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

- Overview of Jython
- Philosophy
- JVM scripting languages
- Why Jython
- Running Jython - a small taste
- Configuration, environment, and registry
- The execution environment
- Examine the Jython installation
- Quiz
- Q & A

Overview of Jython

Preliminaries

Jython

- Jython **IS** Python implemented in the JVM
  - Versions are compatible for language features
    - Jython 2.5 => Python 2.5
    - Jython 2.7 => Python 2.7
  - Some differences between Python and Jython
    - Libraries
    - Features

The future

- Python 3 is the future of Python
  - Replacement of Python 2
- No Jython 3 - yet

Implementations of Python

- CPython - Python written in C
- Cython - Python C/C++ translator
- Jython - Python on the JVM
- IronPython - Python on .NET
- PyPy - Python with a JIT compiler

Philosophy of Python

Philosophy of Python

- Code readability
  - Coherence, aesthetics
- Dynamic typing
- Highly extensible
  - Simplicity
- Sparse, less cluttered grammar (vs. Perl's "there's more than one way to do it" philosophy)
- Beautiful is better than ugly
- Explicit is better than implicit
- Simple is better than complex
- Complex is better than complicated
- Readability counts

JVM scripting languages

JSR-223

- Allows access to the JVM
- As of 2013 - 50 JVM scripting languages

https://jcp.org/en/jsr/detail?id=223
https://en.wikipedia.org/wiki/List_of_JVM_languages

Why Jython

- Dynamic compilation to Java bytecode
- Automatic creation of Java/Python mapping
- Ability to extend existing Java classes in Jython - allows effective use of abstract classes.
- Optional static compilation - allows creation of applets, servlets, beans, etc.
- Bean Properties - make use of Java packages much easier.
- Python Language - combines very clear syntax with the JVM; supports a full object-oriented programming model which makes it a natural fit for Java's OO design.

What Jython does really well

- Prototyping
- Java investigation
  - Making properties accessible to Jython
  - Glues libraries written in Java
  - An excellent embedded scripting language

Where Jython can be used

- Web application servers (servlet container)
  - WebSphere wsadmin scripting
  - WebLogic wlst scripting
  - Apache Tomcat
  - Jetty
- Web applications
  - Django, Pylons
  - Bean scripting
- Ant/Maven scripts
- Console (CLI) and GUI scripts
- Testing
  - Selenium (Jython 2.7)
  - BurpSuite

Differences between Python and Jython

Python	**Jython**
• Written in C	• Written in Java
• Multi-platform	• Multi-platform (any JVM from 1.1 up)
• Extensible with C/C++	• Extensible with Java
• Threading implemented using the Global Interpreter Lock	• True multithreading
• Python garbage collection	• Java garbage collection

Pros and cons

- Pros
  - Access to all Java classes
  - Jython components can be integrated into other Java/ Jython based systems
- Cons
  - Slow

Running Jython

What is Jython

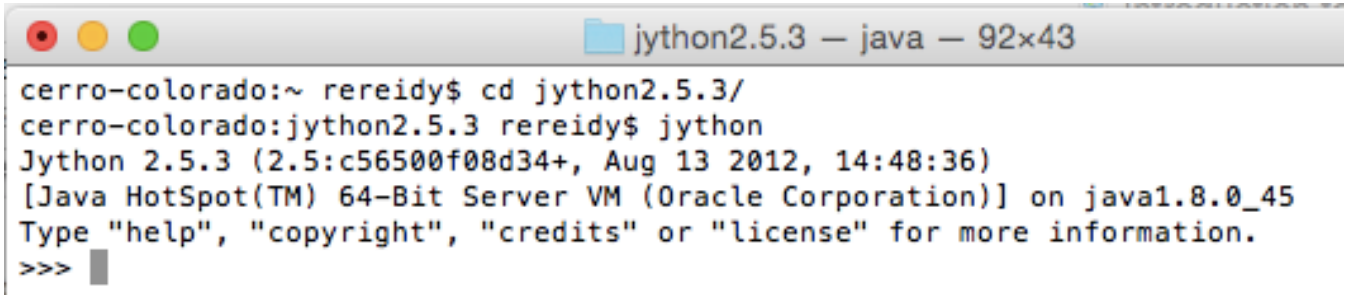
- JAR file
  - Compiles Python expressions to Java bytecode
  - Executes the bytecode

