")
    print(alert.text)
    alert.accept()
finally:
    driver.quit()
```

HTTP 基本認證 (Basic Http Authentication)

- 你可以透過將帳號密碼透過 URL 傳遞來完成認證程序

```
driver.get("https://<username>:<password>@www.example.com/
```

練習題



- http://the-internet.herokuapp.com/basic_auth
  - 帳號 admin
  - 密碼 admin

答案

```
from selenium import webdriver

driver = webdriver.Chrome("./chromedriver")
driver.get("http://admin:admin@the-internet.herokuapp.com/b
assert "The Internet" in driver.title

try:
    result = driver.find_element_by_css_selector('h3')
    assert "Basic Auth" in result.text
finally:
    driver.quit()
```

參考文獻

- Basic HTTP Authentication for Selenium tests - BrowserStack: https://www.browserstack.com/docs/automate/selenium/basic-http-authentication

Request

發送一個請求

HTTP Get

```
import requests
response = requests.get('https://api.github.com/events')
```

```
import requests
payload = {'key1': 'value1', 'key2': 'value2'}
>>> r = requests.get('https://httpbin.org/get', params=pay
```

HTTP Post

```
import requests
response = requests.post('https://httpbin.org/post', data =
```

Response

```
response.text
response.status_code
```

JSON Response

```
import requests
response = requests.get('https://api.github.com/events')
response.json()
```

參考文獻

- Quickstart | Python 官方文件

練習題：檢查是否有損壞的圖片

- http://the-internet.herokuapp.com/broken_images
- 會用到的語法
  - current_url()
  - element.get_attribute("attr_name")
  - requests.get(url)
  - response.status_code

解答

```
from selenium import webdriver
import requests
from time import sleep

driver = webdriver.Chrome("./chromedriver")
driver.get("http://the-internet.herokuapp.com/broken_images

try:
    base_url = driver.current_url
    images = driver.find_elements_by_css_selector("img")
    for img in images:
        response = requests.get(img.get_attribute('src'))
        if response.status_code != 200:
            print(img.get_attribute('outerHTML') + " is bro
        else:
            print(img.get_attribute('outerHTML') + " is unb
    sleep(5)
finally:
    driver.quit()
```

測試框架

目前 Python 常見的測試框架主要是 unittest 和 pytest，在此我們使用 unittest 作為範例。

> unittest 又名 PyUnit

unittest 支持物件導向的一些重要的概念，如下所示：

專有名詞	描述
test fixture	它代表跟測試相關所有的資料或程序，例如建立臨時的測試資料庫或建立一些假的測試用資料。
test case	中文常翻譯為「測試案例」，test case 是一個獨立的測試單元。它會針對特定的輸入跟輸出做檢查。unittest 提供叫 `TestCase` 的基礎的類別，可被使用來建立新的 test case。
test suite	中文常翻譯為「測試套件」，一個 test suite 裡可以包含多個 test case，讓這些測試可以一起執行。
test runner	test runner 是一個會自動執行測試案例，並產出測試結果的元件，測試結果可以是用文字呈現，也可以用圖形化呈現。

Assert 語法

Method	Checks that	New in
`assertEqual(a, b)`	`a == b`	
`assertNotEqual(a, b)`	`a != b`	
`assertTrue(x)`	`bool(x) is True`	
`assertFalse(x)`	`bool(x) is False`	
`assertIs(a, b)`	`a is b`	3.1
`assertIsNot(a, b)`	`a is not b`	3.1
`assertIsNone(x)`	`x is None`	3.1
`assertIsNotNone(x)`	`x is not None`	3.1
`assertIn(a, b)`	`a in b`	3.1
`assertNotIn(a, b)`	`a not in b`	3.1
`assertIsInstance(a, b)`	`isinstance(a, b)`	3.2
`assertNotIsInstance(a, b)`	`not isinstance(a, b)`	3.2

參考文獻

- 英文：https://docs.python.org/3/library/unittest.html
- 中文：https://docs.python.org/zh-tw/3/library/unittest.html

完整程式範例

計算模組

```
# calc.py
def add(x, y):
    return x + y

def subtract(x, y):
    return x - y

def multiply(x, y):
    return x * y

def divide(x, y):
    if y == 0:
        raise ValueError('Can not divide by zero!')
    return x / y
```

