Jython

Introduction to Jython programming

Agenda

- Module 1 - Introduction to Jython
- Module 2 - Jython language and semantics
- Module 3 - Data types
- Module 4 - Regular expressions
- Module 5 - Functions, debugging, modules, and packages
- Module 6 - Objects, classes and exceptions
- Module 7 - Java integration
- Module 8 - Testing
- Module 9 - System programming
- Module 10 - Conclusion

Topics

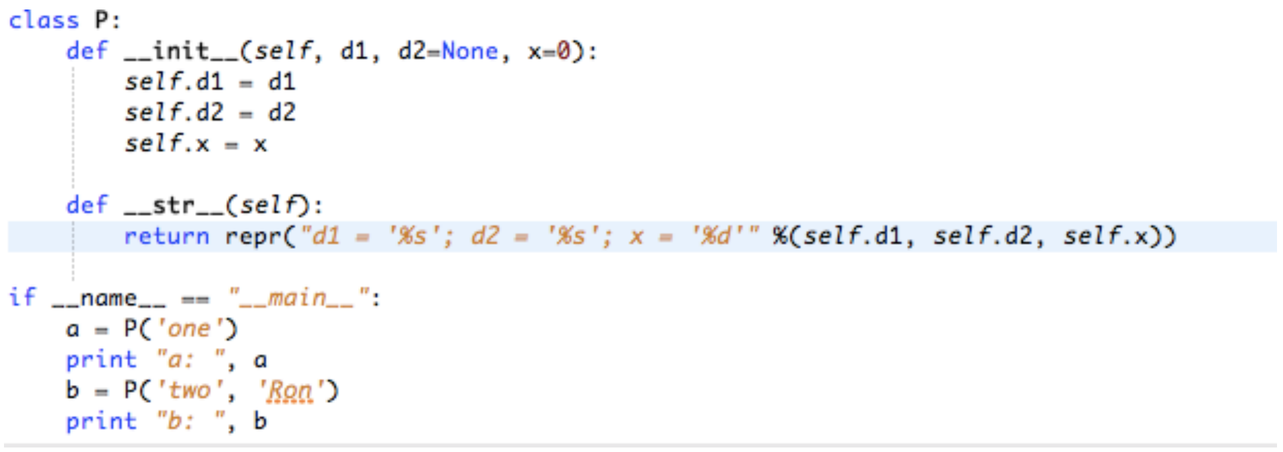
- assert
- doctest
- unittest
- Third party testing frameworks
- Python debugger in-depth
- Quiz
- Q & A

Testing overview

Unit testing during development

- Many modules will contain code to perform basic unit tests

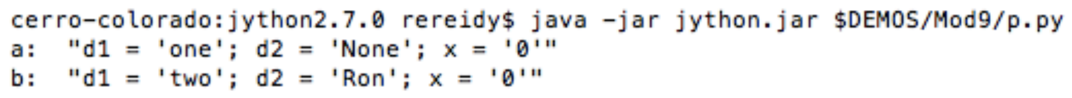
  if __name__ == "__main":



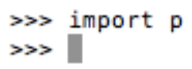
```
class P:
    def __init__(self, d1, d2=None, x=0):
        self.d1 = d1
        self.d2 = d2
        self.x = x

    def __str__(self):
        return repr("d1 = '%s'; d2 = '%s'; x = '%d'" %(self.d1, self.d2, self.x))

if __name__ == "__main__":
    a = P('one')
    print "a: ", a
    b = P('two', 'Ron')
    print "b: ", b
```

