# 79 Adding verifications

## 79.1 About

If we are writing tests in Selenium, there will usually be some point in the script where we want to verify that something is correct in the application under test- for example, a field containing the correct value (such the output of a calculation or a total amount of a list of transactions), the correct number of items being displayed in a list, or a piece of text appearing in the correct place.

By itself, Selenium does not have any mechanisms for deciding if a test is passing or failing. But, we can use the features of the Python language itself, or the test framework being used, to write pass/fail decisions into our tests. Usually this will be a place in the script where we compare the **actual** value of an item against an **expected** value.

In this activity you will do the following tasks:

- Create a project
- Create a script
- Add a verification point using a conditional statement
- Add a verification point using a Pytest assertion

## 79.2 Before you start

Have the editor open but close any scripts which you may have open.

## 79.3 Create a script

Create a new project called **seleniumVerificationsProject** or similar.
Add the necessary require statement for Selenium:

```
from selenium import webdriver
```

Add the code to initialise the browser and go to the home page of the website.

```
wd = webdriver.Chrome()
wd.get("http://<AUT IP>/index.html")
```

Normally we would probably also add a line at the end of the test to close the browser (**wd.quit()**) but in this case it will be useful to keep the browser open to see how the code is working.

Run the script and make sure everything is working OK up to this point:

## 79.4 Add a verification point using a conditional statement

Users who are visually impaired frequently use tools such as screen readers to help them navigate websites. When a screen reader encounters a picture, it uses the HTML attributes of the picture, usually the **alt** tag, to help it decide how to read the information to the user. If the alt tag is empty, the screen reader skips the picture, which means the user might lose some information. Web accessibility guidelines state that every picture (except those used purely for decoration, such as page borders) should have an alt tag, like this:

```
<img border="0" src="img/11.jpg" title="pic title" alt=”alt text">
```

Note that the value of the **title** attribute appears as a popup when you move the mouse over an image.

On the home page there is a link to a page that has some text and a picture. We will write a test that checks if the picture has an alt tag.
First add a line that navigates to the page:

```
wd.find_element(By.LINK_TEXT, "Page with a bit of text and a little picture").click()
```

Now find the image on the page. In this case there is only one picture on the page. As often happens, there is more than one way to identify the picture. In this example, we'll use the CSS selector, like we saw earlier:

```
pic = wd.find_element(By.CSS_SELECTOR, "html > body > img")
```

Use an **if** statement and the **get_attribute** method to check the alt tag and output a message depending on the result.

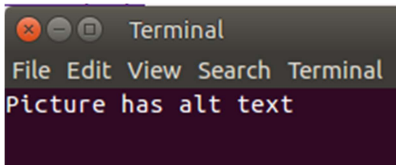
```
if(pic.get_attribute("alt") != ""):
    print("Picture has alt text")
else:
    print("Picture has no alt text")
```

## 79.5 Checkpoint

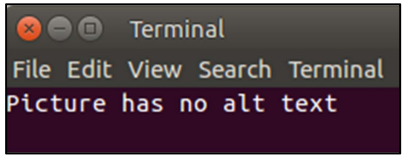
At this stage the code should look similar to this:

```
from selenium import webdriver

wd = webdriver.Chrome()
wd.get("http://<AUT IP>/index.html")
wd.find_element(By.LINK_TEXT, "Page with a bit of text and a little picture").click()

pic = wd.find_element(By.CSS_SELECTOR, "html > body > img")

if(pic.get_attribute("alt") != ""):
    print("Picture has alt text")
else:
    print("Picture has no alt text")
```

Save and run and check you get the expected output. The picture does indeed have an alt tag, so the output should be positive.




We should also check the other branch works successfully as well. There is another page with just a picture with no alt tag- change the code to navigate to that page: **http://<AUT IP>/picture.html**. Re-run the test to make sure you get the expected output.




Change the script back before continuing.

## 79.6 Challenge

All the thumbnails on the page with several big picture files (linked from the home page) should also have alt tags. Modify the code so that it navigates to that page and checks each picture in turn.

