Jython

Introduction to Jython programming

Agenda

Topics

- Using Java within Jython
- Using Jython within Java
- Java/Jython programs
- Compile and deploy Jython
- Quiz
- Q & A

Data type mapping

Java Type	Python Type
char	String(length of 1)
boolean	Integer(true = not zero)
byte, short, int, long	Integer
java.lang.String, byte[], char[]	String
java.lang.Class	JavaClass
Foo[]	Array(containing objects of class or subclass of Foo)
java.lang.Object	String
orb.python.core.PyObject	Unchanged
Foo	JavaInstance representing Java class Foo

Basic data conversions

- Each allowed conversion into a primitive Java type converts into a primitive type wrapper class
  - If a Python type can be converted into a Java int
    - Wrapped as a java.lang.Integer
  - Python None type converted to null
  - Python conversion to Java boolean
    - Python needs to be an integer value of 0 (False); anything else is True
  - Python to java.lang.Object
    - Convert to wrapping in previous slide for precedence
      - Integer is preferred to Short, Double to Float, etc.
      - You can create your own conversion by calling the Java constructor

Convert arrays

- When calling a Java method that requires an array type
  - Python lists and tuples
    - Attempt to coerce the object into component type of the Java array (recursive) - see the slide back 2
      - Can lead to type exceptions

Module: jarray

- Allows creation of Java array type directly
  - Methods
    - zeros(length, type) - create a Java array of a type and a length
    - array(seq, type) - create a Java array of a type and initialize it with elements of a sequence (coerced into the proper type)

jarray type strings

Type string	Java primitive type
“l”	long
“i"	integer
“h"	short
“c"	char
“b"	byte
“d"	double
“f"	float
“z"	boolean

Java to Python conversion

Java type	Python type
java.lang.Long, long	Long
java.lang.Integer, int	Integer
java.lang.Short, short	Integer
java.lang.Byte, byte	Integer
java.lang.Boolean, boolean	Integer (0 maps to False; nonzero maps to True)
java.lang.Double, double	Float
java.lang.Float, float	Float
java.lang.Character, char	String length 1
java.lang.String	String
Java class instance	Wrapped Java class
null	None
Java array	Wrapped Java array

Does not apply to constructor methods (always produces a wrapped instance)

Collection wrappers

- No automatic conversion
  - Python dictionaries to Java mapping classes (java.util.HashMap, java.util.Hashtable, etc.)
  - Python sequences and list classes
- Copy conversion
- Iteration support for
  - java.util.Vector (seen earlier)
  - java.util.Ennumeration
  - java.util.List
  - java.util.Iterator

Using Java within Jython

Using Java classes_I

- Construct Java class instances the same way as Jython
  - Do not use the Java "new" keyword
- When a Java instance is created
  - Class *org.python.core.PyJavaInstance*
    - Transparent wrapper of the Java instance
    - Automates mapping between Python and Java data and naming conventions

Using Java classes$_2$

- Once a Java instance is instantiated
  - Use just like a Python object
    - get, set, delete attributes
    - Pass methods as function and class arguments (first class objects)

Automatic type conversion

- Jython performs automatic conversion at runtime
- Allow Java code to be used easily in Jython
  - Python built-in types to Java types
  - Java proxy types unwrapped to allow Java class passing during method call
    - Cannot use keyword arguments when calling a Java method from Jython

Overloading

- Data conversion and selection of data types happens at the same time
  - Overloaded methods have different argument numbers
    - Jython dispatch based on number of arguments