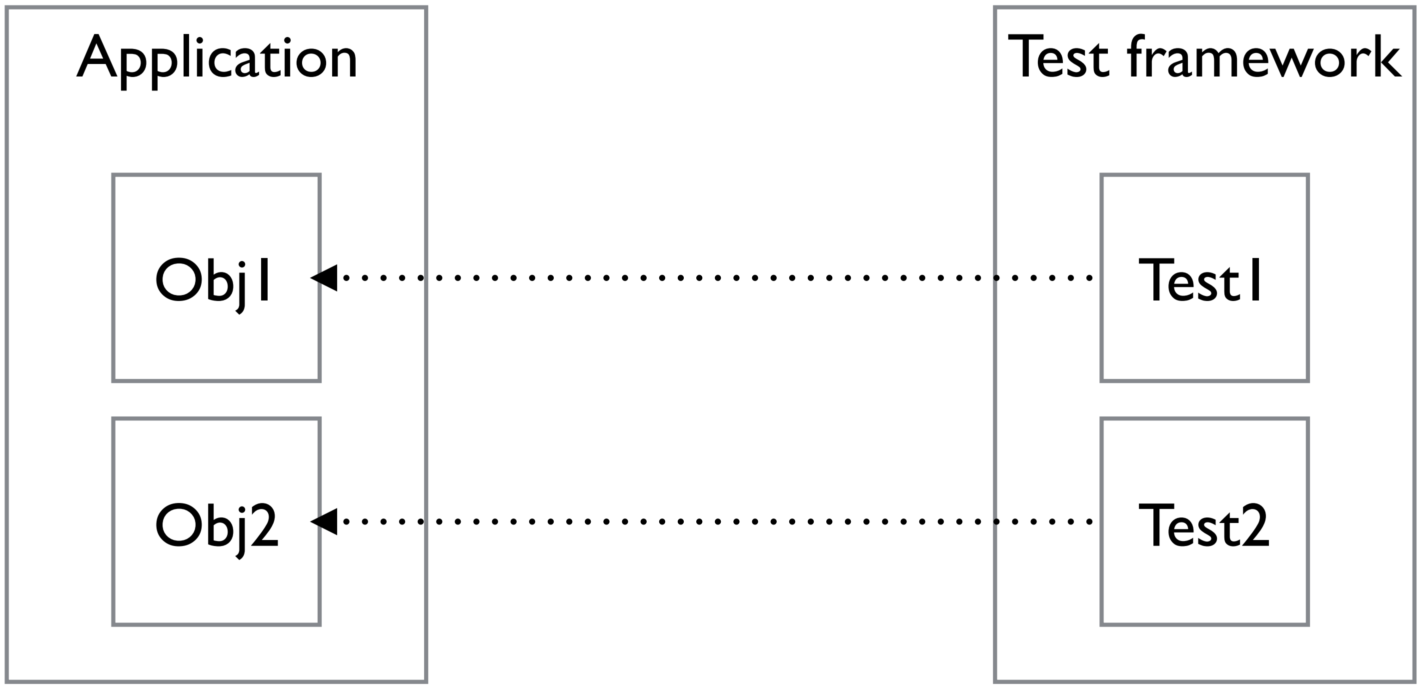


```
cerro-colorado:jython2.7.0 rereidy$ java -jar jython.jar $DEMOS/Mod9/p.py
a:  "d1 = 'one'; d2 = 'None'; x = '0'"
b:  "d1 = 'two'; d2 = 'Ron'; x = '0'"
```



```
>>> import p
>>> 
```

Testing concurrent with software development



Well designed tests

- A unit test should test the behavior of one single unit of code
  - Success or failure is the validation of code
- Set up a scenario independent of other conditions
  - Avoid dependencies on other test results

Test types

- Black box (functional)
- White box (structural)
    - Python and Java both provide access protection
- Stress testing
- Performance testing
- Usability testing
- Security testing

Testing during development

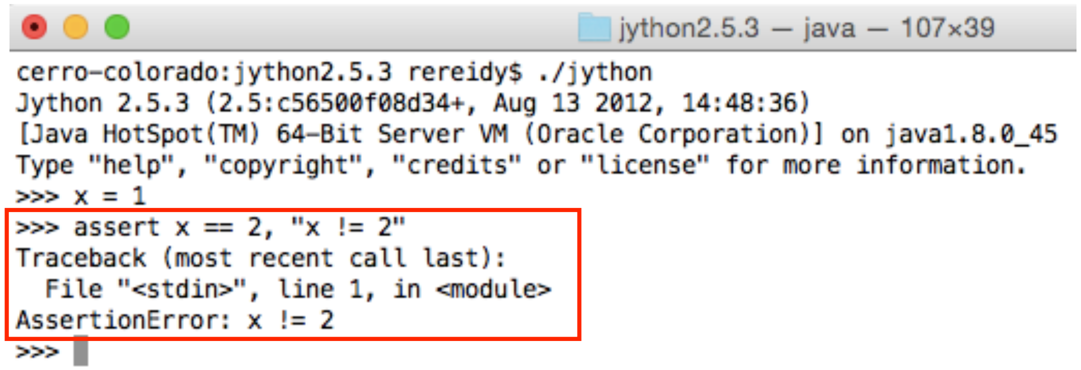
- Programmers usually write tests during development
  - Test functionality
  - Test values, arguments, etc.