```
# test_calc.py
import unittest
import calc

class TestCalc(unittest.TestCase):

    def test_add(self):
        self.assertEqual(calc.add(10, 5), 15)
        self.assertEqual(calc.add(-1, 1), 0)
        self.assertEqual(calc.add(-1, -1), -2)

    def test_subtract(self):
        self.assertEqual(calc.subtract(10, 5), 5)
        self.assertEqual(calc.subtract(-1, 1), -2)
        self.assertEqual(calc.subtract(-1, -1), 0)

    def test_multiply(self):
        self.assertEqual(calc.multiply(10, 5), 50)
        self.assertEqual(calc.multiply(-1, 1), -1)
        self.assertEqual(calc.multiply(-1, -1), 1)

    def test_divide(self):
        self.assertEqual(calc.divide(10, 5), 2)
        self.assertEqual(calc.divide(-1, 1), -1)
        self.assertEqual(calc.divide(-1, -1), 1)
        self.assertEqual(calc.divide(5, 2), 2.5)

        with self.assertRaises(ValueError):
            calc.divide(10, 0)


if __name__ == '__main__':
    unittest.main()
```

如果執行 `python test_calc.py`，輸出結果，如下所示：

```
....
----------------------------------------------------------------------
Ran 4 tests in 0.001s

OK
```

說明

測試案例可以透過繼承 `TestCase` 類別來建立，這裡定義了四個獨立的物件方法，名稱皆以 test 開頭，這樣的命名方式能告知 test runner 哪些物件方法為定義的測試。

而每個測試可呼叫 `assertEqual()` 來確認是否為期望的結果，`assertTrue()` 或是 `assertFalse()` 用來驗證一個條件式，`assertRaises()` 則是用來驗證是否觸發一個特定的 exception。使用 assert 相關語法，將能使 test runner 收集所有的測試結果，並產生一個測試報表。

除此之外，透過 `setUp()` 和 `tearDown()` 方法，可以設定測試開始前或結束時要執行的程序。最後可透過執行 `unittest.main()` 方法，來執行測試腳本。

補充

透過 v 參數，你可以得到更多的測試結果細節，例如 `python -m unittest -v test_calc.py`，輸出結果如下：

```
test_add (test_calc.TestCalc) ... ok
test_divide (test_calc.TestCalc) ... ok
test_multiply (test_calc.TestCalc) ... ok
test_subtract (test_calc.TestCalc) ... ok

----------------------------------------------------------------------
Ran 4 tests in 0.000s

OK
```

如果要個別執行測試，則可以使用類似下面的語法。

```
python -m unittest test_module1 test_module2
python -m unittest test_module.TestClass
python -m unittest test_module.TestClass.test_method
```

延伸閱讀

- Python Tutorial: Unit Testing Your Code with the unittest Module - YouTube

練習題：單元測試

- 消費滿一千，打九折。
- 若是 VIP 客戶，在打 95 折。

```
# order.py
max_range = 1000
vip_discount = 0.95
rebate_discount = 0.9

def calculate_order(original_amount, is_vip):
    amount = original_amount
    if original_amount > max_range:
        if is_vip:
            amount = original_amount * rebate_discount * vi
        else:
            amount = original_amount * rebate_discount
    else:
        if is_vip:
            amount = original_amount * vip_discount
    return amount
```

API 測試

API 範例

https://reqres.in/api/users/2


```
{
  "data": {
    "id": 2,
    "email": "janet.weaver@reqres.in",
    "first_name": "Janet",
    "last_name": "Weaver",
    "avatar": "https://reqres.in/img/faces/2-image.jpg"
  },
  "support": {
    "url": "https://reqres.in/#support-heading",
    "text": "To keep ReqRes free, contributions towards ser
  }
}
```


測試應該驗證什麼?

- 結構
- 資料型別
- 值的正確性

完整的測試程式

