

Web Automation for Testing, Time-Saving, and Profit

By Michael Mintz

About Me

- *I like to automate things.
- *I've built automation for HubSpot, Jana, Veracode, 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
- * The latest iteration is called “WebDriver” (aka Selenium 2.0)

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`
- * Download web browsers (plus drivers) that you want to run automation on.

Try an example test

- *> `git clone https://github.com/seleniumbase/SeleniumBase.git`
- *> `cd SeleniumBase/examples`
- *> `nosetests my_first_test.py --with-selenium`
(Have **Firefox** installed before trying that)

Reliable methods

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

Built-in Functionality

- *MySQL DB integration
- *S3 uploading
- *SeleniumGrid
- *Scanning emails
- *Headless browsers
- *Screenshots

Command-line control

- * Browser selection
- * Log path selection
- * Automation speed selection
- * MySQL option
- * S3 logging option
- * And more...

Configurable

- * Set timeout lengths
- * Set MySQL DB to connect to
- * Set S3 bucket to upload log files to
- * And more...

How to Configure

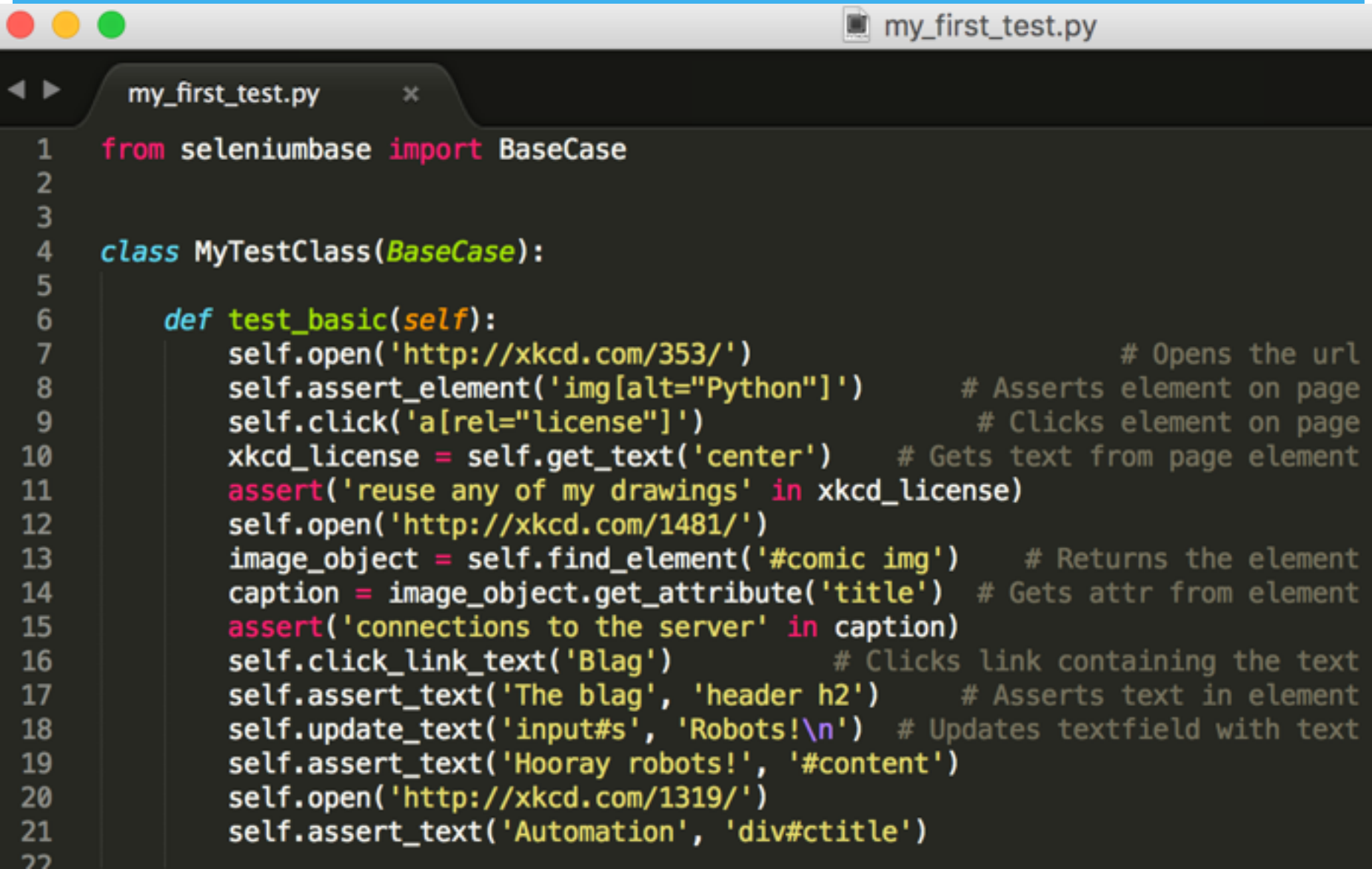
- *Make changes to “settings.py”