assert

Testing during development

- Many programmers user assert to perform testing in their programs

```
assert x == 2, "x != 2"
```



```
cerro-colorado:jython2.5.3 rereidy$ ./jython
Jython 2.5.3 (2.5:c56500f08d34+, Aug 13 2012, 14:48:36)
[Java HotSpot(TM) 64-Bit Server VM (Oracle Corporation)] on java1.8.0_45
Type "help", "copyright", "credits" or "license" for more information.
>>> x = 1
>>> assert x == 2, "x != 2"
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
AssertionError: x != 2
>>> 
```

- Assertions can be optimized away in Python (python -O)
  - This switch is not available in Jython (2.5 or 2.7)

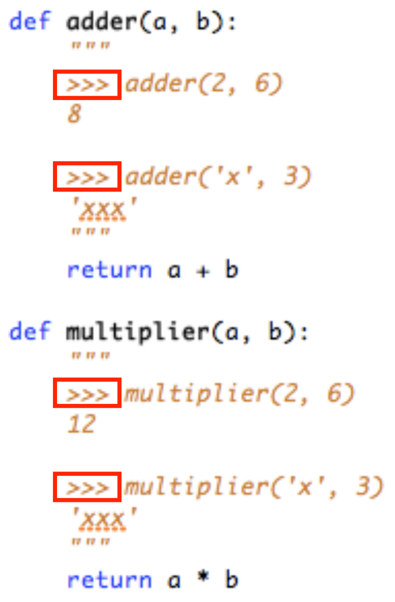
Testing during development$_2$

- `eval()`, `exec()` can also be used
- Not optimized away

doctest

Testing through documentation

- Runs examples embedded in documentation
- Uses the *doctest* module in the standard library
- Only reports failures
  - Can report what was tested (-v)
- Testing can be embedded in `__main__` or invoked on the command line
- Check examples in a separate text file



```
def adder(a, b):
    """
    >>> adder(2, 6)
    8

    >>> adder('x', 3)
    'xxx'
    """
    return a + b

def multiplier(a, b):
    """
    >>> multiplier(2, 6)
    12

    >>> multiplier('x', 3)
    'xxx'
    """
    return a * b
```

How docstrings are recognized

- docstrings are copied from the interactive console
- doctest looks for strings that being with “>>>” and the function name to be tested
- The next line shows the expected output

Cases when output is not predictable

- Objects can change depending on system characteristics
  - Date time
  - Memory usage at time of test
- Modify the docstring in the doctest
  `#doctest: +ELLIPSIS`
  - The expected output will change
    - From *0x1234*
    - To *0x…*

Tracebacks

- Special case of changing output
  - Paths depend on installation location
- doctest recognizes this and ignores pieces that can change between systems
  - Entire body of Traceback ignored

```
Traceback (most recent call last):
```

Other testing

- Normalize whitespace

  `#doctest: +NORMALIZE_WHITESPACE`

- Module level special variable

  `__test__`

  - Dictionary of doctests

unittest

Overview~1~

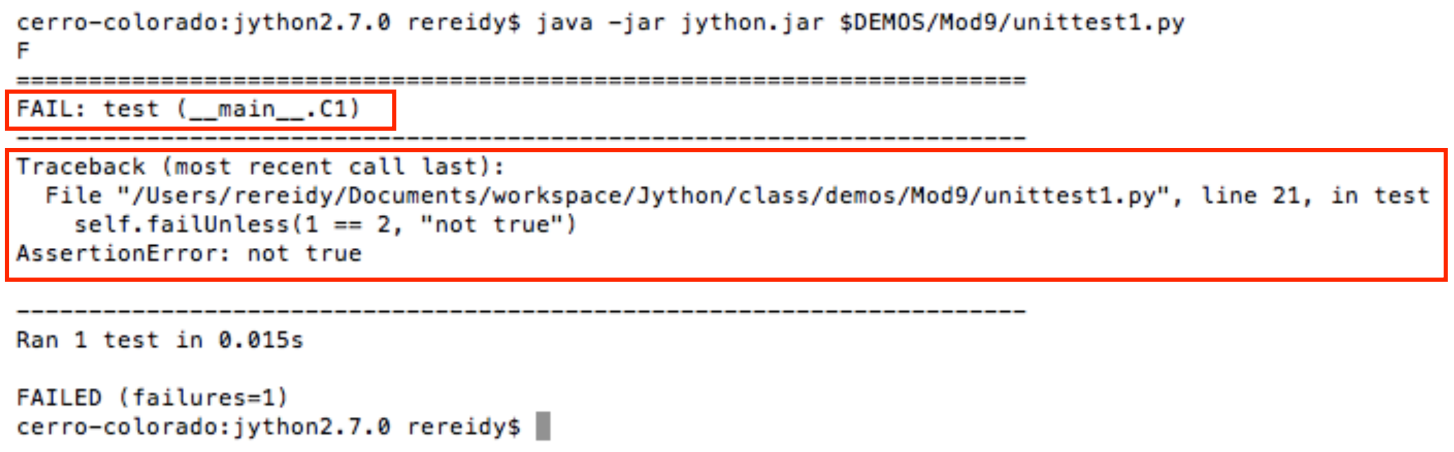
- Python *unittest* module is also known as PyUnit
- Part of the Python standard library
- Based on Java *JUnit* framework
  - *JUnit* is derived from *XUnit* family of testing frameworks
  - Most languages have a testing framework derived from *JUnit*
- Unit tests have three outcomes
  - *OK* - the test passes
  - *FAIL* - Test does not pass and raises *AssertionError* exception
  - *ERROR* - test raises an exception other than *AssertionError*

