

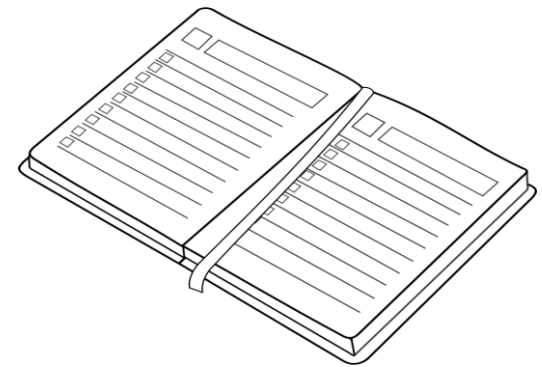


Test and Behavior Driven Development with Python

Lorena Lobato Pardavila, IT-CM-LCS
2nd Developers@CERN Forum
CERN, 31st May 2016

What I am going to talk?

- What do you know about testing?
- What is Test-Driven Development?
- Is Behaviour-Driven Development the same?
- Unit Testing tools: Behave and Selenium Web Driver
- What do we take from this talk?



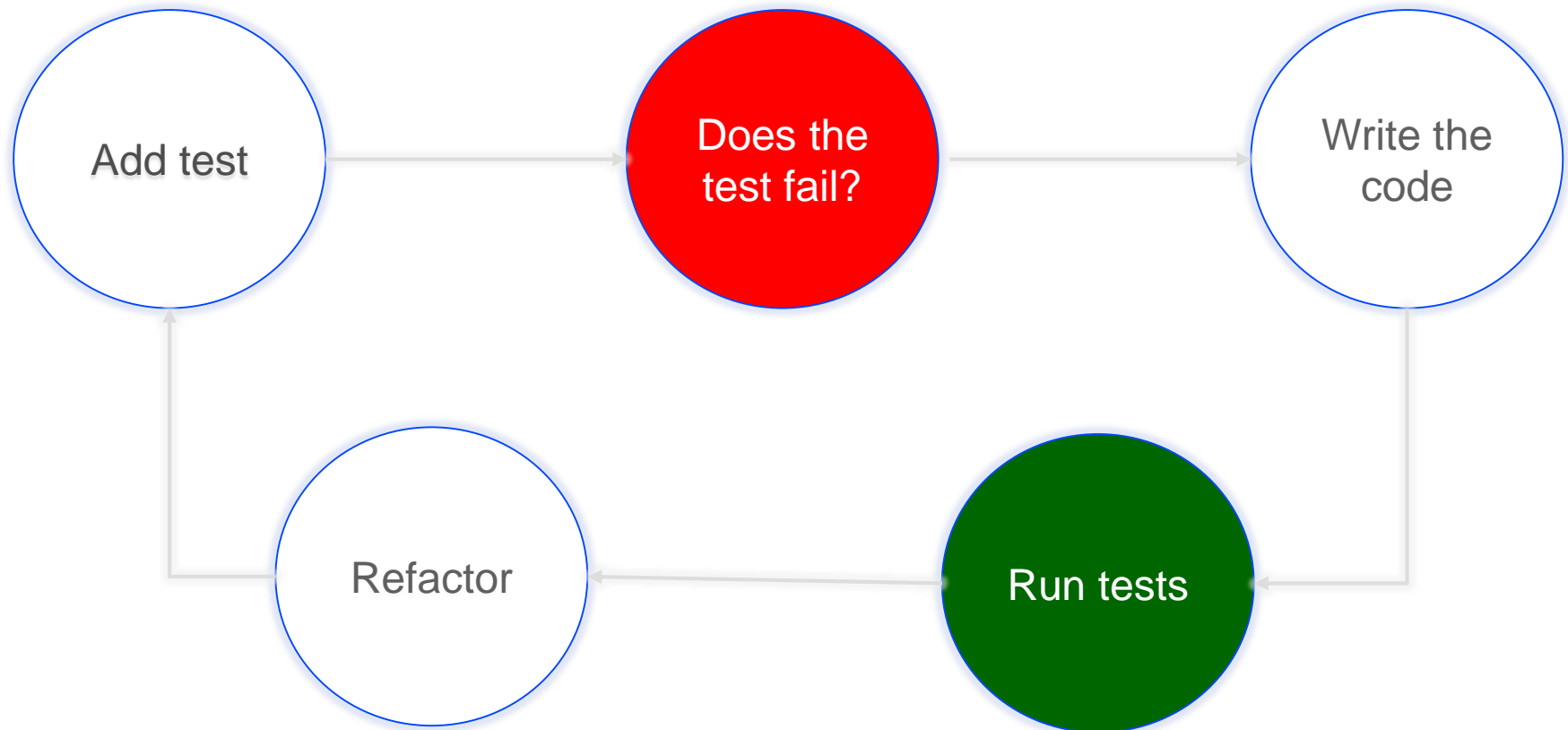
What do you know about Testing?