A Jython program

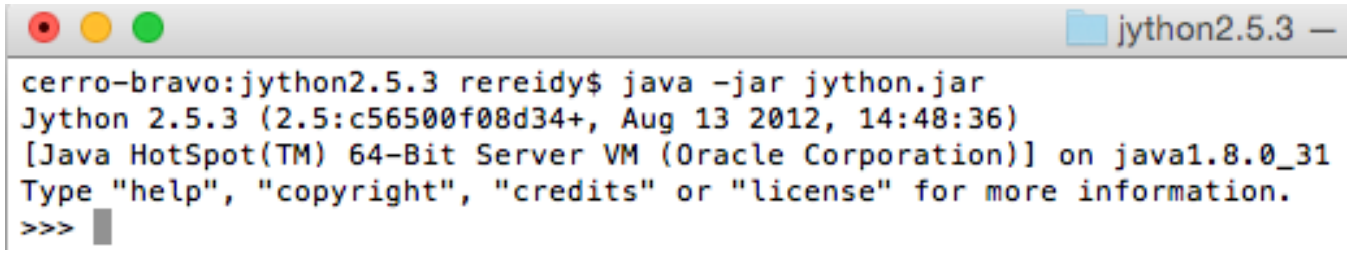
- A text file with Python statements and expressions
- Executed by the Jython interpreter from top to bottom of the file

Invoking the interpreter

- To start Jython:
  - cd to the Jython installation directory
  - Type "./jython" at the prompt



- Start with Java



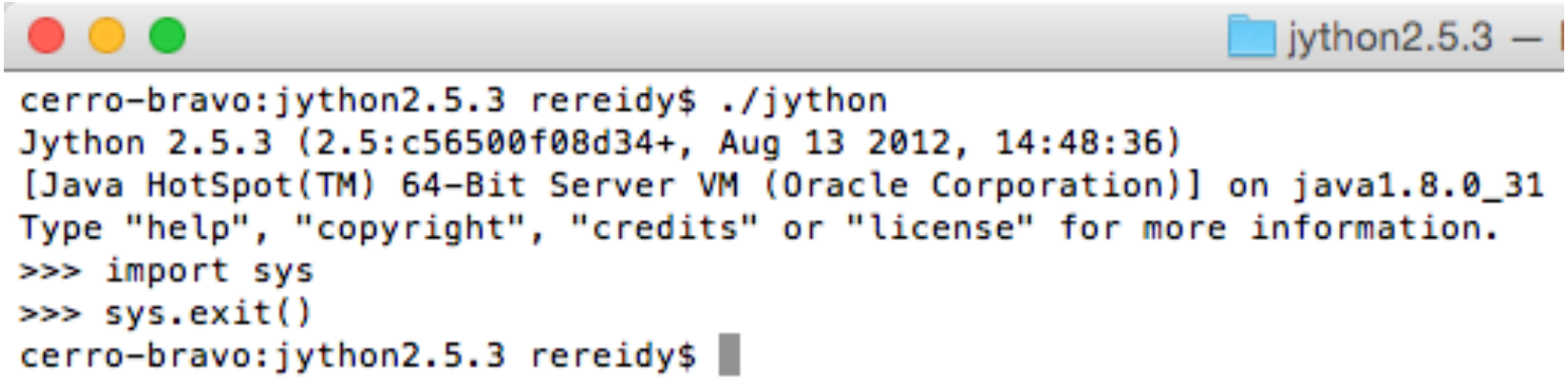
Jython console

- You can enter statements
  - They cannot be saved to a file (like if you were using a GUI console application)
  - There is no Jython GUI console application (jEdit may have this capability)

```
>>> 1 + 1
2
>>> x = 3
>>> x
3
>>> print "Hello Jython!"
Hello Jython!
>>>
```