Overview$_2$

- *unittest* supports test automation
  - test fixture - preparation of objects
  - test case - smaller unit to testing
  - test suite - collection of test cases, tests suites, or both
  - test runner - orchestrates the execution of tests

Failed tests

- Generate a traceback in the output



```
cerro-colorado:jython2.7.0 rereidy$ java -jar jython.jar $DEMOS/Mod9/unittest1.py
F
======================================================================
FAIL: test (__main__.C1)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "/Users/rereidy/Documents/workspace/Jython/class/demos/Mod9/unittest1.py", line 21, in test
    self.failUnless(1 == 2, "not true")
AssertionError: not true

----------------------------------------------------------------------
Ran 1 test in 0.015s

FAILED (failures=1)
cerro-colorado:jython2.7.0 rereidy$
```

Assertion types

Asserting truth

- Four methods
  - *failUnless()*
  - *assertTrue()*
  - *failIf*
  - *assertFalse()*

Asserting truth

- Two methods to test equality of two values
  - *failUnlessEqual()*
  - *failIfEqual()*

Almost equal

- Two methods for testing floating point near equality
  - *failIfAlmostEqual()*
  - *failUnlessAlmostEqual()*
- Specify number of decimal places to test (*places*=*n*)

Exception testing

- One method for testing an exception other than *AssertionError* is raised
  - *failUnlessRaises()*