White-Box Testing	Focus on the activities . You know the source of the program
Black-Box Testing	Focus in the result . You have no idea how the program should work
Unittest	Single piece of code to be tested
Integration testing	Multiple pieces are tested together
Acceptance Testing	Automatic testing of the entire application
TDD vs BDD	Agile Software Development techniques

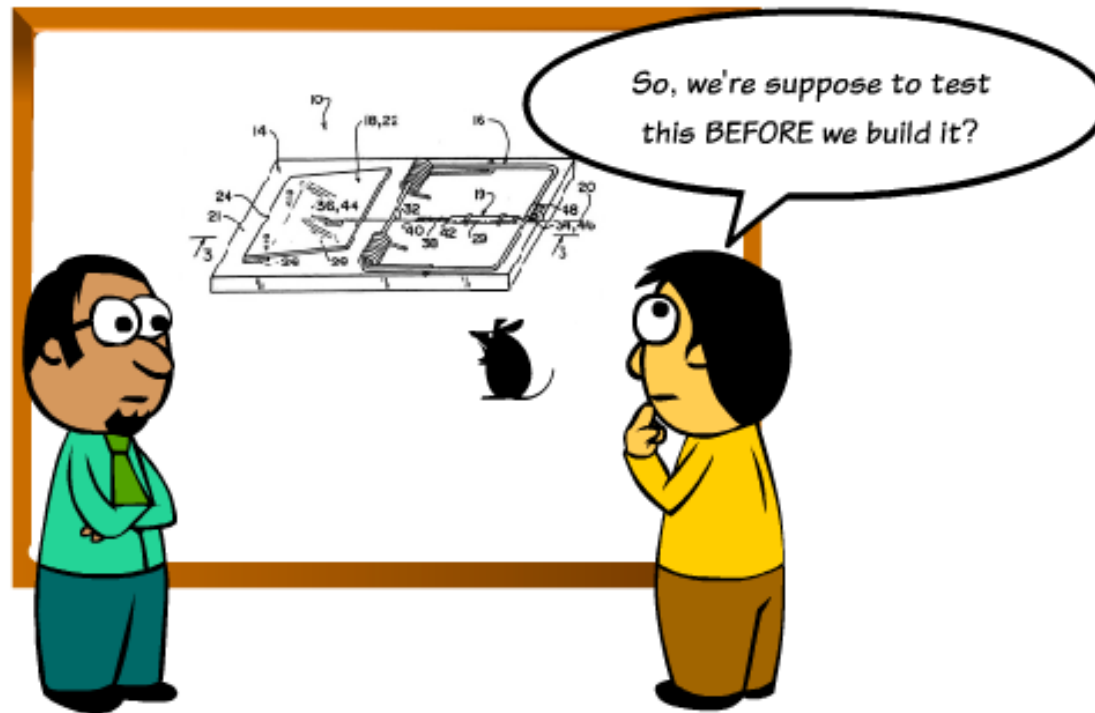
What is Test-Driven Development?

