

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?

- * Browser automation framework for testing web applications
- * Also known as:
“Selenium WebDriver”

Why have browser automation?

- *Unit tests have limited coverage.
- *Web browsers can display the same data differently.

Common issues with automation

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

Issue: Can be slow

- *`time.sleep()` wastes time
(developers use that often)

Issue: Can be flakey

- *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 better:

```
self.update_text("textarea", "text")
```


Issue: Time-consuming setup

- * Takes extra time to add code for:
 - * Writing to a database
 - * Saving screenshots & data automatically on failures
 - * Etc...

Improving on Selenium

*SeleniumBase

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

<http://seleniumbase.com>

SeleniumBase

- * Easy Setup (takes < 3 minutes)
- * Reliable
- * Lots of functionality
- * Easy to write scripts quickly
- * Built on top of Selenium / WebDriver

Easy Setup

*> `pip install seleniumbase`

*> `seleniumbase install chromedriver`

Try an example test

- *> git clone <https://github.com/seleniumbase/SeleniumBase.git>
- *> cd SeleniumBase/examples
- *> pytest my_first_test.py --browser=chrome
(Have Chrome and ChromeDriver installed)

Reliable methods

- *Methods wait for page objects to load before acting on them.
- *click()
- *update_text()
- *Etc...

Built-in Functionality

- * All available Pytest plugins
- * Headless browser automation
- * User-Agent and Proxy control
- * Automated hybrid mode - “MasterQA”
- * Screenshots and advanced log files
- * Website Tour Builder

Command-line control

- * Choose a web browser
- * Demo Mode option (change speed)
- * Choose a proxy server to connect to
- * Choose to log data to a MySQL DB
- * Choose a Selenium Grid to use
- * And more...

How to Configure

- *Make changes to “settings.py”

(located in SeleniumBase/seleniumbase/config/)

- *> python setup.py install

(That deploys your changes)

Sample script (easy to write)

```
my_first_test.py x
1  from seleniumbase import BaseCase
2
3
4  class MyTestClass(BaseCase):
5
6      def test_basic(self):
7          self.open("https://xkcd.com/353/")          # Navigate browser to page
8          self.assert_element('img[alt="Python"]')    # Assert element on page
9          self.click('a[rel="license"]')              # Click element on page
10         self.assert_text("free to copy", "div center") # Assert text in area
11         self.open("https://xkcd.com/1481/")
12         title = self.get_attribute("#comic img", "title") # Get an attribute
13         self.assert_true("86,400 seconds per day" in title)
14         self.click("link=Blag")                      # Click on link
15         self.assert_text("The blag of the webcomic", "h2")
16         self.update_text("input#s", "Robots!\n")      # Type text
17         self.assert_text("Hooray robots!", "#content")
18         self.open("https://xkcd.com/1319/")
19         self.assert_exact_text("Automation", "#ctitle")
20
```

Written in Python

*If you know Python, you can write automation with SeleniumBase.

Built on Selenium / WebDriver

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

```
self.driver.{WEBDRIVER_METHOD}
```

Runs in multiple environments

- *OS X

- *Windows

- *Linux

- *Docker

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$
```

SeleniumBase Linux example

```
Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
```

[illegible]