Exiting the console

- Three ways to exit the console
  - UNIX - ^D
  - Windows - ^Z<CR>
  - Calling sys.exit()

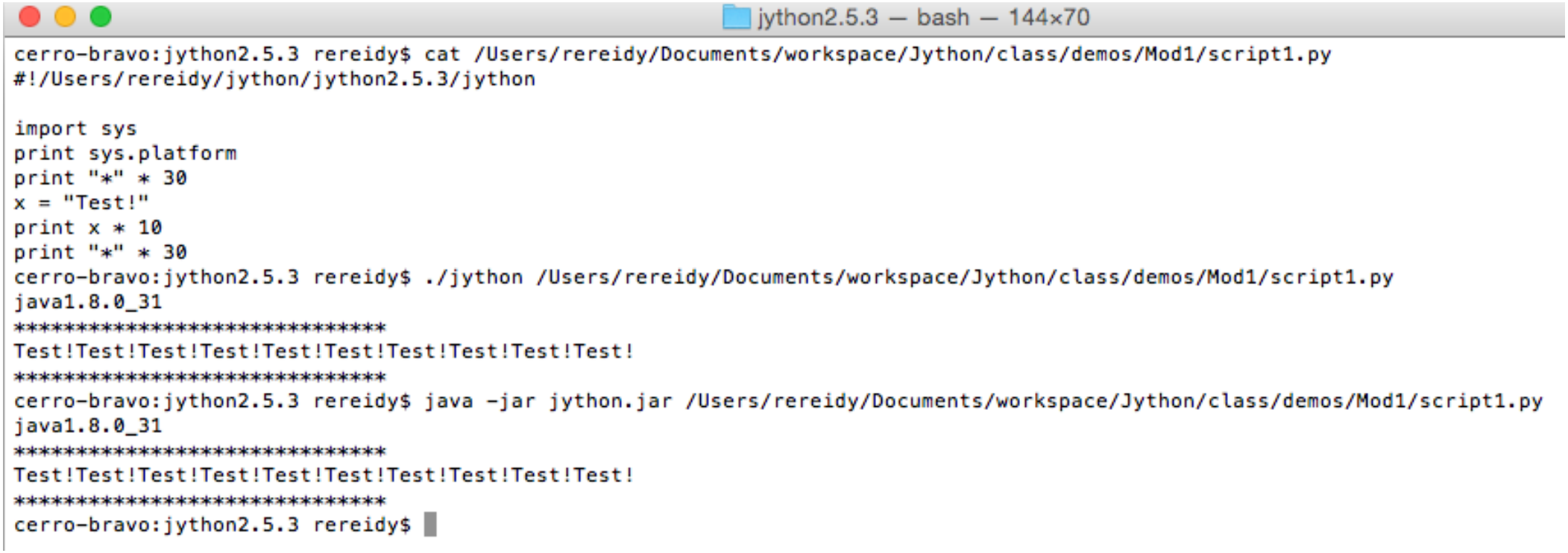


Demonstration of starting Jython

Jython programs

Structure

- Jython programs have an extension of .py
- OS considerations
  - UNIX
    - Use "#!" to identify the location of Jython
  - Windows
    - Use file name extension mapping



Configuration, Environment & Registry

Configure the Jython environment

setuptools

- Module to install or package Python modules
  - Can be used to install Python modules into the Jython environment
  - Modules **must** be pure Python
    - No C/C++ extension code