TDD
ALL CODE IS GUILTY
UNTIL PROVEN INNOCENT

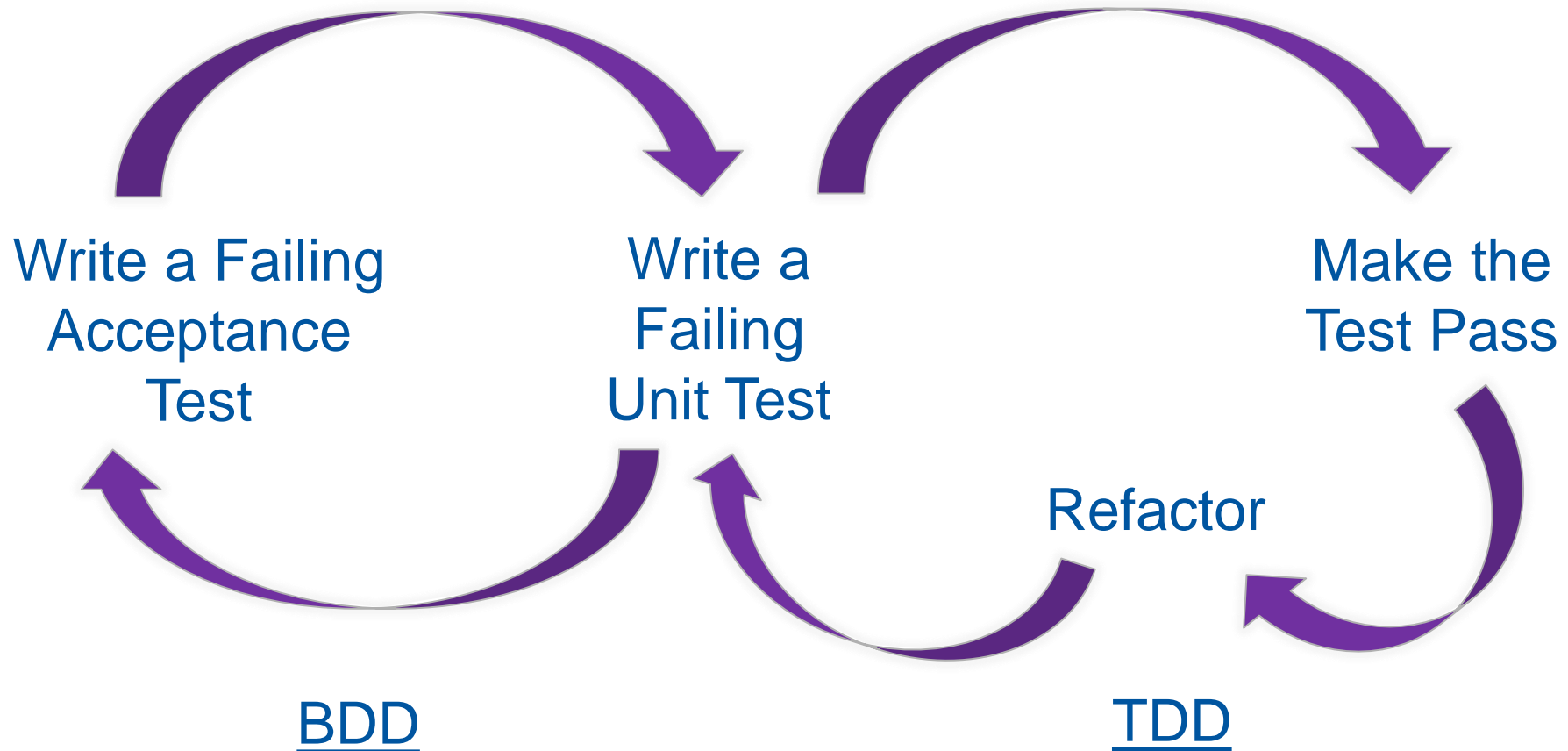
Test-Driven Development (TDD)



Is Behaviour-Driven Development the same?



