Web Automation for Testing, Time-Saving, and Profit

By Michael Mintz

About Me

* I like to automate things.
* I’ve built automation for HubSpot, Veracode, iboss, and others.
* I’ve automated testing, website migrations, customer support, data extraction, and manual labor.

What is Selenium?

* A browser automation library for interacting with web applications
* Also known as: "Selenium-WebDriver"

What Selenium is NOT

* A standalone test framework

(Selenium by itself is not a standalone test framework, meaning that it requires additional work to be useful as a complete tool for end-to-end testing of web applications.)

Why is browser-testing useful?

* Unit tests have limited coverage.
* Different web browsers can display the same HTML differently.
* Browser tests can interact with apps in the same way that customers do.

Common web automation issues

* Can be slow
* Can be flaky/unreliable
* Tricky to write/maintain scripts
* Tricky to read others' scripts
* Time-consuming setup, etc.

Issue: Can be slow

* Hard-coded waiting commands such as “time.sleep()” waste time. (developers use that often to prevent flaky tests inefficiently)

Issue: Can be flaky

* There can be unexpected behavior when interacting with page objects that haven’t finished loading.

Issue: Tricky to read/write scripts

* Long lines of code are common:

driver.find_element_by_css_selector("textarea").send_keys("text")

(This is a standard command from pure Selenium WebDriver)

* This is better:

self.type("textarea", "text")

(This is a SeleniumBase command, which includes smart-waiting.)

Issue: Time-consuming setup

* Without a prebuilt e2e test framework, it takes extra time to add code for:
  * Test management
  * Browser management
  * Logging and report-generation
  * Dashboards, charts, and screenshots
  * CI setup, DB setup, etc…

Improving on Selenium

* SeleniumBase

An open-source Python framework that makes it easier to write reliable browser automation for testing and more.

(Includes test management, browser management, reports, charts, dashboards, and screenshots.)

seleniumbase.io / SeleniumBase on GitHub

SeleniumBase

* Easy Setup (takes < 3 minutes)
* Reliable
* Lots of functionality
* Easy to write scripts quickly
* Built on top of Selenium-WebDriver
* Extends the pytest unit-testing framework

Easy Setup

* > pip install seleniumbase
* > seleniumbase install chromedriver

(Make sure that Chrome is already installed)

Try an example test

* > git clone https://github.com/seleniumbase/SeleniumBase.git
* > cd SeleniumBase/examples
* > pytest my_first_test.py --browser=chrome

(Chrome is the default browser if not set)

Reliable methods

* SeleniumBase methods wait for page objects to fully load before interacting with them. This prevents flaky tests.

Built-in Functionality

* The pytest plugin ecosystem
* Headless browser automation
* User-agent, proxy, and mobile control
* Logging and report-generation
* Dashboards, charts, and screenshots
* Tools for building website components

Command-line control

* Choose a web browser to use
* Choose Demo Mode (can change speed)
* Choose a proxy server to connect to
* Choose a MySQL DB to send results to
* Choose a Selenium Grid to connect to
* And more. (These are all optional settings)

Configure default settings

* Make changes to “settings.py”
  (located in SeleniumBase/seleniumbase/config/)
* > python setup.py install
  (That makes your changes take effect when not using a developer-mode install)
* It’s easier to keep using the default settings.

Sample script (easy to write)

my_first_test.py

```python
from seleniumbase import BaseCase


class MyTestClass(BaseCase):

    def test_basic(self):
        self.open("https://store.xkcd.com/search")
        self.type('input[name="q"]', "xkcd book")
        self.click('input[value="Search"]')
        self.assert_text("xkcd: volume 0", "h3")
        self.open("https://xkcd.com/353/")
        self.assert_title("xkcd: Python")
        self.assert_element('img[alt="Python"]')
        self.click('a[rel="license"]')
        self.assert_text("free to copy and reuse")
        self.go_back()
        self.click_link_text("About")
        self.assert_exact_text("xkcd.com", "h2")
        self.click_link_text("geohashing")
        self.assert_element("#comic img")

```

Written in Python

* If you know Python, you can write automation with SeleniumBase.
* If you don’t know Python, it’s very easy to learn the basics you need.