Jython 2.5

- Copy the script ez_setup.py into the Jython directory
  ./jython ez_setup.py

```
cerro-colorado:jython2.5.3 rereidy$ ./jython ez_setup.py
Downloading http://pypi.python.org/packages/2.5/s/setuptools/setuptools-0.6c11-py2.5.egg
Processing setuptools-0.6c11-py2.5.egg
Copying setuptools-0.6c11-py2.5.egg to /Users/rereidy/jython2.5.3/Lib/site-packages
Adding setuptools 0.6c11 to easy-install.pth file
Installing easy_install script to /Users/rereidy/jython2.5.3/bin
Installing easy_install-2.5 script to /Users/rereidy/jython2.5.3/bin

Installed /Users/rereidy/jython2.5.3/Lib/site-packages/setuptools-0.6c11-py2.5.egg
Processing dependencies for setuptools==0.6c11
Finished processing dependencies for setuptools==0.6c11
```

Jython 2.7

- Comes with *`pip`* installed

Lab: Install setuptools

virtualenv

- virtualenv is a script to create/build Python (Jython) environments
  - Creates installation directories
  - Does not share libraries
- Very good for different versions of Python (Jython)
  - Create a virtual environment for each project
- We will be using this tool to ensure we do not mix environments

Jython 2.5

- Jython 2.5.3
  - Copy virtualenv-1.9.1.tar.gz into the Jython 2.5.3 directory
  - Unpack the file
  - cd into the virtualenv-1.9.1 directory

```
../jython setup.py install
```