Behaviour-Driven Development (BDD)



Behaviour-Driven Development (BDD)



Feature: Title (one line describing the story/feature)

```
As a [role]
I want [feature]
So that [benefit]
```

Scenario: Title1 (for Behaviour 1)

```
Given [context or setup]
And [some more context]...
When [event occurs]
Then [expected outcome]
And [another outcome]...
```

Scenario: Title2 (for Behaviour 2)

...

UNIT TESTING TOOLS



Behave



- ❖ BDD framework based on Python implementation of the Cucumber specification
- ❖ *Behave* uses tests written in a natural language style, backed up by Python code.
- ❖ BDD concepts: When, Given, Then

```
$PROJECT/  
+-- features/                -- Contains all feature files.  
|   +-- steps/  
|   |   +-- step_*.py        -- Step definitions for features.  
|   +-- environment.py       -- Environment for global setup...  
|   +-- tutorial*.feature     -- Feature files.
```

Behave: Environment Controls



```
import threading
from wsgiref import simple_server
from selenium import webdriver
from my_application import model
from my_application import web_app

def before_all(context):
    context.server = simple_server.WSGIServer('', 8000)
    context.server.set_app(web_app.main(environment='test'))
    context.thread = threading.Thread(target=context.server.serve_forever)
    context.thread.start()
    context.browser = webdriver.Chrome()

def after_all(context):
    context.server.shutdown()
    context.thread.join()
    context.browser.quit()

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

Behave: Write the feature test



```
# file:features/tutorial02_natural_language.feature
Feature: Fight or Flight (Natural Language, tutorial02)
```

```
In order to increase the ninja survival rate,
As a ninja commander
I want my ninjas to decide whether to take on an opponent
based on their skill levels.
```

```
Scenario: Weaker opponent
```

```
  Given the ninja has a third level black-belt
  When attacked by a samurai
  Then the ninja should engage the opponent
```

```
Scenario: Stronger opponent
```

```
  Given the ninja has a third level black-belt
  When attacked by Chuck Norris
  Then the ninja should run for his life
```

Behave: Provide the Test Automation



```
# file:features/steps/step_tutorial02.py
# -----
# STEPS:
# -----

from behave import given, when, then
from hamcrest import assert_that, equal_to, is_not

@given('the ninja has a {achievement_level}')
def step_the_ninja_has_a(context, achievement_level):
    context.ninja_fight = NinjaFight(achievement_level)

@when('attacked by a {opponent_role}')
def step_attacked_by_a(context, opponent_role):
    context.ninja_fight.opponent = opponent_role

@when('attacked by {opponent}')
def step_attacked_by(context, opponent):
    context.ninja_fight.opponent = opponent

@then('the ninja should {reaction}')
def step_the_ninja_should(context, reaction):
    assert_that(reaction, equal_to(context.ninja_fight.decision()))
```

Behave: Run your test



```
$ behave ../features/tutorial02_natural_language.feature
Feature: Fight or Flight (Natural Language, tutorial02) # ../features/tutorial02_natural_language.feature:1
  In order to increase the ninja survival rate,
  As a ninja commander
  I want my ninjas to decide whether to take on an opponent
  based on their skill levels.

Scenario: Weaker opponent # ../features/tutorial02_natural_language.feature:8
  Given the ninja has a third level black-belt # ../features/steps/step_tutorial02.py:57
  When attacked by a samurai # ../features/steps/step_tutorial02.py:61
  Then the ninja should engage the opponent # ../features/steps/step_tutorial02.py:69

Scenario: Stronger opponent # ../features/tutorial02_natural_language.feature:13
  Given the ninja has a third level black-belt # ../features/steps/step_tutorial02.py:57
  When attacked by Chuck Norris # ../features/steps/step_tutorial02.py:65
  Then the ninja should run for his life # ../features/steps/step_tutorial02.py:69