Test fixtures

Creation and clean up hooks

- Implemented with the *unittest.TestCase* method
  - *setUp()*
  - *tearDown()*

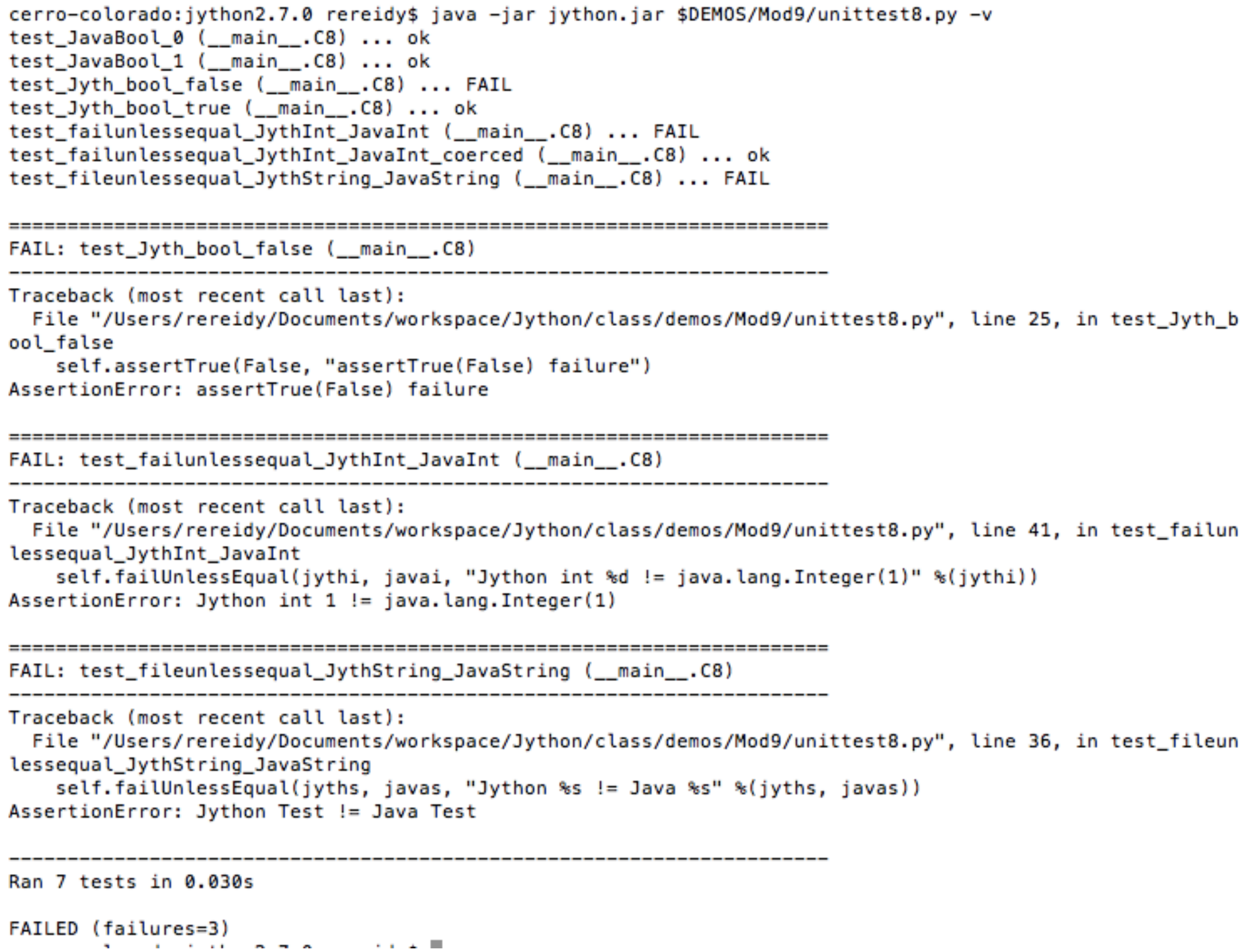
Combining doctest and unittest

doctest extended

- *doctest* has two classes to allow integration with *unittest*
- *DocTestSuite()* - run embedded doctests
- *DocFileSuite()* - run doctests stored in external file
- Executed from *unites.TestRunner()*

Basic Java testing

- Java types should map to Jython types



```
cerro-colorado:jython2.7.0 rereidy$ java -jar jython.jar $DEMOS/Mod9/unittest8.py -v
test_JavaBool_0 (__main__.C8) ... ok
test_JavaBool_1 (__main__.C8) ... ok
test_Jyth_bool_false (__main__.C8) ... FAIL
test_Jyth_bool_true (__main__.C8) ... ok
test_failunlessequal_JythInt_JavaInt (__main__.C8) ... FAIL
test_failunlessequal_JythInt_JavaInt_coerced (__main__.C8) ... ok
test_fileunlessequal_JythString_JavaString (__main__.C8) ... FAIL

======================================================================
FAIL: test_Jyth_bool_false (__main__.C8)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "/Users/rereidy/Documents/workspace/Jython/class/demos/Mod9/unittest8.py", line 25, in test_Jyth_b
ool_false
    self.assertTrue(False, "assertTrue(False) failure")
AssertionError: assertTrue(False) failure

======================================================================
FAIL: test_failunlessequal_JythInt_JavaInt (__main__.C8)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "/Users/rereidy/Documents/workspace/Jython/class/demos/Mod9/unittest8.py", line 41, in test_failun
lessequal_JythInt_JavaInt
    self.failUnlessEqual(jythi, javai, "Jython int %d != java.lang.Integer(1)" %(jythi))
AssertionError: Jython int 1 != java.lang.Integer(1)

======================================================================
FAIL: test_fileunlessequal_JythString_JavaString (__main__.C8)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "/Users/rereidy/Documents/workspace/Jython/class/demos/Mod9/unittest8.py", line 36, in test_fileun
lessequal_JythString_JavaString
    self.failUnlessEqual(jyths, javas, "Jython %s != Java %s" %(jyths, javas))
AssertionError: Jython Test != Java Test

----------------------------------------------------------------------
Ran 7 tests in 0.030s

FAILED (failures=3)
```