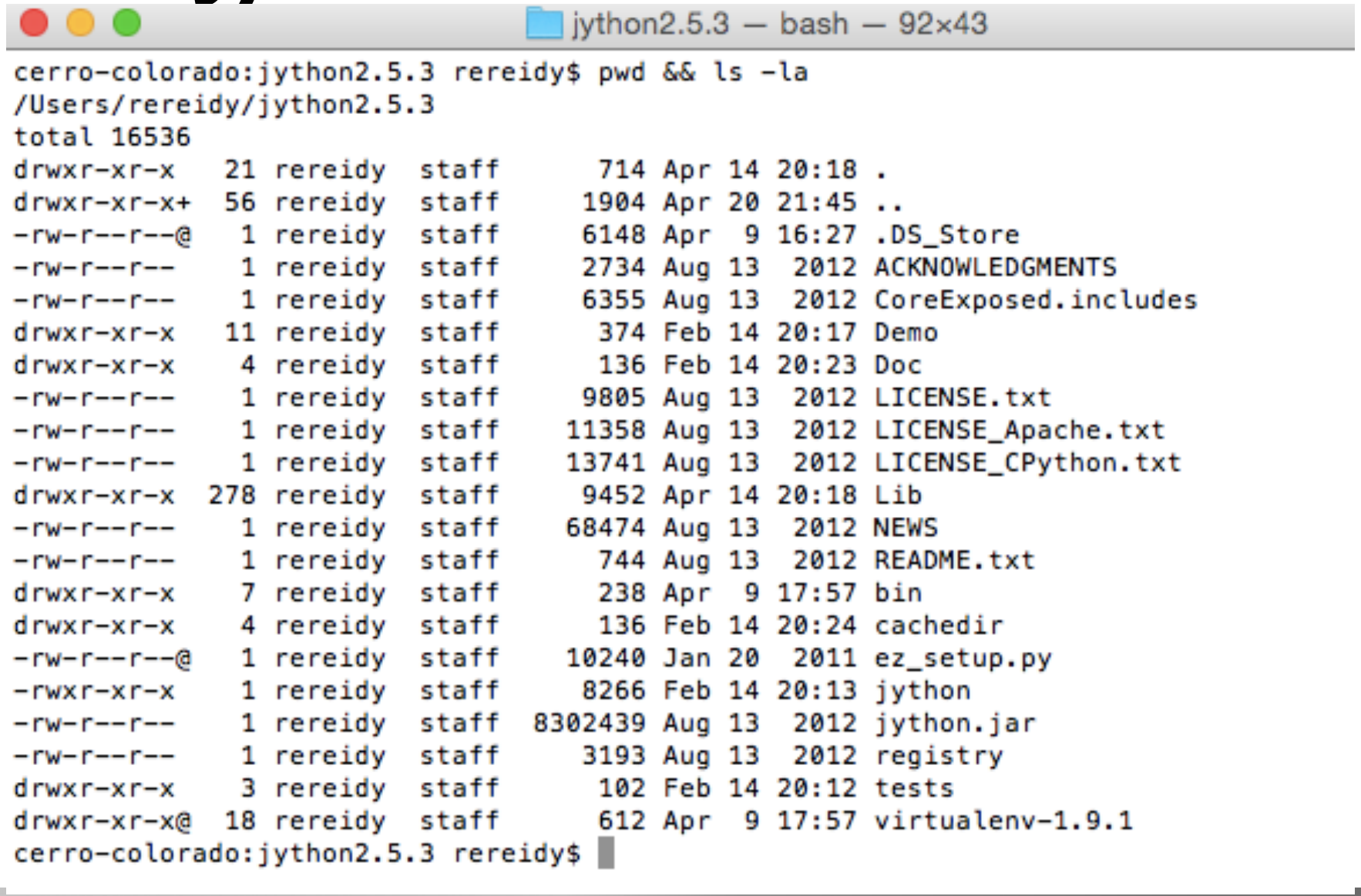
Jython 2.7

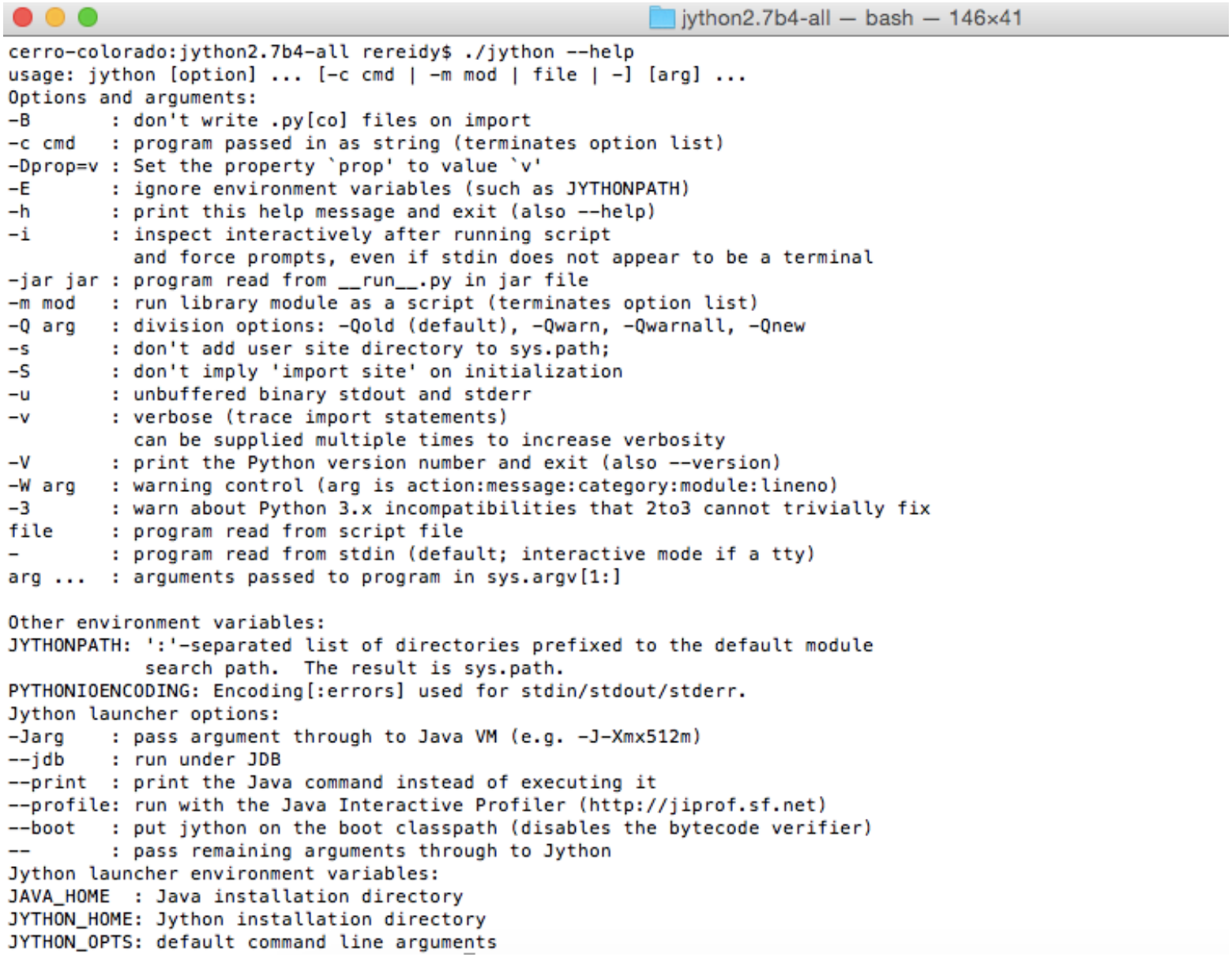
```
bin/pip install virtualenv
```

Lab: Create virtual environments

Examine the Jython installation

Jython installation





Environment

JYTHON_HOME

- The JYTHON_HOME environment variable is the Jython installation directory
  - Good for a system-wide installation