(located in SeleniumBase/seleniumbase/config/)

- *> python setup.py install

(That deploys your changes)

Sample script (easy to write)



```
1 from seleniumbase import BaseCase
2
3
4 class MyTestClass(BaseCase):
5
6     def test_basic(self):
7         self.open('http://xkcd.com/353/') # Opens the url
8         self.assert_element('img[alt="Python"]') # Asserts element on page
9         self.click('a[rel="license"]') # Clicks element on page
10        xkcd_license = self.get_text('center') # Gets text from page element
11        assert('reuse any of my drawings' in xkcd_license)
12        self.open('http://xkcd.com/1481/')
13        image_object = self.find_element('#comic img') # Returns the element
14        caption = image_object.get_attribute('title') # Gets attr from element
15        assert('connections to the server' in caption)
16        self.click_link_text('Blag') # Clicks link containing the text
17        self.assert_text('The blag', 'header h2') # Asserts text in element
18        self.update_text('input#s', 'Robots!\n') # Updates textfield with text
19        self.assert_text('Hooray robots!', '#content')
20        self.open('http://xkcd.com/1319/')
21        self.assert_text('Automation', 'div#ctitle')
22
```

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.

```

  ____  _
 / ___|| | | |
| |___| |_| |
 \___|_____|_|

```

```
*** 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  qqg.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

- * `py.test examples/my_first_test.py`
`--with-selenium --browser=chrome`
- * `nosetests examples/my_first_test.py`
`--with-selenium --browser=firefox`

All the help you need

- * ReadMe files to assist you with the following integrations:
 - * Docker
 - * Google Cloud
 - * Selenium Grid