- Java types should map to Jython types, but clearly not always…

Third party test frameworks

nose

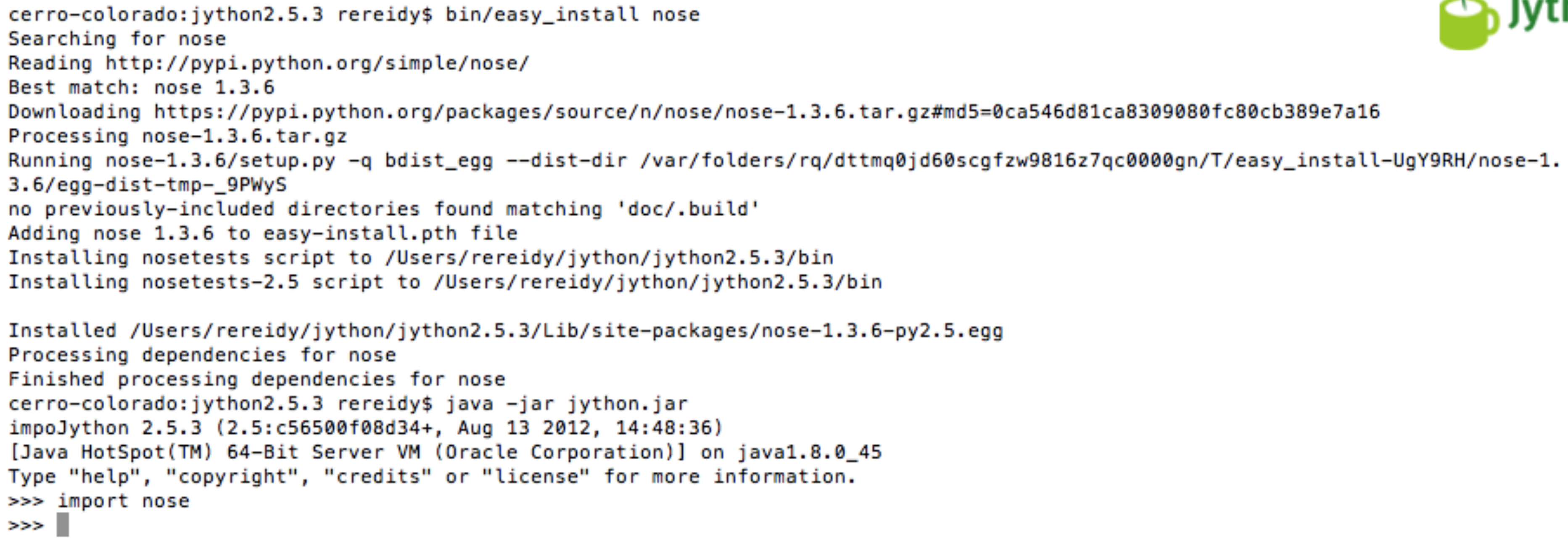
Overview

- Extension to *unittest*
- Not part of the standard library
  - Jython 2.5.3 install

    ```
    bin/easy_install nose
    ```
  - Jython 2.7.0 install

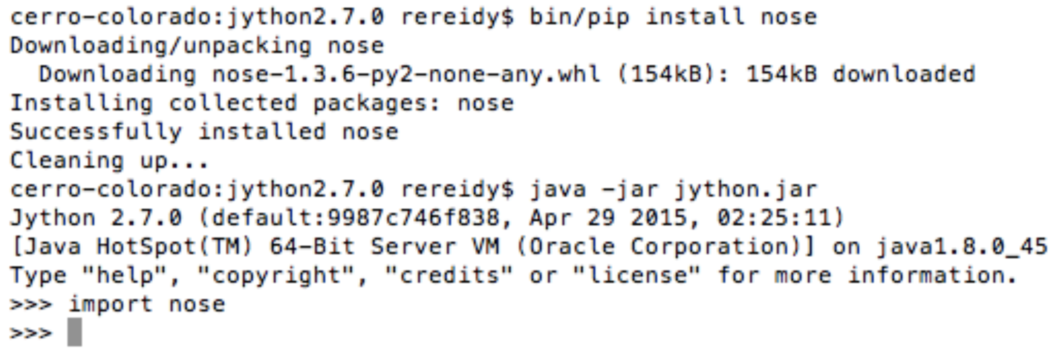
    ```
    bin/pip install nose
    ```
  - Installs a script to execute the tests