Java type	Python types and Java wrapped types
long	Integer, long, java.lang.Long
int	Integer, long, java.lang.Integer
short	Integer, long, java.lang.Short
char	String (length must be 1), java.lang.Character
byte	Integer, long, java.lang.Byte
double	Integer, long, float, java.lang.Double
float	Integer, long, float, java.lang.Float
boolean	Integer: 0 maps to False, nonzero maps to True, java.lang.Boolean
java.lang.String	String
array (long, short, etc.)	Gerneic array rules
SubFoo (subclass of Foo)	SubFoo, Foo
Foo	Foo
java.lang.Class	Wrapped Java classes or Python classes subclassing a Java class
java.lang.Integer	Integer, long, java.lang.Integer
java.math.BigInteger	Long
java.lang.Number	Integer (as java.lang.Integer), long (as java.math.BigInteger), float (as java.lang.Double)
java.io.Serializable	Similar to java.lang.Object
java.lang.Object	Java wrapped instances/arrays or instances of Python classes subclassed as Java class; integer, long, and float as in java.lang.Number; String (as java.lang.String)

Source: Jython Essentials

Inheriting from Java

Inheriting from Java

- Jython support direct inheritance from Java
- Jython creates a "proxy" class in Java and associates it with the Python class
- You cannot set arbitrary attributes to a Java class
  - You can do this through a Python class that inherits the Java class.

Calling super methods and constructors

- Overridden Java public methods invokes the same way as in Python

Java interoperability and overriding

- Each method in a Jython child class that overrides a base class or interface method

Reflection

Basics

- Ability of objects to find out about structure of other objects (and themselves)
  - List all attributes
  - Invoke method from name
- Managed by *java.lang.reflect*
  - Defines objects to encapsulate methods and attributes

Bean-based reflection

- JavaBeans expose properties that can be discovered using reflection

Using Jython within Java

Adding Jython to a Java application

- A class better implemented in Python or some really great Jython code that can be useful

Java/Jython programs

Command line programs

- We have already seen some command line programs
- Command line argument parsing
  - getopt
  - argparse
- Configuration parsing
  - ConfigParser

Command line argument parsing

Positional - sys.argv

- All example code to this point has used the arguments found
- This means you have to write code to test each argument is correct by position

Module: getopt

- Similar to the C library getopt
- Arguments are not positional
    - Identified by a “switch”

Module: argparse

- Not available in Jython 2.5
- Versatile
  - Positional args
  - Optional arguments
  - Short and long versions of args
  - Combine positional and optional args

Java GUI programs

Swing programs

- Implemented in Java swing
- Top level containers

Container type	Class Hierarchy
JApplet	java.lang.Object, java.awt.Component, java.awt.Container, java.awt.Panel, java.applet.Applet, javax.swing.JApplet
JDialog	java.lang.Object, java.awt.Component, java.awt.Container, java.awt.Window, java.awt.Dialog, javax.swing.JDialog
JFrame	java.lang.Object, java.awt.Component, java.awt.Container, java.awt.Window, java.awt.Frame, javax.swing.JFrame
JWindow	java.lang.Object, java.awt.Component, java.awt.Container, java.awt.Window, javax.swing.JWindow

- **Beware of deprecated methods and functionality**

Compile and deploy Jython

Jython exists on target host

Server setup

- The Jython installation directory should be free of application code (obviously)
- The CLASSPATH variable must include the location of the Jython JAR file