Run tests with pytest or nosetests

* pytest my_first_test.py --browser=chrome
* nosetests my_first_test.py --browser=firefox

(Using pytest is recommended)

Built on Selenium-WebDriver

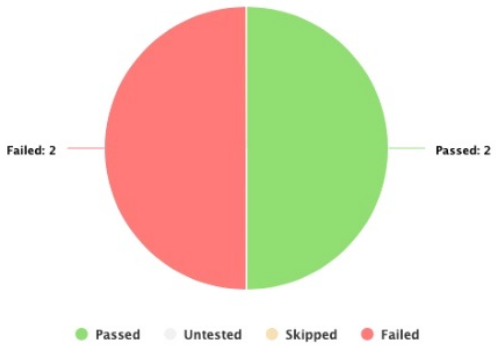
* You can run any WebDriver method you want by typing:

self.driver.{WEBDRIVER_METHOD}

The Dashboard

* Status chart
* Test results
* Stack traces
* Screenshots
* Links to logs

SeleniumBase Test Results Dashboard



Report generated on 21-Dec-2020 at 12:06:04 by pytest-html v2.0.1

Summary

4 tests ran in 18.87 seconds.

(Un)check the boxes to filter the results.

2 passed, 0 skipped, 2 failed, 0 errors, 0 expected failures, 0 unexpected passes, 0 rerun

Results

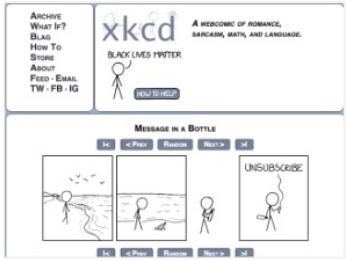
Status: Test Run Complete: **2 tests failed!** (See latest logs for details)

Result	Test	Duration	Links
Failed *(hide details)*	examples/test_suite.py::MyTestSuite::test_2	4.10	URL Screenshot

```
self = <examples.test_suite.MyTestSuite testMethod=test_2>

    @pytest.mark.expected_failure
    def test_2(self):
        print("\n(This test fails on purpose)")
        self.open("https://xkcd.com/1675/")
>       raise Exception("FAKE EXCEPTION: This test fails on purpose.")
E       Exception: FAKE EXCEPTION: This test fails on purpose.

test_suite.py:20: Exception
```

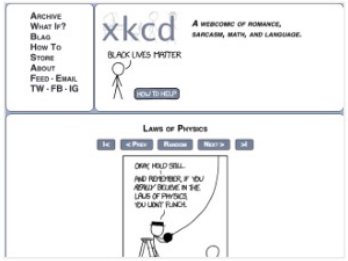


Result	Test	Duration	Links
Failed *(hide details)*	examples/test_suite.py::MyTestSuite::test_4	4.68	URL Screenshot

```
exception = <class 'selenium.common.exceptions.NoSuchElementException'>
message = '\n Element {FakeElement.DoesNotExist} was not present after 0.5
seconds!'

    def timeout_exception(exception, message):
        exc, message = s_utils.format_exc(exception, message)
>       raise exc(message)
E       selenium.common.exceptions.NoSuchElementException: Message:
E        Element {FakeElement.DoesNotExist} was not present after 0.5 seconds!

../seleniumbase/fixtures/page_actions.py:117: NoSuchElementException
```



Result	Test	Duration	Links
Passed *(show details)*	examples/test_suite.py::MyTestSuite::test_1	5.59	
Passed *(show details)*	examples/test_suite.py::MyTestSuite::test_3	3.71	