```
import unittest
import requests

class ApiTest(unittest.TestCase):

    def test_get_user(self):
        url = "https://reqres.in/api/users/2"
        res = requests.get(url)
        self.assertEqual(res.status_code, 200)
        body = res.json()

        for body_key in body.keys():
            self.assertIn(body_key, ['data', 'support'])
        for data_key in body["data"]:
            self.assertIn(data_key, ['id', 'email', 'first_
        for support_key in body["support"]:
            self.assertIn(support_key, ['url', 'text'])

        self.assertIsInstance(body["data"]["id"], int)
        self.assertEqual(body["data"]["id"], 2)

if __name__ == '__main__':
    unittest.main()
```

測試程式邏輯說明

1. 首先繼承 unittest 模組的 TestCase 類別
2. 然後再使用 test 作為前置命名，例如 test_1, test_2。
3. 最後在實作測試細節

延伸閱讀

- https://github.com/public-apis/public-apis
- https://reqres.in/

練習題：API 測試

- 測試 GET `https://reqres.in/api/users?page=1` API

```
{
  "page": 1,
  "per_page": 6,
  "total": 12,
  "total_pages": 2,
  "data": [
    {
      "id": 1,
      "email": "george.bluth@reqres.in",
      "first_name": "George",
      "last_name": "Bluth",
      "avatar": "https://reqres.in/img/faces/1-image.jpg"
    },
    {
      "id": 2,
      "email": "janet.weaver@reqres.in",
      "first_name": "Janet",
      "last_name": "Weaver",
      "avatar": "https://reqres.in/img/faces/2-image.jpg"
    },
    {
      "id": 3,
      "email": "emma.wong@reqres.in",
      "first_name": "Emma",
      "last_name": "Wong",
      "avatar": "https://reqres.in/img/faces/3-image.jpg"
    },
    {
      "id": 4,
      "email": "eve.holt@reqres.in",
      "first_name": "Eve",
      "last_name": "Holt",
      "avatar": "https://reqres.in/img/faces/4-image.jpg"
    },
    {
      "id": 5,
      "email": "charles.morris@reqres.in",
      "first_name": "Charles",
      "last_name": "Morris",
      "avatar": "https://reqres.in/img/faces/5-image.jpg"
    },
    {
      "id": 6,
      "email": "tracey.ramos@reqres.in",
      "first_name": "Tracey",
      "last_name": "Ramos",
      "avatar": "https://reqres.in/img/faces/6-image.jpg"
    }
```