    - We are not using because we have 2 Jython installations

JYTHON_OPTS

- Options to pass to Jython when starting either the interpreter or a program

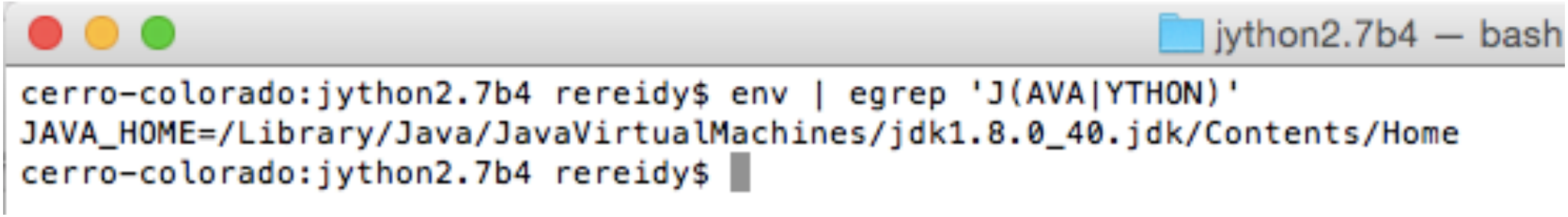
JAVA_HOME

- The JAVA_HOME is the location of Java on your computer

JYTHONPATH

- The path to local scripts and modules
  - Not the path of the system installed modules