    ```
    bin/nosetests
    ```



```
cerro-colorado:jython2.5.3 rereidy$ bin/easy_install nose
Searching for nose
Reading http://pypi.python.org/simple/nose/
Best match: nose 1.3.6
Downloading https://pypi.python.org/packages/source/n/nose/nose-1.3.6.tar.gz#md5=0ca546d81ca8309080fc80cb389e7a16
Processing nose-1.3.6.tar.gz
Running nose-1.3.6/setup.py -q bdist_egg --dist-dir /var/folders/rq/dttmq0jd60scgfzw9816z7qc0000gn/T/easy_install-UgY9RH/nose-1.
3.6/egg-dist-tmp-_9PWyS
no previously-included directories found matching 'doc/.build'
Adding nose 1.3.6 to easy-install.pth file
Installing nosetests script to /Users/rereidy/jython/jython2.5.3/bin
Installing nosetests-2.5 script to /Users/rereidy/jython/jython2.5.3/bin

Installed /Users/rereidy/jython/jython2.5.3/Lib/site-packages/nose-1.3.6-py2.5.egg
Processing dependencies for nose
Finished processing dependencies for nose
cerro-colorado:jython2.5.3 rereidy$ java -jar jython.jar
impoJython 2.5.3 (2.5:c56500f08d34+, Aug 13 2012, 14:48:36)
[Java HotSpot(TM) 64-Bit Server VM (Oracle Corporation)] on java1.8.0_45
Type "help", "copyright", "credits" or "license" for more information.
>>> import nose
>>> 
```



```
cerro-colorado:jython2.7.0 rereidy$ bin/pip install nose
Downloading/unpacking nose
  Downloading nose-1.3.6-py2-none-any.whl (154kB): 154kB downloaded
Installing collected packages: nose
Successfully installed nose
Cleaning up...
cerro-colorado:jython2.7.0 rereidy$ java -jar jython.jar
Jython 2.7.0 (default:9987c746f838, Apr 29 2015, 02:25:11)
[Java HotSpot(TM) 64-Bit Server VM (Oracle Corporation)] on java1.8.0_45
Type "help", "copyright", "credits" or "license" for more information.
>>> import nose
>>> 
```

Running nose

- Use the *nosetests* command that is installed with the nose framework
- Several plug-ins built in

Nose fixtures

- *nose* extends the *unittest* fixture model
    - *setup_module()*
    - *teardown_module()*

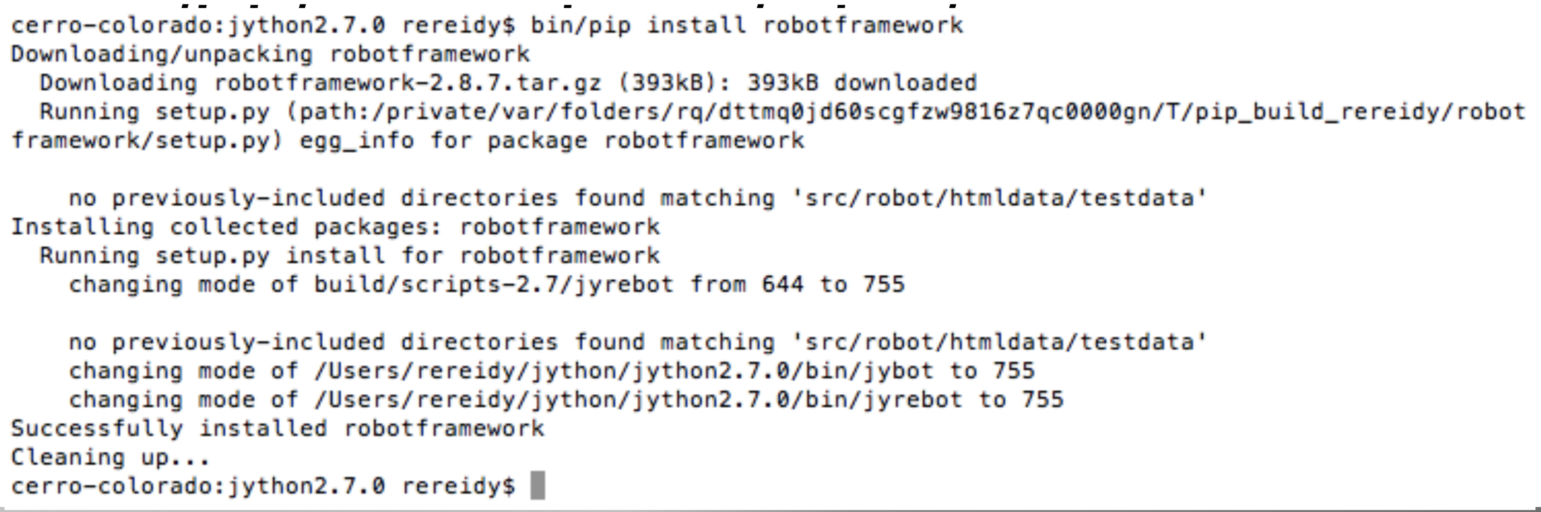
Robot framework

Overview

- Not part of the standard library
- Generic test automation and acceptance tool
  - Keyword driven approach, table driven test data approach
- Not available for Jython 2.5.3
  - All examples will be using Jython 2.7.0