```
  ],
  "support": {
    "url": "https://reqres.in/#support-heading",
    "text": "To keep ReqRes free, contributions towards ser
  }
}
```

- 測試 POST `https://reqres.in/api/login` API

活文件 (living document)

- 所謂的活文件 Living Document 意思是說，文件的會隨著時間、處境而跟著改變的。
- 在軟體上；一份活的文件或動態文件就是一份不斷修改和更新的文件。

Given When Then

每個 scenario 可以分為三個部分：

- Given: 測試的前置條件
- When: 執行某些操作
- Then: 預期的行為

Story	As a customer I need to cancel a window cleaning job
Scenario	Customer contacts Squeeky Kleen two business days before the job's scheduled date
Business rule	A job canceled less than one business day before the Job's scheduled date must be charged a cancellation fee.
Given	
precondition(s)	Job status: scheduled
fixed data	Job number: 1255 Job scheduled date: 24 October 2012
When	
Action	Request to cancel a job
Input data	Request date: 22 October 2012
Then	
output data	Your job has been canceled. Would you like to reschedule for a future date?
postcondition	Job status: canceled

延伸閱讀

- GivenWhenThen | Martin Fowler
- 行为驱动开发：一篇文章带你用 Python 玩转 BDD

- https://www.tenlong.com.tw/products/9789862019481

Specification by Example 中文版：團隊如何交付正確的軟體 (Specification by Example: How Successful Teams Deliver the Right Software)

Gojko Adzic 著、張昌貴、張博超、石永超 譯、傅育文 審校

出版商:	博碩文化
出版日期:	2014-07-29
定價:	$420
售價:	7.8 折 $328
語言:	繁體中文
頁數:	296
ISBN:	9862019484
ISBN-13:	9789862019481
相關分類:	DevOps、Agile Software 敏捷軟體開發
此書翻譯自:	Specification by Example: How Successful Teams Deliver the Right Software (Paperback)

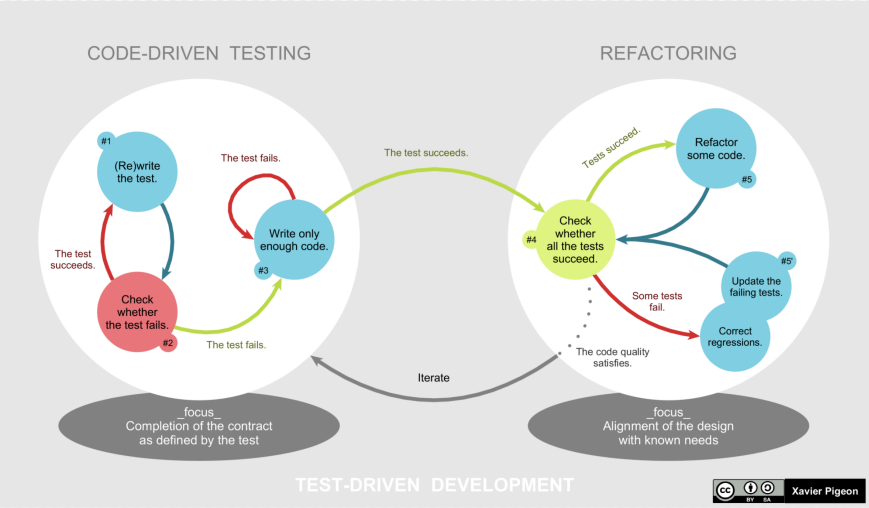
常見開發流程

TDD

- TDD（Test-Driven Development）是一種開發流程，中文是「測試驅動開發」。
- 用一句白話形容，就是「先寫測試再開發」。
- 先寫測試除了能確保測試程式的撰寫，且有助於在開發初期釐清程式介面，如何設計。

具體來說，TDD 流程可以分成五個步驟

1. 選定一個功能，新增測試案例
2. 執行測試，得到 Failed（紅燈）
3. 實作「夠用」的產品程式
4. 再次執行測試，得到 Passed（綠燈）
5. 重構程式



參考文獻

- TDD 開發五步驟，帶你實戰 Test | Alpha Camp

BDD

為什麼需要 BDD？

- TDD 是一種軟體開發流程，先測試再開發。

- TDD 所實作的測試程式碼，可以作為工程師之間討論測試案例或使用情境的基礎
  - 但是對非開發人員很難透過程式碼，去理解測試案例，也更難根據測試案例，進一步討論軟體的功能。

什麼是 BDD？

- TDD 是實作前先寫測試，BDD 比 TDD 更進一步在寫測試前，還要先寫測試規格。
- 這份測試規格會用更接近人類語意的方式描述軟體功能和測試案例
- 這份規格並不是單純的敘述軟體的功能，而是一份「可以被執行的規格」，也就是可以被轉成自動化測試。
- BDD 框架皆支援 Gherkin 語法，這是一種簡單易懂的語言，使用關鍵字來定義系統特徵和測試。

https://behave.readthedocs.io/en/stable/gherkin.html

BDD 常見的熱門框架

- behave: https://behave.readthedocs.io/en/stable/
- cucumber: https://cucumber.io/
- ...

behave

官方文件：https://behave.readthedocs.io/en/stable/

安裝

```
pip install behave
pip install behave2cucumber
```

配置

最低限制

```
features/
features/everything.feature
features/steps/
features/steps/steps.py
```

更複雜的例子

```
features/
features/signup.feature
features/login.feature
features/account_details.feature
features/environment.py
features/steps/
features/steps/website.py
features/steps/utils.py
```

完整範例

feature 範例

`/features/tutorial.feature`

```
Feature: showing off behave

  Scenario: run a simple test
     Given we have behave installed
      When we implement a test
      Then behave will test it for us!
```

step 範例

`features/steps/tutorial.py`

```python
from behave import *

@given('we have behave installed')
def step_given(context):
    pass

@when('we implement a test')
def step_when(context):
    assert True is not False

@then('behave will test it for us!')
def step_then(context):
    assert context.failed is False
```

- context: 可以用來...