```
*** 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  ggg.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$
```

Run tests in pytest or nosetest

- *`pytest examples/my_first_test.py`
`--browser=chrome`

- *`nosetests examples/my_first_test.py`
`--browser=firefox`

Learn More

<http://seleniumbase.com>

There's also MasterQA

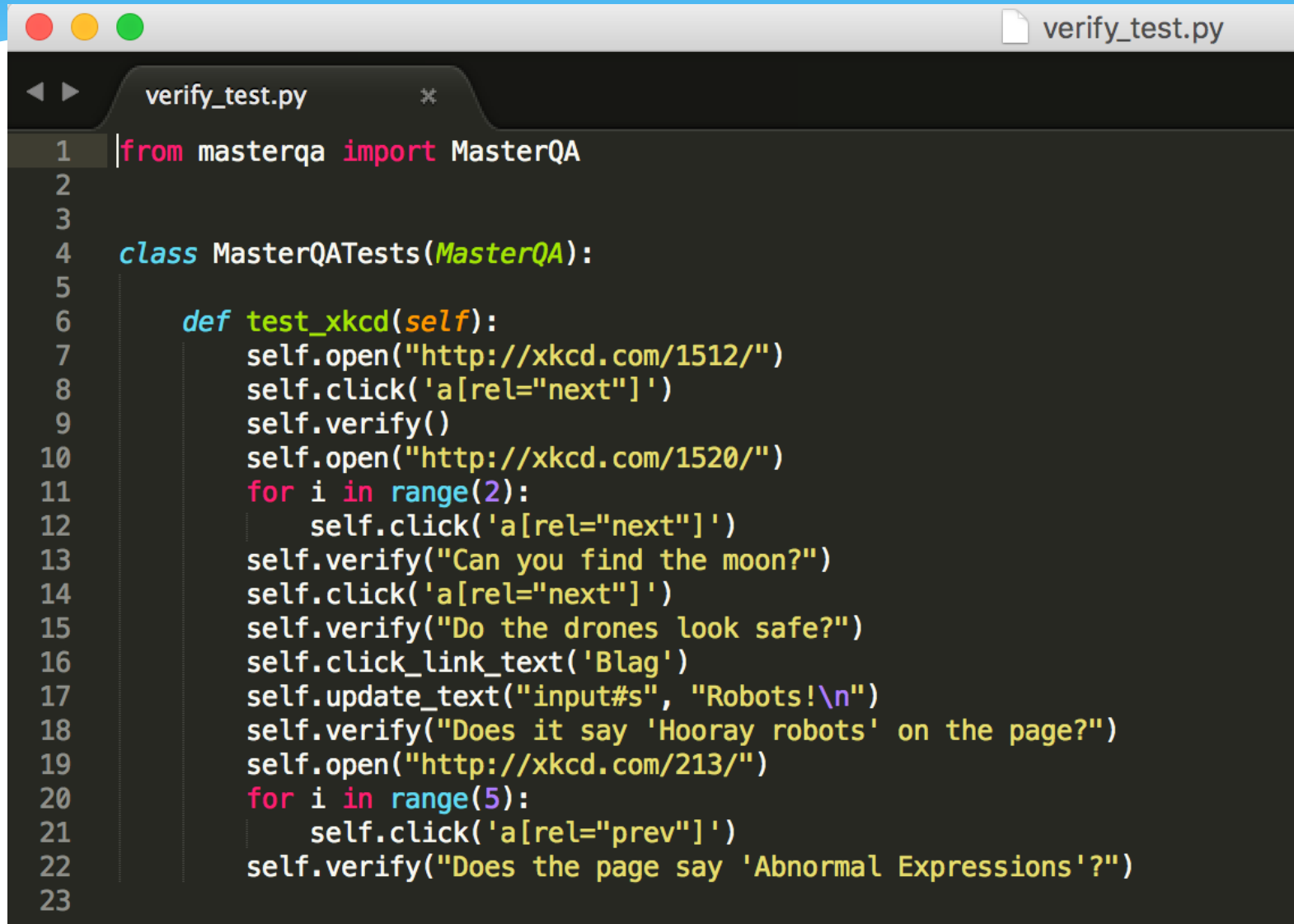
Automation-Powered Acceptance Testing

<http://masterqa.com>

Built on top of the SeleniumBase platform.

100% Open Source

MasterQA - example test



The image shows a code editor window with a tab labeled 'verify_test.py'. The code is a Python script for testing the MasterQA application. It starts with an import statement for 'MasterQA' from the 'masterqa' module. Then, a class 'MasterQATests' is defined, inheriting from 'MasterQA'. Inside this class, a method 'test_xkcd' is defined. This method performs a series of actions: it opens a URL, clicks a 'next' link, verifies a page, opens another URL, clicks 'next' twice, verifies two specific questions, clicks a link labeled 'Blag', updates an input field with 'Robots!', verifies another question, opens a third URL, clicks a 'prev' link, and finally verifies a question about 'Abnormal Expressions'.