Runs in multiple environments

* OS X
* Windows
* Linux
* Docker

SeleniumBase Linux example

```
Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
       ___ _ _                    _
      | _ |_) |_ _ _   __ _ _ __ (_)
      | _ \ |  _| ' \/ _` | '  \| |
      |___/_|\__|_|_|\__,_|_|_|_|_|

  *** Welcome to the Bitnami Jenkins 1.644-1         ***
  *** Bitnami Wiki:   https://wiki.bitnami.com/      ***
  *** Bitnami Forums: https://community.bitnami.com/ ***
mdmintz@jenkins-7:~$ ls
apps  htdocs  qqq.sh  SeleniumBase  selenium-server.jar  stack  www.sh
mdmintz@jenkins-7:~$ cd SeleniumBase/
mdmintz@jenkins-7:~/SeleniumBase$ ls
build        dist    Dockerfile        examples    LICENSE    requirements.txt  seleniumbase.egg-info    setup.cfg
conftest.py  docker  Docker_README.md  grid_files  README.md  seleniumbase      server_requirements.txt  setup.py
mdmintz@jenkins-7:~/SeleniumBase$ py.test examples/my_first_test.py --with-selenium --headless
============================================= test session starts ==============================================
platform linux2 -- Python 2.7.3, pytest-2.8.5, py-1.4.31, pluggy-0.3.1
rootdir: /home/mdmintz/SeleniumBase, inifile:
collected 1 items

examples/my_first_test.py .

========================================== 1 passed in 9.66 seconds ============================================
mdmintz@jenkins-7:~/SeleniumBase$
```

SeleniumBase Docker example

```
Installed /usr/local/lib/python2.7/dist-packages/seleniumbase-1.1.23-py2.7.egg
Processing dependencies for seleniumbase==1.1.23
Finished processing dependencies for seleniumbase==1.1.23
 ---> 80b6861d9aa9
Removing intermediate container d08ee43edd67
Step 28 : COPY integrations/docker/docker-entrypoint.sh /
 ---> 0ec4c9d04fe0
Removing intermediate container a0445980cb8f
Step 29 : COPY integrations/docker/run_docker_test_in_firefox.sh /
 ---> c3712bdf8dcc
Removing intermediate container 1bdb8e1e106a
Step 30 : COPY integrations/docker/run_docker_test_in_chrome.sh /
 ---> dfb57940ff87
Removing intermediate container ef68d02bb69b
Step 31 : COPY integrations/docker/docker_config.cfg /SeleniumBase/examples/
 ---> 1d4ad4b59696
Removing intermediate container 159d380523d4
Step 32 : ENTRYPOINT /docker-entrypoint.sh
 ---> Running in 89bacc46243e
 ---> 15c1a7f9940c
Removing intermediate container 89bacc46243e
Step 33 : CMD /bin/bash
 ---> Running in e783085582c3
 ---> 216acd9b8fe3
Removing intermediate container e783085582c3
Successfully built 216acd9b8fe3
DrSeleniums-MacBook-Pro:SeleniumBase michael$
```

Support for 10 spoken languages

* Japanese example

```
# Japanese Language Test
from seleniumbase.translate.japanese import セレニウムテストケース


class 私のテストクラス(セレニウムテストケース):

    def test_例1(self):
        self.を開く("https://ja.wikipedia.org/wiki/")
        self.テキストを確認する("ウィキペディア")
        self.要素を確認する('[title="メインページに移動する"]')
        self.入力("#searchInput", "アニメ")
        self.クリックして("#searchButton")
        self.テキストを確認する("アニメ", "#firstHeading")
        self.入力("#searchInput", "寿司")
        self.クリックして("#searchButton")
        self.テキストを確認する("寿司", "#firstHeading")
        self.要素を確認する('img[alt="握り寿司"]')
        self.入力("#searchInput", "レゴランド・ジャパン")
        self.クリックして("#searchButton")
        self.要素を確認する('img[alt="Legoland japan.jpg"]')
        self.リンクテキストを確認する("名古屋城")
        self.リンクテキストをクリックします("テーマパーク")
        self.テキストを確認する("テーマパーク", "#firstHeading")
```