1 feature passed, 0 failed, 0 skipped
2 scenarios passed, 0 failed, 0 skipped
6 steps passed, 0 failed, 0 skipped, 0 undefined
Took 0m0.001s
```

Behave example: DNS LB

Feature: Testing API calls for Ermis

Scenario: test that ermis is up by performing a get request

Given that we have a valid kerberos ticket of a user in "ermis-lbaas-admins" egroup

when we do a "get" request

then we get a "200" back

```
@when('we do a "{req}" request')
```

```
@then('we get a "{n}" back')
```

```
def step_impl(context,n):
```

```
    if n == "200":
```

```
        assert context.response.status_code == 200,"code %d not expected" % context.response.status_code
```

```
    elif n == "202":
```

```
        assert context.response.status_code == 202,"code %d not expected" % context.response.status_code
```

```
    elif n == "400":
```

```
        assert context.response.status_code == 400,"code %d not expected" % context.response.status_code
```

```
    elif n == "401":
```

```
        assert context.response.status_code == 401,"code %d not expected" % context.response.status_code
```

```
    elif n == "409":
```

```
        assert context.response.status_code == 409,"code %d not expected" % context.response.status_code
```

```
    elif n == "400 or 401":
```

```
        assert (context.response.status_code == 401 or context.response.status_code == 400) #case of unprivileged user
```

```
    else:
```

```
        assert False
```

```
        context.response = requests.patch(url+str(alias_id)+'/', data=json.dumps(payload), headers=headers, auth=
```

```
    except Exception as e:
```

```
        print(str(e))
```

```
        assert False
```

Behave example: DNS LB



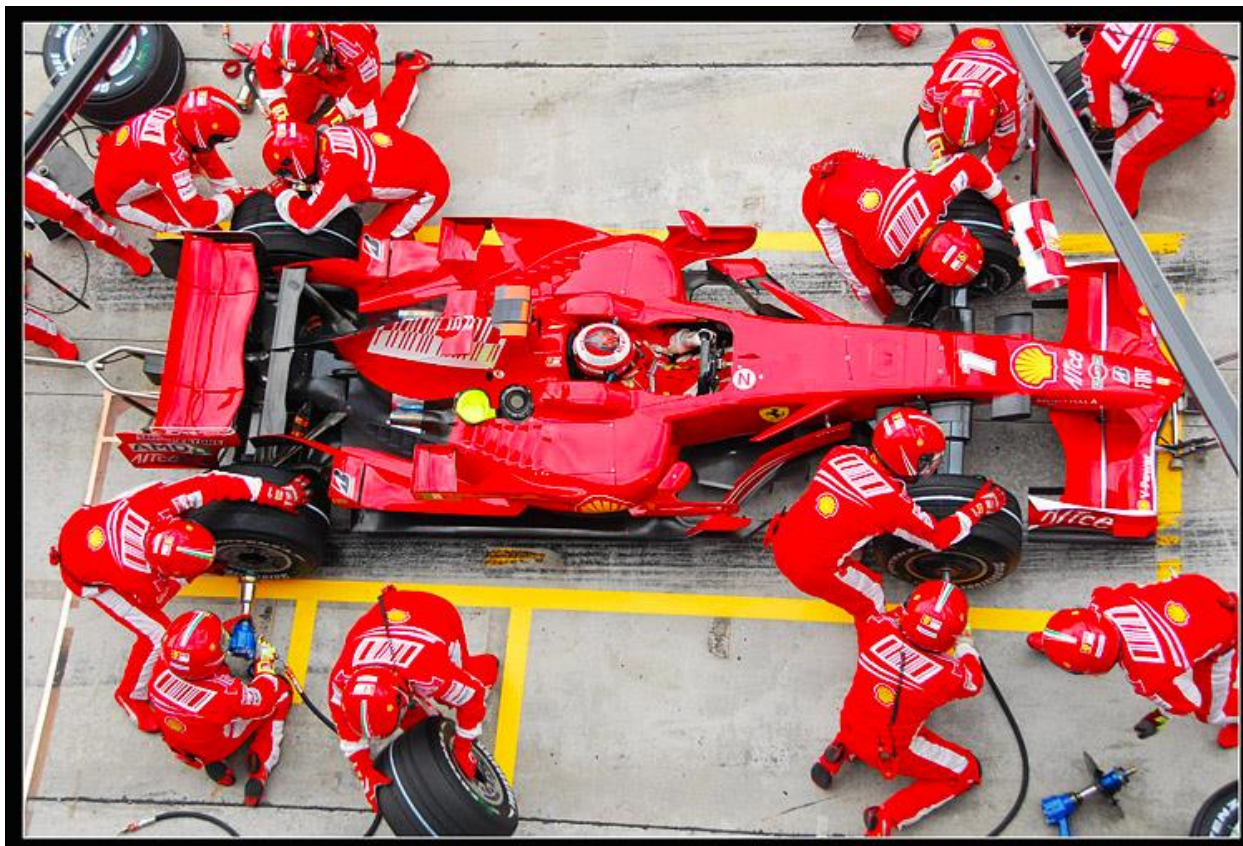
```
ermis_tests.feature: steps
[[llobato@aiadm060 ermis_tests(loreana-test)]$ behave
Feature: Testing API calls for Ermis # ermis_tests.feature:1