Example



Lab

- Set up your Java and Jython environments
- Validate they are correct

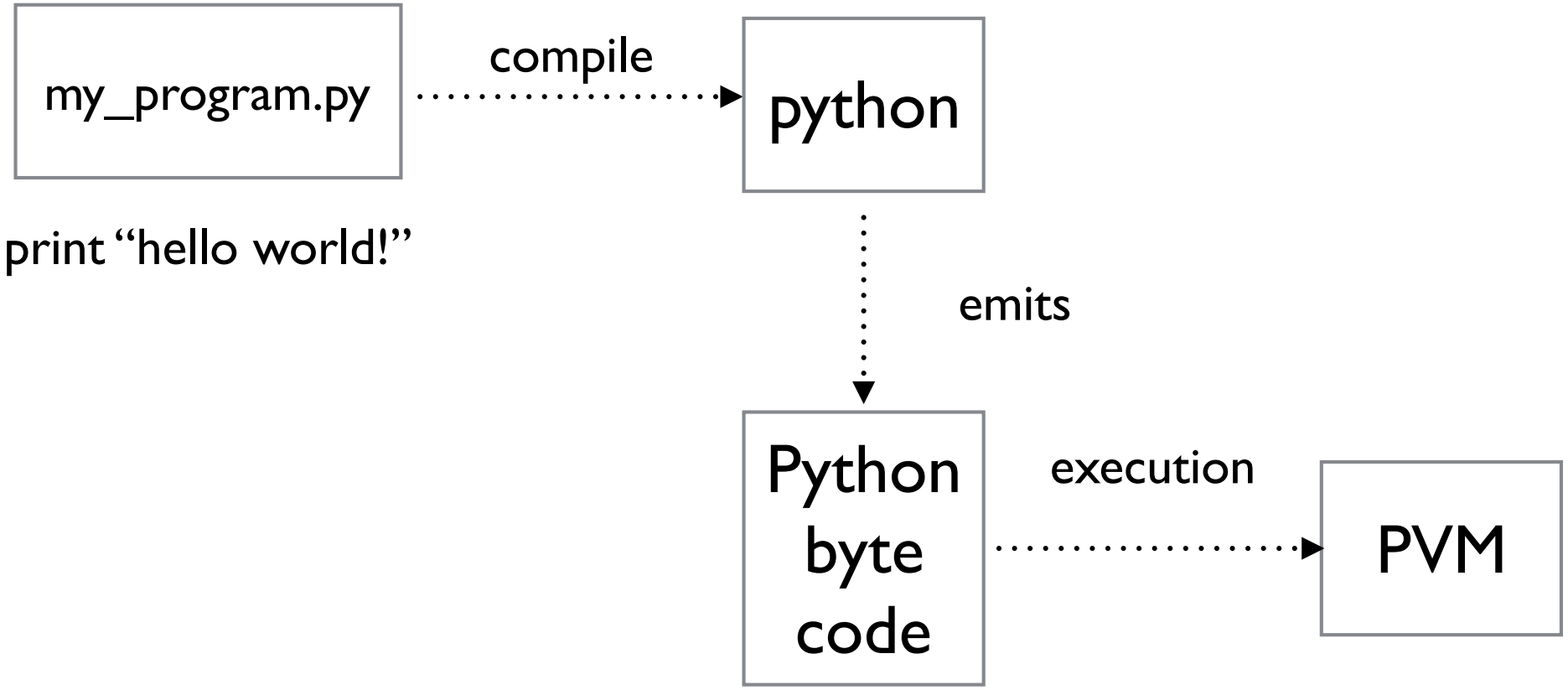
Registry

Jython registry

- Platform independent "Windows registry" for Java namespace
- Jython namespace acquired from
  - Java system properties
  - Jython registry file
  - Account's personal registry file (user.home + "/.jython")
  - Properties specified on the command line when invoking Jython

