CSCI E-33a

CS50's Web Programming
with Python and JavaScript

Spring 2020

Barbara Karakyriakou · Teaching Fellow

- Email: karakyriakou@fas.harvard.edu
- Phone: 617 820 6930

- Section Meetings: Wednesdays 8:30pm-10:00pm EST
- Office Hours: Saturdays 1:00pm - 2:30pm EST

Section 6: Testing, CI/CD

Agenda

- Testing
  - Python assert function
  - Unittest library
  - Selenium webdriver
  - Django testing
- CI/CD
- Q&A – Problem Set 4 Review

Testing

Testing your code is very important.

- Manual Testing
  - Users test their code by running their applications in an exploratory way
- Automated Testing
  - A script is created with the purpose of testing the code
- Integration Testing
  - Integration test checks multiple components of an application
- Unit Testing
  - Unit test checks each component of an application

Python: assert function

When an assert function runs and finds a bug, the output is an exception: AssertionError.

If there is no error in the code, there is no output.

Example:

```
>>> assert sum([1, 2, 2]) == 6, "Should be 6"
Traceback (most recent call last):
File "<stdin>", line 1, in <module>
AssertionError: Should be 6
```

Python: assert function cont'd

You can return an output even when there is no error in your code, if you want.

Example: (create a python file with name test_assert.py)

```
def test_sum():
        assert sum([1, 2, 2]) == 5, "Should be 5"


if __name__ == "__main__":
        test_sum()
        print("Test passed")
```

On your shell:

```
$ python test_assert.py or $ python3 test_assert.py
Test passed
```

Python: unittest library

unittest has been built into the Python standard library and contains both a testing framework and a test runner. It has, however, some important requirements for writing and executing tests.

- Tests are methods that go into into classes
- Instead of the built-in assert statement, use a series of special assertion methods in the `unittest.TestCase` class

Python: **unittest** library cont'd

Unittest Methods

- `assertEqual`
- `assertNotEqual`
- `assertTrue`
- `assertFalse`
- `assertIn`
- `assertNotIn`

Python: **unittest** library cont'd

Unittest example: (Create a file test_unitest.py)

```
import unittest

class TestSum(unittest.TestCase):

        def test_sum(self):
                self.assertEqual(sum([1, 2, 2]), 5, "Should be 5")

        def test_sum_tuple(self):
                self.assertEqual(sum((1, 2, 3)), 5, "Should be 5")

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

At your shell:

```
$ python3 test_unitest.py

.F
```

You executed two tests: the first passed, but the second failed.

Python: selenium webdriver

Selenium is a tool that is used to test the web applications. To start a web browser, the Selenium module needs a web driver. Python interacts with the selenium web driver and the web driver interacts with the browser.

You can install selenium with `pip` or `pip3 install –U selenium`

To start a browser, you will need to download the corresponding web driver, ChromeDriver, FirefoxDriver etc.

Python: selenium webdriver example code

Create a python file: test_selenium.py

```
import time
from selenium import webdriver

# add the path to the ChromeDriver in your computer
driver = webdriver.Chrome('/Users/barbarak/Downloads/chromedriver’)
# open any website
driver.get('https://python.org’)
# 3sec delay
time.sleep(3)
# quit web driver
driver.quit()
```

Python: selenium webdriver example code

Create a python file: test1_selenium.py

```
import time
from selenium import webdriver

driver = webdriver.Chrome('/Users/barbarak/Downloads/chromedriver')
driver.get('http://www.google.com/’)
time.sleep(5)
search_box = driver.find_element_by_name('q')
search_box.send_keys(‘ChromeDriver')
search_box.submit()
time.sleep(5)
driver.quit()
```

Testing in Django

- Django's unit tests use Python's unittest library
- Tests can be written in the tests.py file which is created by default when starting a Django app
- Create your test classes as subclasses of django.test.TestCase
  - Import: `from django.test import TestCase`
  - Create class: `class SomeTestCase(TestCase):`
- Run tests using Django's test command
  - `$ ./manage.py test`

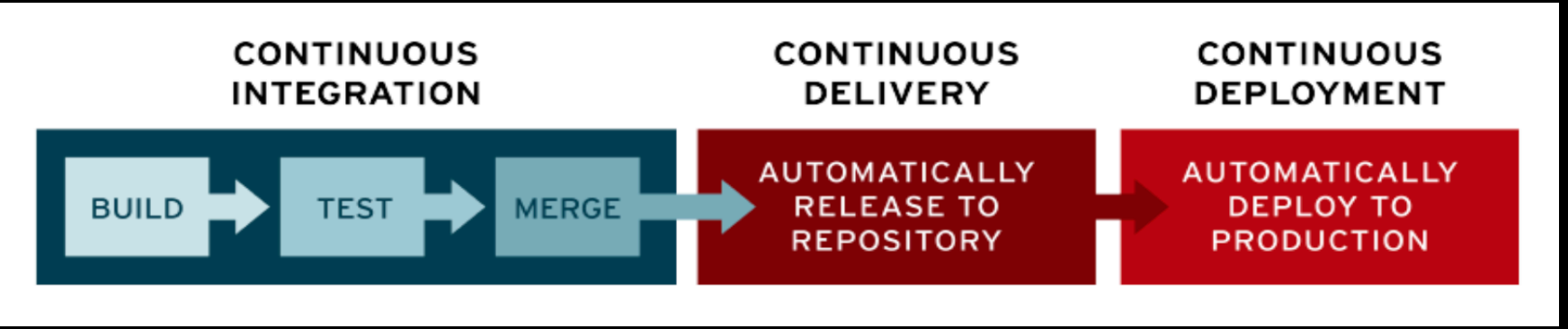
CI/CD

CI/CD stands for continuous integration & continuous delivery and/or continuous deployment.

Continuous *integration* means that new code changes to an app are regularly built, tested, and merged to a shared repository.

Continuous *delivery* means that changes to an application are automatically bug tested and uploaded to a repository, where they can after be deployed to a live production environment.

Continuous *deployment* refers to automatically releasing changes from the repository to production.



Recommended Resources

- Testing
  - https://docs.python.org/3/library/unittest.html
  - https://selenium-python.readthedocs.io/getting-started.html
  - https://docs.djangoproject.com/en/3.0/topics/testing/overview/
- CI/CD
  - https://help.github.com/en/actions/configuring-and-managing-workflows/configuring-a-workflow
  - https://gabrieltanner.org/blog/an-introduction-to-github-actions

Q&A