1. Create a test script "CreateAndDeletePage" in the IDE which creates an article "MyArticle" with some body text, and then delete it. (To make the script robust, you should ensure you always start by entering an URL in the web browser, so that the starting point is well defined.)

2. Replay this script several times, and check everything goes well.

You can see in the Selenium IDE window that Selenium creates the script as a list of commands, each command consisting of a command name, and target and an optional value. A full reference of all available commands in available in Selenium manual.

3. Edit the script "CreateAndDeletePage" in order to change the name of the article and the body text.

4. Create a new script "CreatePage" in the IDE which just creates an article "MySecondArticle".

5. Replay this script several times. It should fail at the second command. Why?

6. Delete this second script.

Element Locators.
Selenium IDE has several mechanisms to identify elements on a web page, see the manual.

7. Edit the script "CreateAndDeletePage" to use a CSS locator instead of a name one.

Some commands in Selenium IDE allows to make additional checks while the script is executed.

8. The command assertTitle checks the current value of the page title. Extends the script "CreateAndDeletePage" to check the title page of the article once created. Try this with the correct page title and also with a wrong one.

9. (optional) The command assertText checks the current contents of some HTML element in the page. Extends the script "CreateAndDeletePage" to check the article body of the article once created. Try this with the correct page title and also with a wrong one.

10. (optional) The command assertValue checks the current value of some HTML form element in the page. Extends the script "CreateAndDeletePage" to check the default contents of the article when creating it.

Selenium IDE also allows to store some values in variable and to re-use them later in the script.

11. (optional) Create a script "CopyPage" which copies the contents of the article "MyArticle" into a new article "MyCopiedArticle".

Part 2: Selenium WebDriver
In this second part, we will use Selenium WebDriver to write test scripts in Python. The Python API for Selenium WebDriver is included in the archived you downloaded at the beginning of this practical. You can find an intro to this API and its full documentation online.

1. Create a first script that just opens a browser on the main page of the Wiki. Your script should look like:

```python
#!/usr/bin/python
import sys
```
This command is needed because Selenium is not installed in the main Python path. Update the path to point to the right directory in your home directory.
sys.path.append('/home/vtst/lecture7-practical/selenium-2.43.0/py')

from selenium import webdriver

The proxy settings may have to be adapted to your network settings.
PROXY_HOST_PORT = 'proxy:3128'

proxy = webdriver.common.proxy.Proxy(
    'proxyType': webdriver.common.proxy.ProxyType.MANUAL,
    'httpProxy': PROXY_HOST_PORT,
    'ftpProxy': PROXY_HOST_PORT,
    'sslProxy': PROXY_HOST_PORT,
    'noProxy': 'localhost'
)

browser = webdriver.Firefox(proxy=proxy)
browser.get('http://localhost:8000/cgi-bin/monkeywiki.py')

Make your script executable (chmod u+x script.py) and execute it (./script.py).

2. When using Selenium for testing in Python, it's convenient to use the unittest module from the Python Standard Library:

```python
import unittest
class MyTestCase(unittest.TestCase):
    def setUp(self):
        self.browser = webdriver.Firefox(proxy=proxy)
        self.addCleanup(self.browser.quit)

    def testSuccess(self):
        self.browser.get('http://localhost:8000/cgi-bin/monkeywiki.py'

    def testFailure(self):
        self.browser.get('http://localhost:8000/cgi-bin/monkeywiki.py')

if __name__ == '__main__':
    unittest.main(verbosity=2)
```

The most important class of the API is WebElement. An instance of WebElement represents an HTML Element on a page. You can get a WebElement by using the find_* methods you can see in the documentation. The obtained object has several methods which allow to perform actions like entering text or clicking.

3. Write a test script which creates an article in the Wiki, and then deletes it.
4. Write a test script which creates 10 articles in the Wiki.