Execution environment

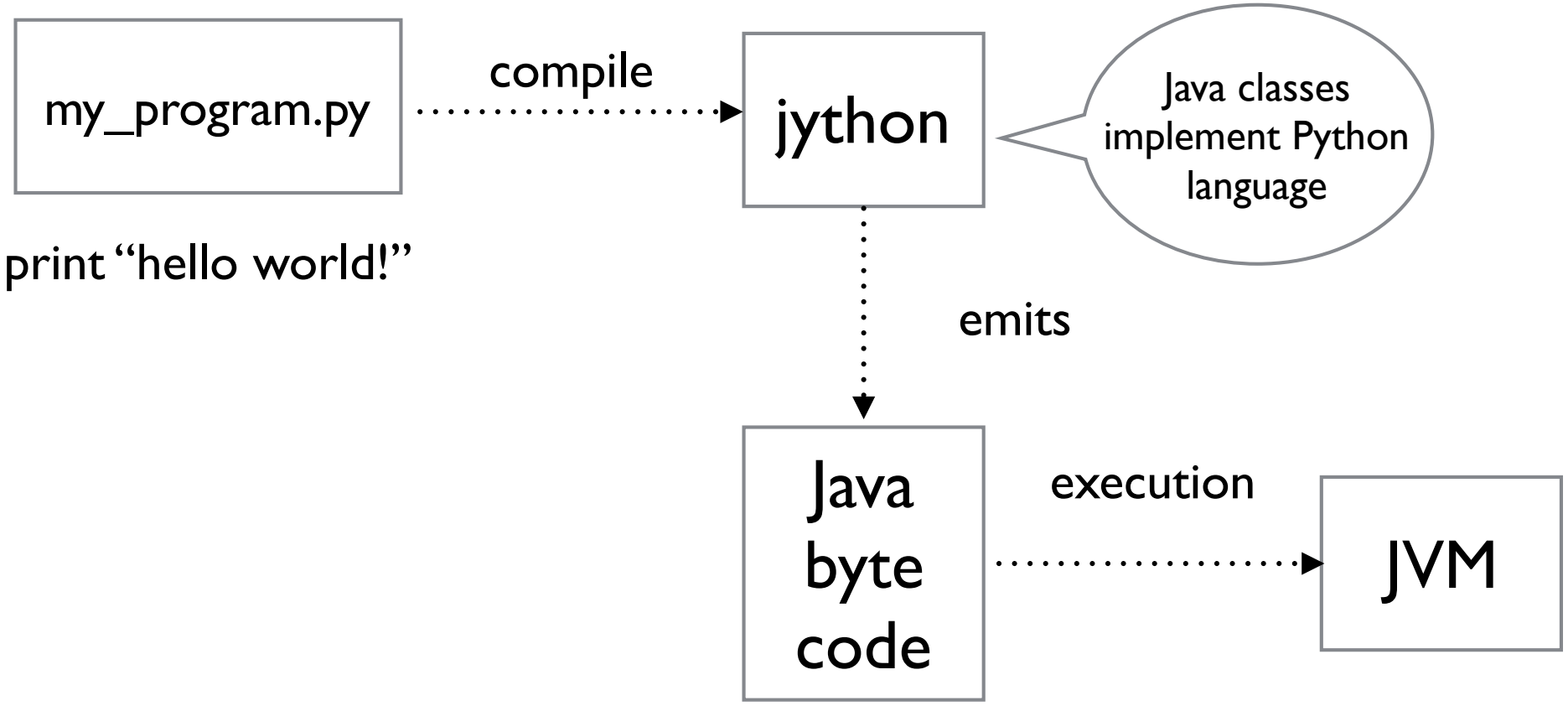
How Python works



Compiled Python code

- The Python interpreter will compile Python source (.pyc - Python compiled)

Jython execution model



Quiz

1. Describe 2 of the design philosophy goals of Python.

A: Code readability
Dynamic typing
Highly extensible
Sparse grammar

Jython

2. Name 3 places where Jython can be used.

A: Web application servers (with servlet containers)

- Web applications
- Build scripting (ant/mavin)
- Console and GUI scripts
- Testing

3. How is extensibility with Jython different than with Python?

A: Jython can be extended with either pure Jython modules or with Java classes; Python can be extended with pure Python modules, or modules written in C/C++.

4. Where does Jython code execute?

A: JVM

5. What are the two methods to start the Jython interpreter?

A. ./jython
   java -jar jython.jar

6. What is the Jython interpreter?

A. A Java JAR file that compiles and executes Python statements and expressions

Jython

7. What is the Jython bytecode?

A. Low-level code executed in the Java JVM

Q & A

Exercises

1. Find your Jython registry file. Identify all entries which begin with the word "python".

   Discuss.

2. Open your editor or IDE and enter the following lines of code:

```
import sys
print sys.platform
print 2 * 100
x = "SPAM!"
print 8 * x
```

Execute the script.