Scenario: test that ermis is up by performing a get request # ermis_tests.feature:3
  Given that we have a valid kerberos ticket of a user in "ermis-lbaas-admins" egroup # steps/ermis_tests.py:22 1.280s
  When we do a "get" request # steps/ermis_tests.py:66 0.997s
  Then we get a "200" back # steps/ermis_tests.py:117 0.000s

Scenario: test if unprivileged user can access data even if he/she has a valid kerberos ticket (works as we currently allow R0 access to anyone) # ermis_tests.feature:8
  Given that we have a valid kerberos ticket of a user in "other" egroup # steps/ermis_tests.py:22 0.892s
  When we do a "get" request # steps/ermis_tests.py:66 0.951s
  Then we get a "200" back # steps/ermis_tests.py:117 0.000s

Scenario: test if unprivileged user can create data if he/she has a valid kerberos ticket # ermis_tests.feature:13
  Given that we have a valid kerberos ticket of a user in "other" egroup # steps/ermis_tests.py:22 1.085s
  Then we get a "401" back # steps/ermis_tests.py:117 0.000s
  Then we get a "401" back # steps/ermis_tests.py:117
  Then the object should "be created" # steps/ermis_tests.py:134 0.001s
  Then the object should "be created" # steps/ermis_tests.py:134
  And the object should "be updated" # steps/ermis_tests.py:134 0.946s
  And the object should "be updated" # steps/ermis_tests.py:134
  Then the object should "be deleted" # steps/ermis_tests.py:134 0.000s
  Then the object should "be deleted" # steps/ermis_tests.py:134
  Then the object should "be created" # steps/ermis_tests.py:134 0.000s
  Then the object should "be created" # steps/ermis_tests.py:134
  Then we get a "400" back # steps/ermis_tests.py:117 0.000s
  Then we get a "400" back # steps/ermis_tests.py:117
  Then we get a "401" back # steps/ermis_tests.py:117 0.000s
  Then we get a "401" back # steps/ermis_tests.py:117
  Given that we have a valid kerberos ticket of a user in "ermis-lbaas-admins" egroup # steps/ermis_tests.py:22 1.611
  And that we are "unauthorized" in the hostgroup # steps/ermis_tests.py:36 0.001
  And the LB alias "exists" # steps/ermis_tests.py:51 0.933 Wh
```


What if...?



Selenium Web Driver



- Open source, web-based testing automation tool
- It aims to mimic the behaviour of a real user, and as such interacts with the HTML of the application.
- Multi-language backend support (Java, Ruby, Python, C#, PHP...)
- Supports Cross-Browser Testing. The tests can be run on multiple browsers.



Selenium Web Driver