```
Feature: Scenario Outline
 Scenario Outline: Blenders
    Given I put <thing> in a blender
    When I switch the blender on
    Then it should transform into <other thing>
  Examples: Amphibians
    | thing         | other thing |
    | Red Tree Frog | mush        |
    | apples        | apple juice |
  Examples: Consumer Electronics
    | thing         | other thing |
    | iPhone        | toxic waste |
    | Galaxy Nexus  | toxic waste |
```

執行

```
behave
behave /features/tutorial.feature
```

```
(venv) alin@liuailinde-MacBook-Pro demo % behave
ConfigError: No steps directory in '/Users/alin/PycharmProjects/demo/features'
(venv) alin@liuailinde-MacBook-Pro demo % cd week5-2
(venv) alin@liuailinde-MacBook-Pro week5-2 % behave
Feature: showing off behave # features/tutorial.feature:1

  Scenario: run a simple test        # features/tutorial.feature:3
    Given we have behave installed   # features/steps/tutorial.py:3 0.000s
    When we implement a test         # features/steps/tutorial.py:7 0.000s
    Then behave will test it for us! # features/steps/tutorial.py:11 0.000s

1 feature passed, 0 failed, 0 skipped
1 scenario passed, 0 failed, 0 skipped
3 steps passed, 0 failed, 0 skipped, 0 undefined
Took 0m0.000s
```

step 參數

```
# feature
When Enter first name "AILIN" and last name "LIOU"
```

```
@when(u'Enter first name "{first_name}" and last name "{las
def step_impl(context, first_name, last_name):
```

Environmental Controls

https://behave.readthedocs.io/en/stable/tutorial.html#environmental-controls

```
# -- FILE: features/environment.py
from behave import fixture, use_fixture
from behave4my_project.fixtures import wsgi_server
from selenium import webdriver

@fixture
def selenium_browser_chrome(context):
    # -- HINT: @behave.fixture is similar to @contextlib.c
    context.browser = webdriver.Chrome()
    yield context.browser
    # -- CLEANUP-FIXTURE PART:
    context.browser.quit()

def before_all(context):
    use_fixture(wsgi_server, context, port=8000)
    use_fixture(selenium_browser_chrome, context)
    # -- HINT: CLEANUP-FIXTURE is performed after after_al

def before_feature(context, feature):
    model.init(environment='test')
```

參考文獻

- Part 1: Selenium with Python Behave (BDD) Introduction - YouTube
- behave.example

練習題： LOGO

Step1

1. 建立 `features/` 和 `features/steps/` 資料夾
2. 建立 feature 檔案，例如 `logo.feature`。
3. 執行 `behave`

```
(venv) alin@liuailinde-MacBook-Pro week5-3 % behave
Feature: OrangeHRM Logo # features/logo.feature:1

  Scenario: Logo presence on OrangeHRM home page  # features/logo.feature:3
    Given launch chrome browser                   # features/steps/logo.py:4 2.208s
    When open orange hrm homepage                 # features/steps/logo.py:9 32.267s
    Then verify that the logo present on page     # features/steps/logo.py:14 0.021s
    And close browser                             # features/steps/logo.py:20 0.102s
```

Step2

1. 建立 step 檔案，例如 `features/steps/logo.py`。
2. 添加 `behave` 和 `selenium` 模組

Step3

1. 實作 `logo.py` 細節
2. 執行 `behave`

練習題：登入

Step1

1. 建立 `features/` 和 `features/steps/` 資料夾
2. 建立 feature 檔案，例如 `login.feature`。
3. 執行 `behave`

```
(venv) alin@liuailinde-MacBook-Pro week5-3 % behave features/login.feature
Feature: OrangeHRM Login # features/login.feature:1