Learn More

seleniumbase.io

SeleniumBase on GitHub

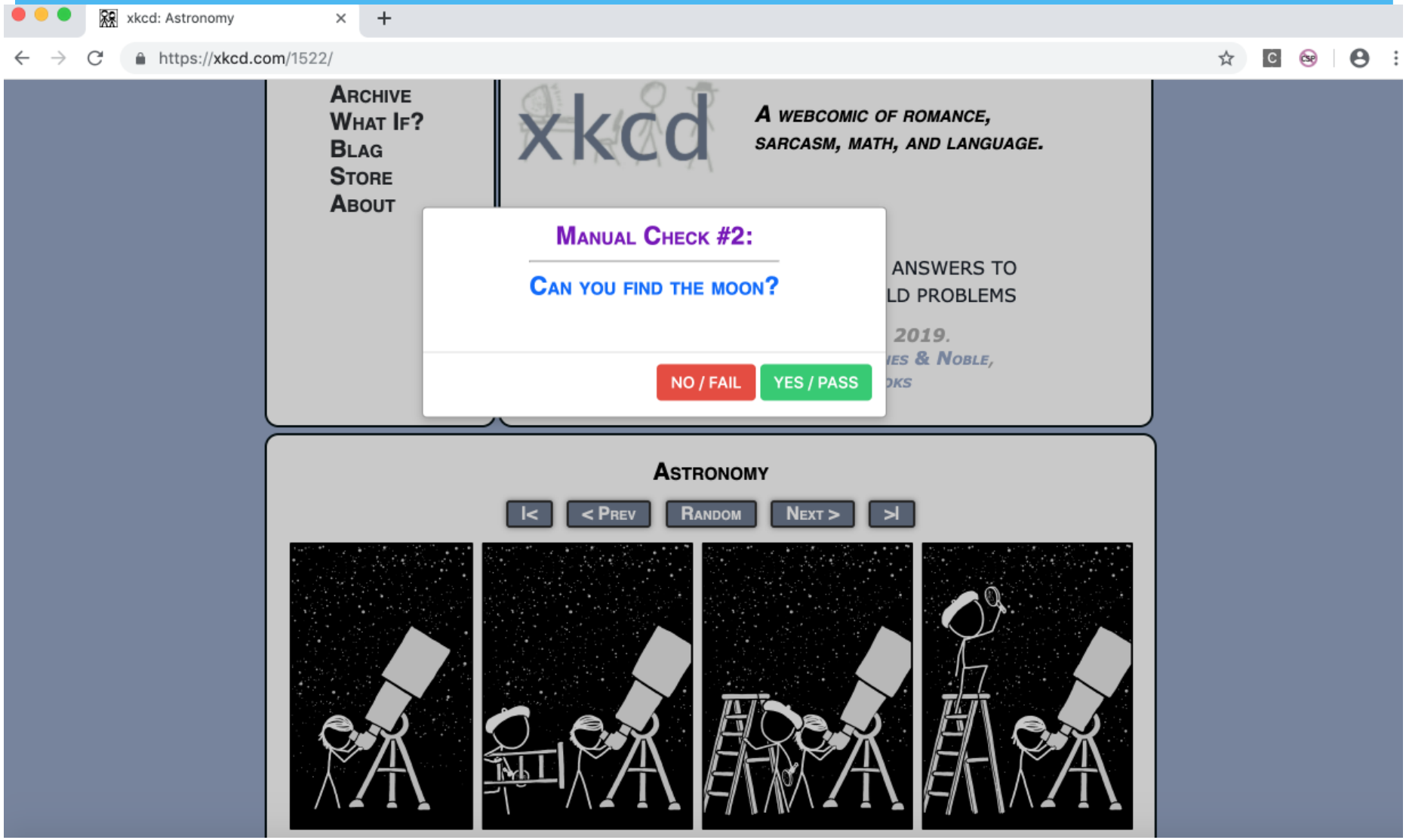
There’s also MasterQA

Automation-Powered Acceptance Testing

Built on top of SeleniumBase

100% Open Source

MasterQA - example run



MasterQA - example test

verify_test.py

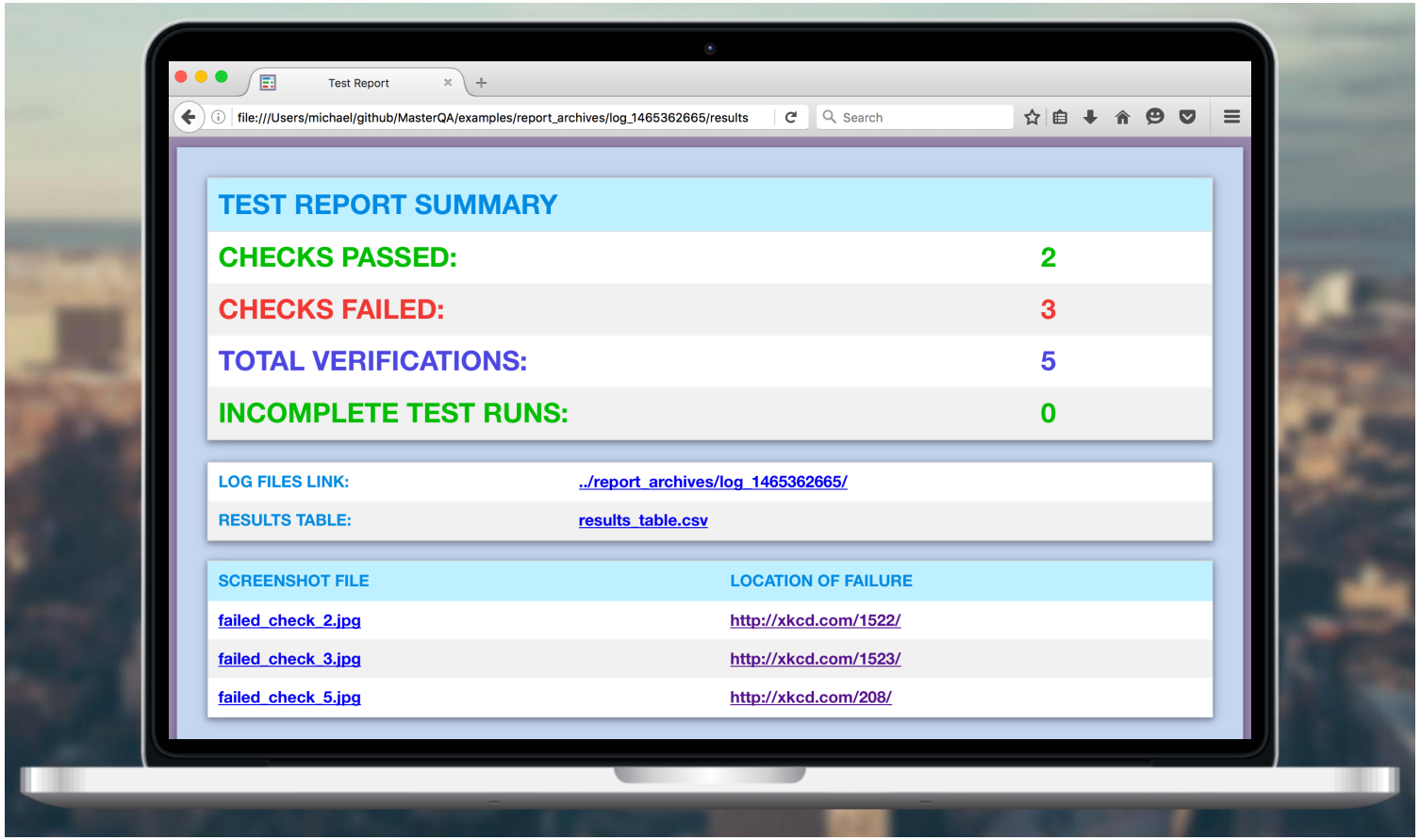
```python
from masterqa import MasterQA


class MasterQATests(MasterQA):

    def test_xkcd(self):
        self.open("http://xkcd.com/1512/")
        self.click('a[rel="next"]')
        self.verify()
        self.open("http://xkcd.com/1520/")
        for i in range(2):
            self.click('a[rel="next"]')
        self.verify("Can you find the moon?")
        self.click('a[rel="next"]')
        self.verify("Do the drones look safe?")
        self.click_link_text('Blag')
        self.update_text("input#s", "Robots!\n")
        self.verify("Does it say 'Hooray robots' on the page?")
        self.open("http://xkcd.com/213/")
        for i in range(5):
            self.click('a[rel="prev"]')
        self.verify("Does the page say 'Abnormal Expressions'?")

```

MasterQA - results page



LIVE DEMO TIME

* Get ready…

The robots are coming

* Robots will steal jobs
* Automation will steal jobs
* The future is all about automation
* Learn to automate, or risk getting automated
* Start learning automation today…

Conclusion

The End

> Questions?

> Twitter: @mintzworld