```
from selenium import webdriver
from selenium.common.exceptions import TimeoutException
from selenium.webdriver.support.ui import WebDriverWait # available since 2.4.0
from selenium.webdriver.support import expected_conditions as EC # available since 2.26.0

# Create a new instance of the Firefox driver
driver = webdriver.Firefox()

# go to the google home page
driver.get("http://www.google.com")

# the page is ajaxy so the title is originally this:
print driver.title

# find the element that's name attribute is q (the google search box)
inputElement = driver.find_element_by_name("q")

# type in the search
inputElement.send_keys("cheese!")

# submit the form (although google automatically searches now without submitting)
inputElement.submit()

try:
    # we have to wait for the page to refresh, the last thing that seems to be updated is the title
    WebDriverWait(driver, 10).until(EC.title_contains("cheese!"))

    # You should see "cheese! - Google Search"
    print driver.title

finally:
    driver.quit()
```

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

```
class Newvisitor(unittest.TestCase):
```

```
def setUp(self):
    self.browser = webdriver.Chrome()
    self.browser.get('https://aiermis.cern.ch')
```

```
def tearDown(self):
    self.browser.close()
    pass
```

```
def test_sso_login(self):
    self.browser.get('https://aiermis.cern.ch')
```

```
link = self.browser.find_element_by_id('id_new_alias')
```

```
link = self.browser.find_element_by_id('id_new_alias')
```

```
inputbox = self.browser.find_element_by_id('id_new_alias')
inputbox.send_keys('hola-developers!')
inputbox.send_keys(Keys.ENTER)
```

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

DNS Load Balancing Service

Logged in as llobato

[CERN SSO Logout](#)

Main menu

- [LBWeb](#)
 - [Add LB Alias](#)
 - [Modify LB Alias](#)
 - [Display LB Alias](#)
 - [Delete LB Alias](#)
 - [LB Alias Logs](#)

Add an LB Alias


Alias Name (unqualified)*

 Not RFC compliant! 

External *

Yes

No

This cluster will be available only inside the CERN network 

Alias Members

Only Puppet managed nodes can be used as alias members.

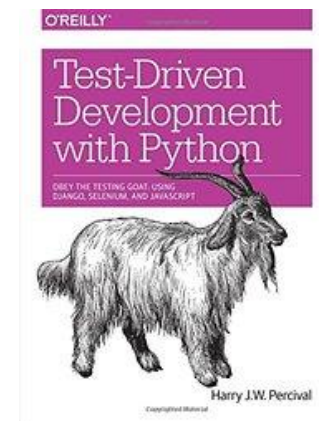
Alias members have to be defined following the documentation in the following URL:

<https://configdocs.web.cern.ch/configdocs/dnslb/aliasmembers.html>

From this talk..

- Unit Testing gives you the *what*. Test-Driven Development gives you the *when*. Behavior Driven-Development gives you the *how*.
- Run tests often by integrating these tools with **CI systems**.
- *Behave* is great to have the code documented and to know in advance the requirements
- *Selenium Web Driver* for “lazy people” 😊

I like it! Where can I start from?



Unit testing Python framework	https://docs.python.org/2.7/library/unittest.html
General rules Python testing	http://docs.python-guide.org/en/latest/writing/tests/
Behave	http://pythonhosted.org/behave/
Selenium Web Driver	http://selenium-python.readthedocs.io/
Test-Driven Development with Python	http://chimera.labs.oreilly.com/books/1234000000754/

Questions?

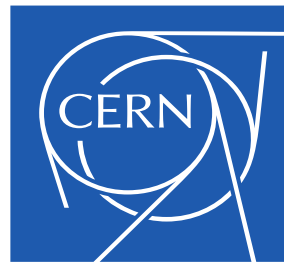


THANKS FOR YOUR ATTENTION!

CONTACT

Work: lorena.lobato@cern.ch

Twitter: [@lobatopardavila](https://twitter.com/lobatopardavila)



www.cern.ch