Installation

- Install using *bin/pip install robot framework*
  - Option JAR file with Jython installed
- Startup and post processing commands
  - *jybot and jyrebot*
- Can also be started using



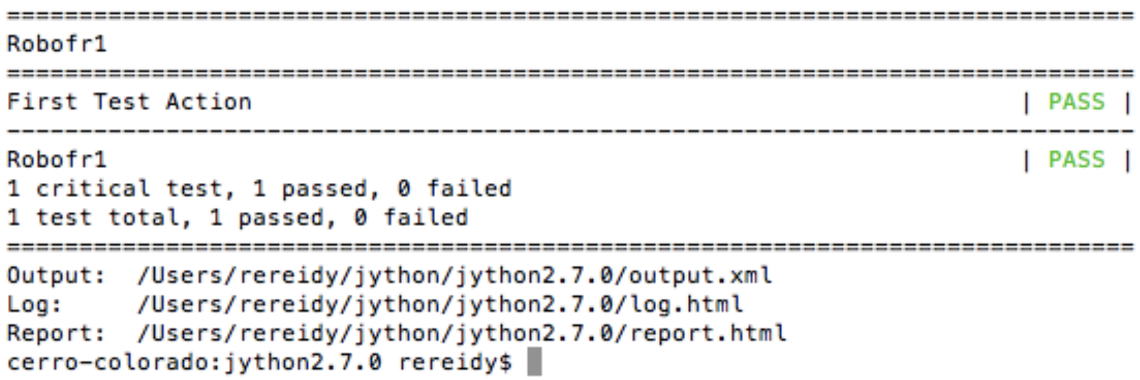
```
cerro-colorado:jython2.7.0 rereidy$ bin/pip install robotframework
Downloading/unpacking robotframework
  Downloading robotframework-2.8.7.tar.gz (393kB): 393kB downloaded
  Running setup.py (path:/private/var/folders/rq/dttmq0jd60scgfzw9816z7qc0000gn/T/pip_build_rereidy/robot
framework/setup.py) egg_info for package robotframework

    no previously-included directories found matching 'src/robot/htmldata/testdata'
Installing collected packages: robotframework
  Running setup.py install for robotframework
    changing mode of build/scripts-2.7/jyrebot from 644 to 755

    no previously-included directories found matching 'src/robot/htmldata/testdata'
    changing mode of /Users/rereidy/jython/jython2.7.0/bin/jybot to 755
    changing mode of /Users/rereidy/jython/jython2.7.0/bin/jyrebot to 755
Successfully installed robotframework
Cleaning up...
cerro-colorado:jython2.7.0 rereidy$ 
```

First test

- Empty test file



```
==============================================================================
Robofr1
==============================================================================
First Test Action                                                     | PASS |
------------------------------------------------------------------------------
Robofr1                                                               | PASS |
1 critical test, 1 passed, 0 failed
1 test total, 1 passed, 0 failed
==============================================================================
Output:  /Users/rereidy/jython/jython2.7.0/output.xml
Log:     /Users/rereidy/jython/jython2.7.0/log.html
Report:  /Users/rereidy/jython/jython2.7.0/report.html
cerro-colorado:jython2.7.0 rereidy$
```

Python debugger in-depth

Module: pdb

- Part of the standard library
- Interactive debugging
- Embed in program

Module: pdb

- Post-mortem debugging
  - Interactive console

```
>>> import script_name
>>> import pdb
>>> script_name.crash_function()
>>> pdb.pm()
```

Quiz

1. Why is *assert* not a proper method for long term testing?

A. Assertions can be optimized away using the *-O* command line option.

2. How does doctest identify tests and expected output?

A. Tests are identified in docstrings. Tests begin with the string “>>> *object_to_test*”. Expected results are on the next line.

3. What is the __*test*__ module level variable and how is it used in doctest testing?

A. __*test*__ is a special variable containing a dictionary of tests to be performed at the module level.

Q & A

Exercises

1. Write module to implement doctests at the module, class, class method, and function layers.

Execute doctest.

4. Explore Java data types we did not use in this module and test against Jython types using *unittest*:

Jython type	Java type
None	null
Float	Double
Integer	Short
Integer	Byte

Java null == Python None

5. Create a module which includes the python debugger and set trace() points (multiple). Examine the behavior of the debugger when running the program.

6. Create a program which will loop through a range of integers and print them. Run this program through an interactive pdb session. Set a conditional break point to stop on the print statement when the loop reaches a particular value (e.g. i == 10).