## 79.7 Convert the script to a Pytest test

In place of conditional statements, we can use pytest to add testing features to Selenium scripts.
Convert the script to a pytest test by doing the following:

- Save the script with a new name: **test_selenium-verifications.py**
- Put the steps of the test into a method by adding the **def** statement and indenting the script appropriately
- Replace the **if** statement with an **assert** statement

Like this:

```
from selenium import webdriver

def test_alt():
    wd = webdriver.Chrome()
    wd.get("http://<AUT IP>/index.html")
    wd.find_element(By.LINK_TEXT, "Page with a bit of text and a little picture").click()

    pic = wd.find_element(By.CSS_SELECTOR, "html > body > img")
    assert pic.get_attribute("alt") != ""
```

Save the script and run it by opening a terminal window and using the command **pytest -v test_selenium-verifications.py**. The output should look similar to this:

```
ubuntu@ubuntu-VirtualBox:~/python-workspace$ pytest -v test_selenium-verificatio
ns.py
============================= test session starts ==============================
platform linux -- Python 3.5.2, pytest-3.4.1, py-1.5.2, pluggy-0.6.0 -- /usr/bin
/python3
cachedir: .pytest_cache
metadata: {'Python': '3.5.2', 'Plugins': {'metadata': '1.6.0', 'html': '1.16.1'}
, 'Platform': 'Linux-4.4.0-116-generic-x86_64-with-Ubuntu-16.04-xenial', 'Packag
es': {'pytest': '3.4.1', 'pluggy': '0.6.0', 'py': '1.5.2'}}
rootdir: /home/ubuntu/python-workspace, inifile:
plugins: metadata-1.6.0, html-1.16.1
collected 1 item

test_selenium-verifications.py::test_alt PASSED                          [100%]

=========================== 1 passed in 7.42 seconds ===========================
```

Check what happens if the assert fails. Change the script to point to **picture.html** as described in the previous section then run the test again. Observe the output:

```
ubuntu@ubuntu-VirtualBox:~/python-workspace$ pytest -v test_selenium-verificatio
ns.py
============================= test session starts ==============================
platform linux -- Python 3.5.2, pytest-3.4.1, py-1.5.2, pluggy-0.6.0 -- /usr/bin
/python3
cachedir: .pytest_cache
metadata: {'Packages': {'pluggy': '0.6.0', 'py': '1.5.2', 'pytest': '3.4.1'}, 'P
ython': '3.5.2', 'Plugins': {'metadata': '1.6.0', 'html': '1.16.1'}, 'Platform':
 'Linux-4.4.0-116-generic-x86_64-with-Ubuntu-16.04-xenial'}
rootdir: /home/ubuntu/python-workspace, inifile:
plugins: metadata-1.6.0, html-1.16.1
collected 1 item

test_selenium-verifications.py::test_alt FAILED                          [100%]

=================================== FAILURES ===================================
_________________________________ test_alt _________________________________

    def test_alt():
        wd = webdriver.Firefox()
        wd.get("http://localhost/index.html")
        wd.find_element_by_link_text("Page with a bit of text and a little pictu
re").click()

        pic = wd.find_element_by_css_selector("html > body > img")
>       assert pic.get_attribute("alt") != ""
E    assert '' != ''
E     +  where '' = <bound method WebElement.get_attribute of <selenium.webdrive
r.firefox.webelement.FirefoxWebElement (session="2066d8d3-5958-4d98-a463-f74839f
42062", element="71b8a466-0fcd-4f39-9ff4-a4b29435944e")>>('alt')
E     +    where <bound method WebElement.get_attribute of <selenium.webdriver.f
irefox.webelement.FirefoxWebElement (session="2066d8d3-5958-4d98-a463-f74839f420
62", element="71b8a466-0fcd-4f39-9ff4-a4b29435944e")>> = <selenium.webdriver.fir
efox.webelement.FirefoxWebElement (session="2066d8d3-5958-4d98-a463-f74839f42062
", element="71b8a466-0fcd-4f39-9ff4-a4b29435944e")>.get_attribute

test_selenium-verifications.py:10: AssertionError
========================== 1 failed in 8.14 seconds ===========================
```

Change the page back before continuing.

## 79.8 Challenge

Write a script that inserts a new comment into the demo website , then returns to the home page, then verifies that the top entry in the comment wall is the comment that was entered. Use pytest asserts to do the verification.

What would be the potential problem with this challenge, if we were running parallel or distributed text execution? How could we overcome the problem?