 - * And more...

Learn More

<http://seleniumbase.com>

Announcing MasterQA

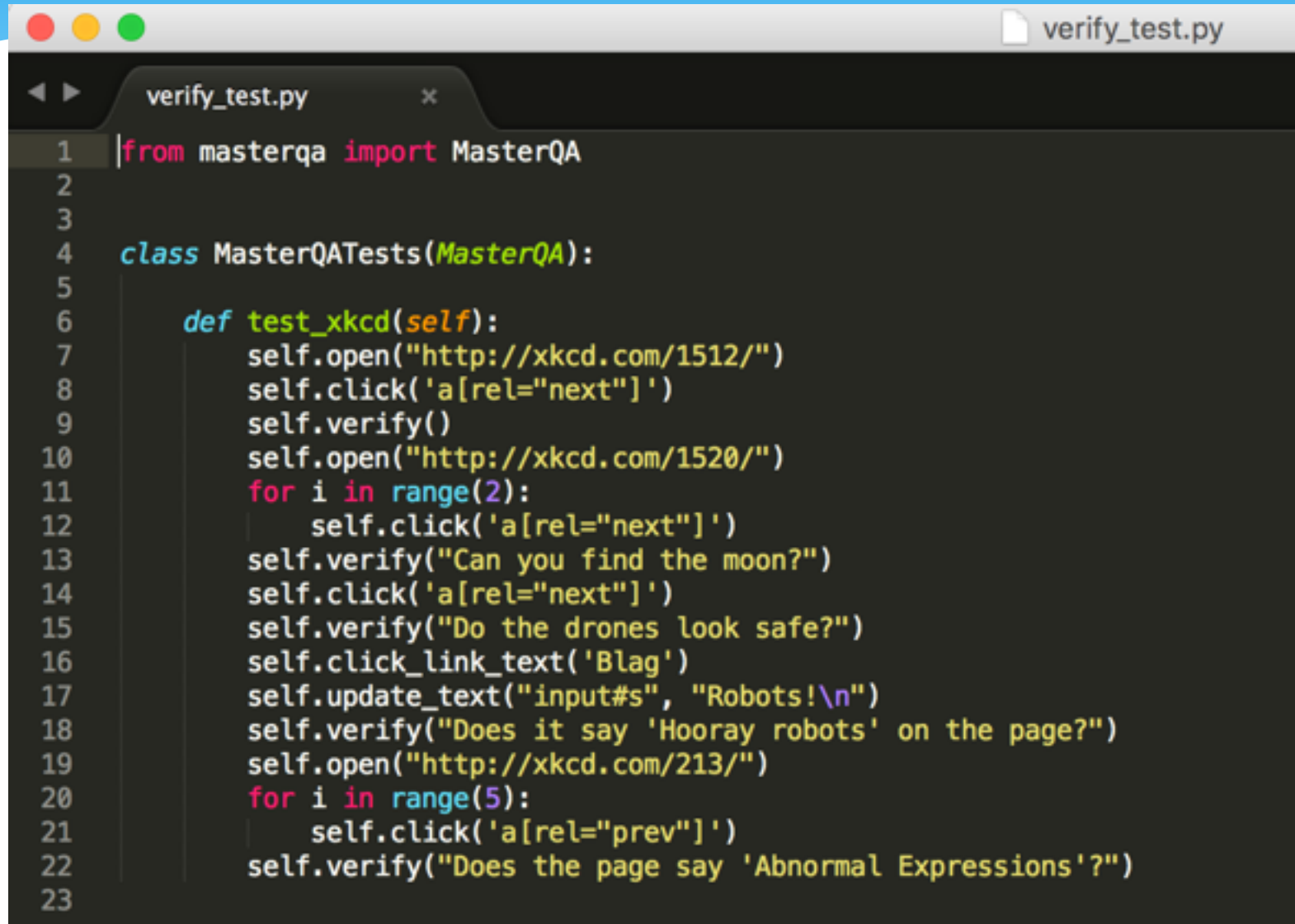
Automation-Powered Acceptance Testing

<http://masterqa.com>

Built on top of the SeleniumBase platform.

100% Open Source

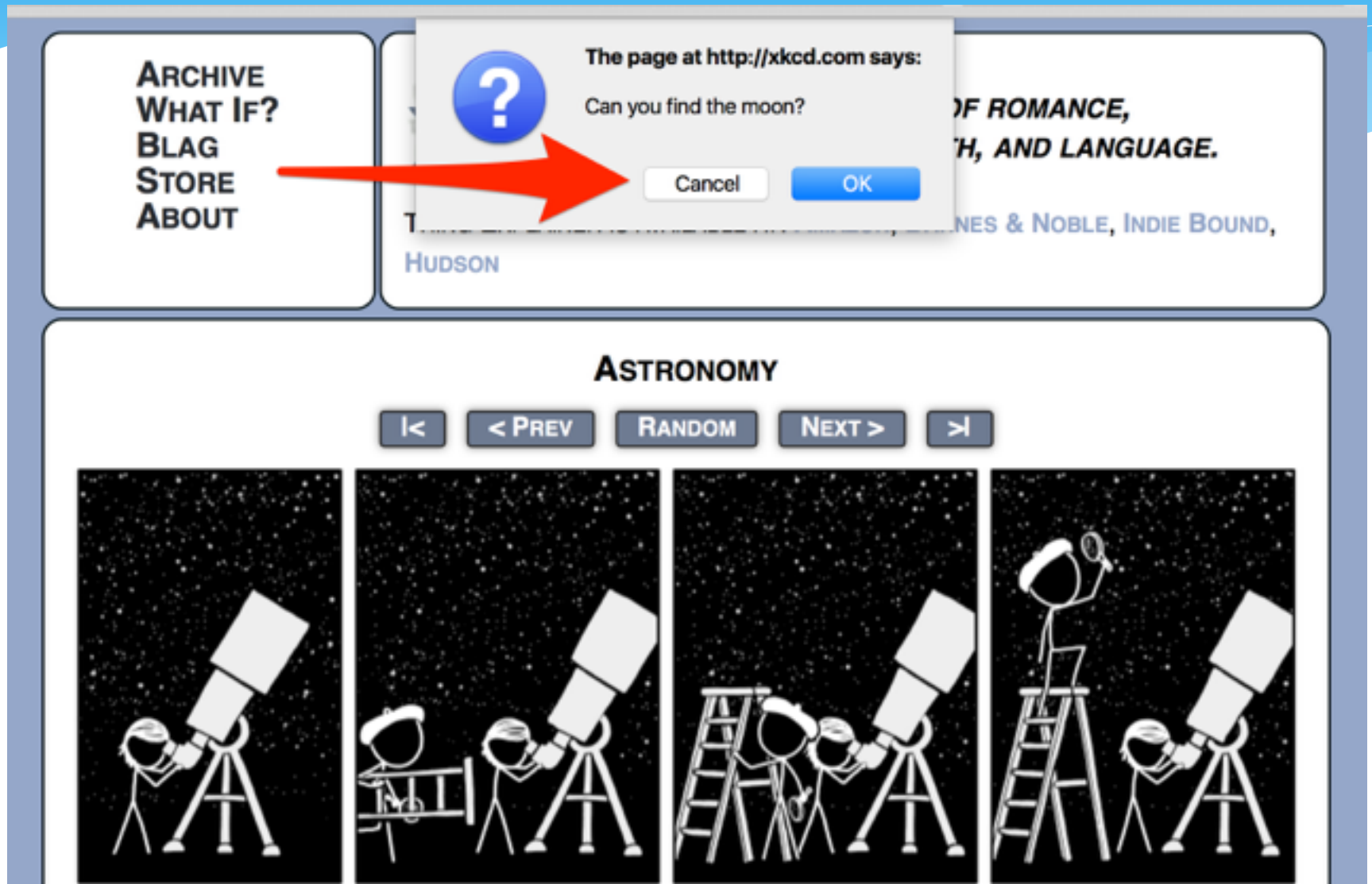
MasterQA - example test



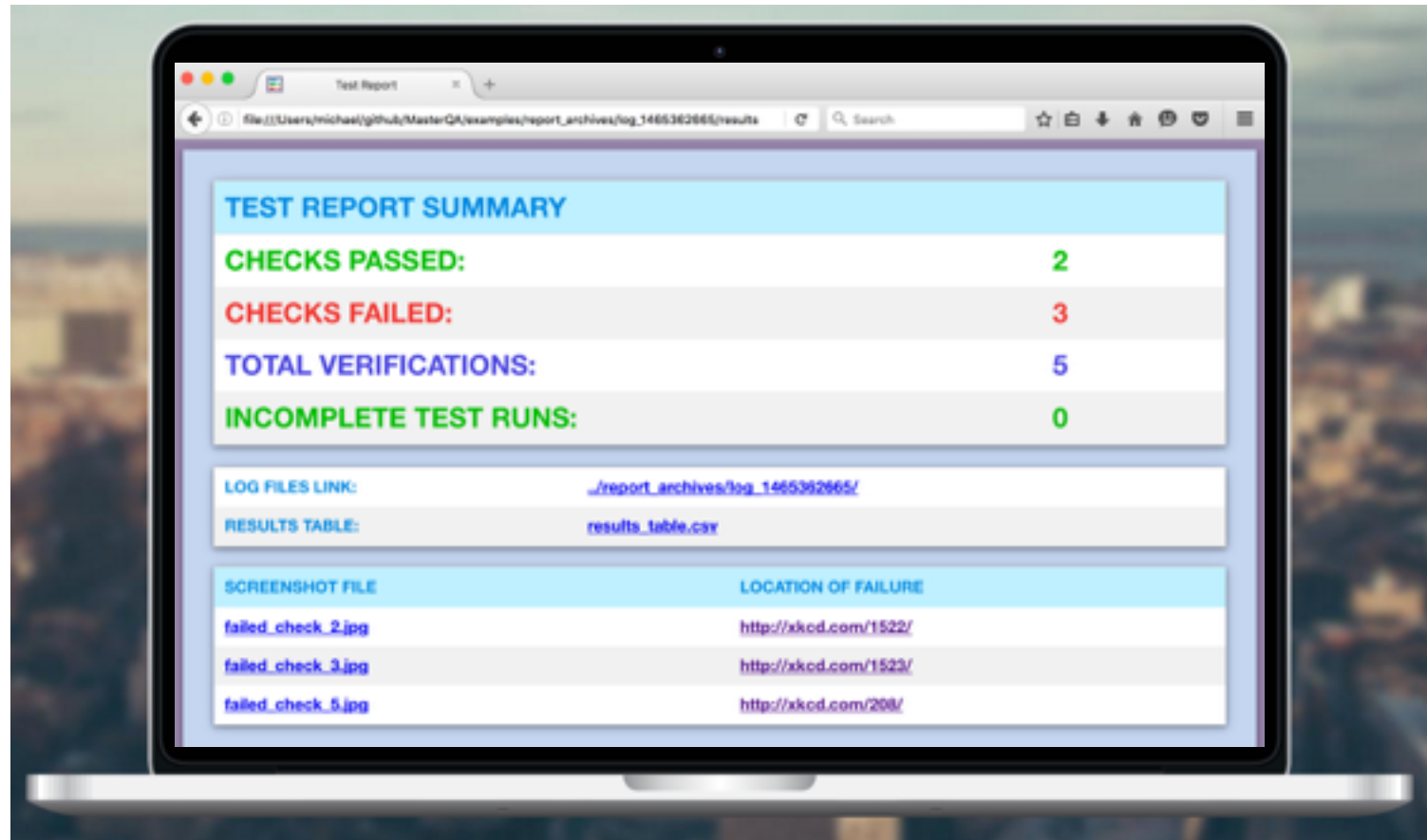
The image shows a code editor window with a tab labeled 'verify_test.py'. The code is a Python script for testing the MasterQA application. It starts with an import statement for MasterQA. Then, a class MasterQATests is defined, inheriting from MasterQA. Inside the class, there is a method test_xkcd. 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 different questions, clicks a link, updates an input field, 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



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