```
1 |from masterqa import MasterQA
2
3
4 |class MasterQATests(MasterQA):
5
6 |    def test_xkcd(self):
7 |        self.open("http://xkcd.com/1512/")
8 |        self.click('a[rel="next"]')
9 |        self.verify()
10 |        self.open("http://xkcd.com/1520/")
11 |        for i in range(2):
12 |            self.click('a[rel="next"]')
13 |            self.verify("Can you find the moon?")
14 |            self.click('a[rel="next"]')
15 |            self.verify("Do the drones look safe?")
16 |            self.click_link_text('Blag')
17 |            self.update_text("input#s", "Robots!\n")
18 |            self.verify("Does it say 'Hooray robots' on the page?")
19 |            self.open("http://xkcd.com/213/")
20 |            for i in range(5):
21 |                self.click('a[rel="prev"]')
22 |            self.verify("Does the page say 'Abnormal Expressions'?")
23
```

MasterQA - example run

Browser window showing the xkcd Astronomy comic page (https://xkcd.com/1522/).

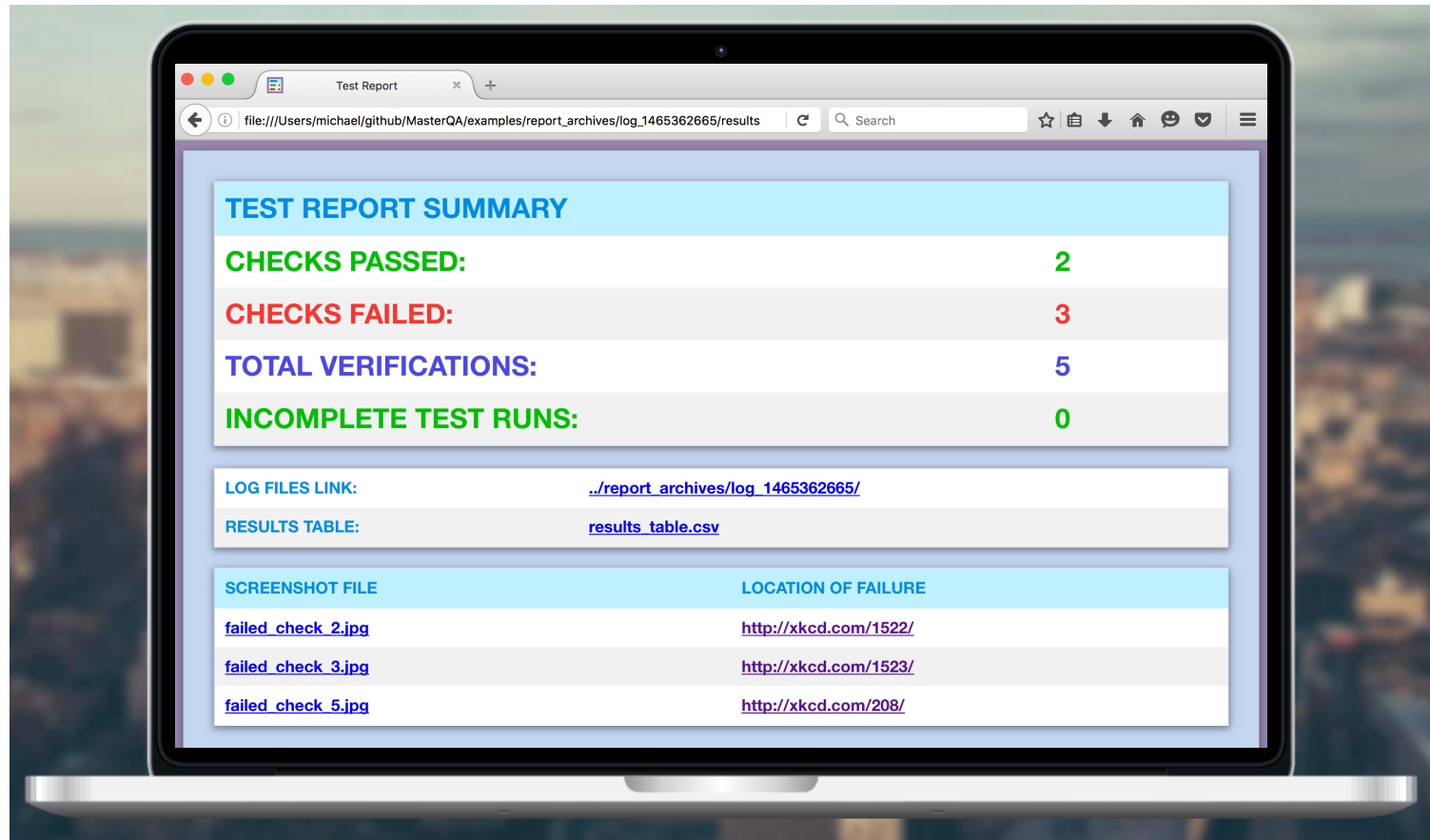
The page features a navigation menu on the left: ARCHIVE, WHAT IF?, BLOG, STORE, ABOUT.

The main header displays the xkcd logo and the tagline: A WEBCOMIC OF ROMANCE, SARCASM, MATH, AND LANGUAGE.

A modal dialog box titled "MANUAL CHECK #2:" is overlaid on the page, asking: "CAN YOU FIND THE MOON?". The dialog includes two buttons: "NO / FAIL" (red) and "YES / PASS" (green).

The comic strip is titled "ASTRONOMY" and consists of four panels. The panels show two characters observing the night sky through a telescope. In the first panel, one character is looking through the telescope while the other stands nearby. In the second panel, the second character is using a ladder to look through the telescope. In the third panel, the first character is using a ladder to look through the telescope. In the fourth panel, both characters are using ladders to look through the telescope.

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