  Scenario: Login to OrangeHRM with valid parameters        # features/login.feature:3
    Given I launch Chrome browser                           # features/steps/login.py:4 2.333s
    When I open orange HRM Homepage                         # features/steps/login.py:9 16.837s
    And Enter username "Admin" and password "admin123"      # features/steps/login.py:14 0.153s
    And Click on login button                               # features/steps/login.py:20 6.353s
    Then User must successfully login to the Dashboard Page # features/steps/login.py:25 0.127s
```

Step2

1. 建立 step 檔案，例如 `features/steps/login.py`。
2. 添加 `behave` 和 `selenium` 模組

Step3

1. 實作 `login.py` 細節
2. 執行 `behave`

POM (Page Object Model)

POM (Page Object Model) 是一種設計模式。

POM 的好處，如下所示：

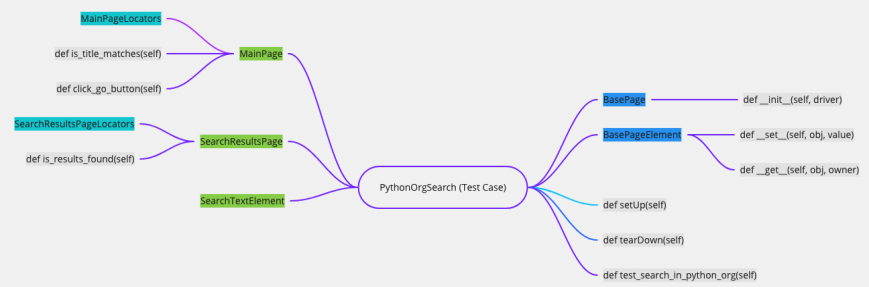
- 易於閱讀的測試案例
- 建立在多個測試案例之間共享，且可重複使用的程式碼。
- 減少重複程式碼的數量
- 如果 UI 發生改變，則僅需在一個位置進行更改程式碼。

> 設計模式（design pattern）：在軟體工程中，設計模式是對軟體設計中普遍存在的各種問題，所提出的解決方案。

設計思維

1. 然後再想測試步驟
2. 然後想要切分成幾個 Page
3. 最後在抽出 element 跟 locators 的細節

測試案例 (Test case)



這是一個測試案例，是執行測試的進入點，它在 Python 官網上搜尋一個關鍵字，並取得搜尋結果。

```
# demo.py
import unittest
from selenium import webdriver
import page

class PythonOrgSearch(unittest.TestCase):

    def setUp(self):
        self.driver = webdriver.Chrome("../chromedriver")
        self.driver.get("http://www.python.org")

    def test_search_in_python_org(self):
        main_page = page.MainPage(self.driver)
        assert main_page.is_title_matches(), "python.org t
        main_page.search_text_element = "pycon"
        main_page.click_go_button()
        search_results_page = page.SearchResultsPage(self.
        assert search_results_page.is_results_found(), "No

    def tearDown(self):
        self.driver.close()

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

頁面物件類別 (Page Object Class)

```
# page.py
from element import BasePageElement
from locators import MainPageLocators

class SearchTextElement(BasePageElement):
    locator = 'q'

class BasePage(object):
    def __init__(self, driver):
        self.driver = driver


class MainPage(BasePage):
    search_text_element = SearchTextElement()

    def is_title_matches(self):
        return "Python" in self.driver.title

    def click_go_button(self):
        element = self.driver.find_element(*MainPageLocato
        element.click()


class SearchResultsPage(BasePage):
    def is_results_found(self):
        return "No results found." not in self.driver.page_
```

頁面元素

```
# element.py
from selenium.webdriver.support.ui import WebDriverWait


class BasePageElement(object):

    def __set__(self, obj, value):
        driver = obj.driver
        WebDriverWait(driver, 100).until(
            lambda driver: driver.find_element_by_name(self
        driver.find_element_by_name(self.locator).clear()
        driver.find_element_by_name(self.locator).send_keys

    def __get__(self, obj, owner):
        driver = obj.driver
        WebDriverWait(driver, 100).until(
            lambda driver: driver.find_element_by_name(self
        element = driver.find_element_by_name(self.locator)
        return element.get_attribute("value")
```

定位 (Locators)

```
# locators.py
from selenium.webdriver.common.by import By

class MainPageLocators(object):
    GO_BUTTON = (By.ID, 'submit')

class SearchResultsPageLocators(object):
    pass
```

參考文獻

- 6. Page Objects

POM 練習題

Cheat Sheet

- Python: https://www.pythoncheatsheet.org/
- Pip: https://opensource.com/sites/default/files/gated-content/cheat_sheet_pip.pdf

```
pip list
```

測試相關資源

可供練習的測試網站

- http://demoqa.com/
- https://example.testproject.io/web/
- http://www.globalsqa.com/demo-site/
- https://opensource-demo.orangehrmlive.com/
- http://the-internet.herokuapp.com/
- http://sahitest.com/demo/
- http://www.way2automation.com/demo.html
- http://juliemr.github.io/protractor-demo/
- http://automationpractice.com/

其他

- Docker images for the Selenium Grid Server: https://github.com/SeleniumHQ/docker-selenium
- christian-bromann / awesome-selenium: https://github.com/christian-bromann/awesome-selenium