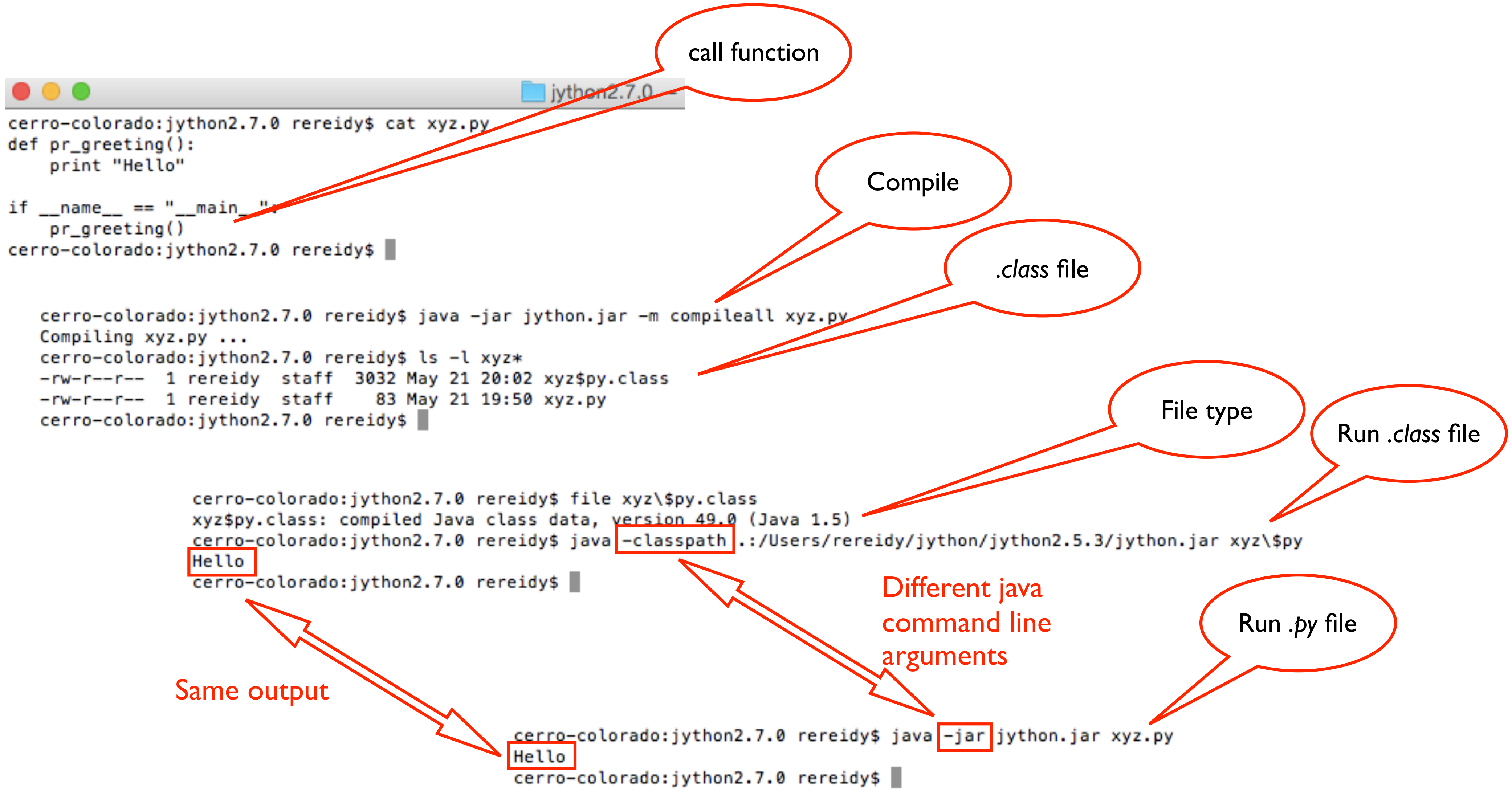
Code deployment

- Two methods
  - Copy source code to the host server
  - Package the code into a JAR file and deploy

Compile

- Compiler existed in early versions of Jython
  - Last available in Jython 2.1
  - Not in the current versions (2.5 and 2.7)
- Simplified by the methods described in Module 5 - Functions, debugging, modules, and packages - Compile to bytecode

Recap of Jython compile to bytecode



```
cerro-colorado:jython2.7.0 rereidy$ cat xyz.py
def pr_greeting():
    print "Hello"

if __name__ == "__main__":
    pr_greeting()
cerro-colorado:jython2.7.0 rereidy$
```

```
cerro-colorado:jython2.7.0 rereidy$ java -jar jython.jar -m compileall xyz.py
Compiling xyz.py ...
cerro-colorado:jython2.7.0 rereidy$ ls -l xyz*
-rw-r--r--  1 rereidy  staff  3032 May 21 20:02 xyz$py.class
-rw-r--r--  1 rereidy  staff    83 May 21 19:50 xyz.py
cerro-colorado:jython2.7.0 rereidy$
```

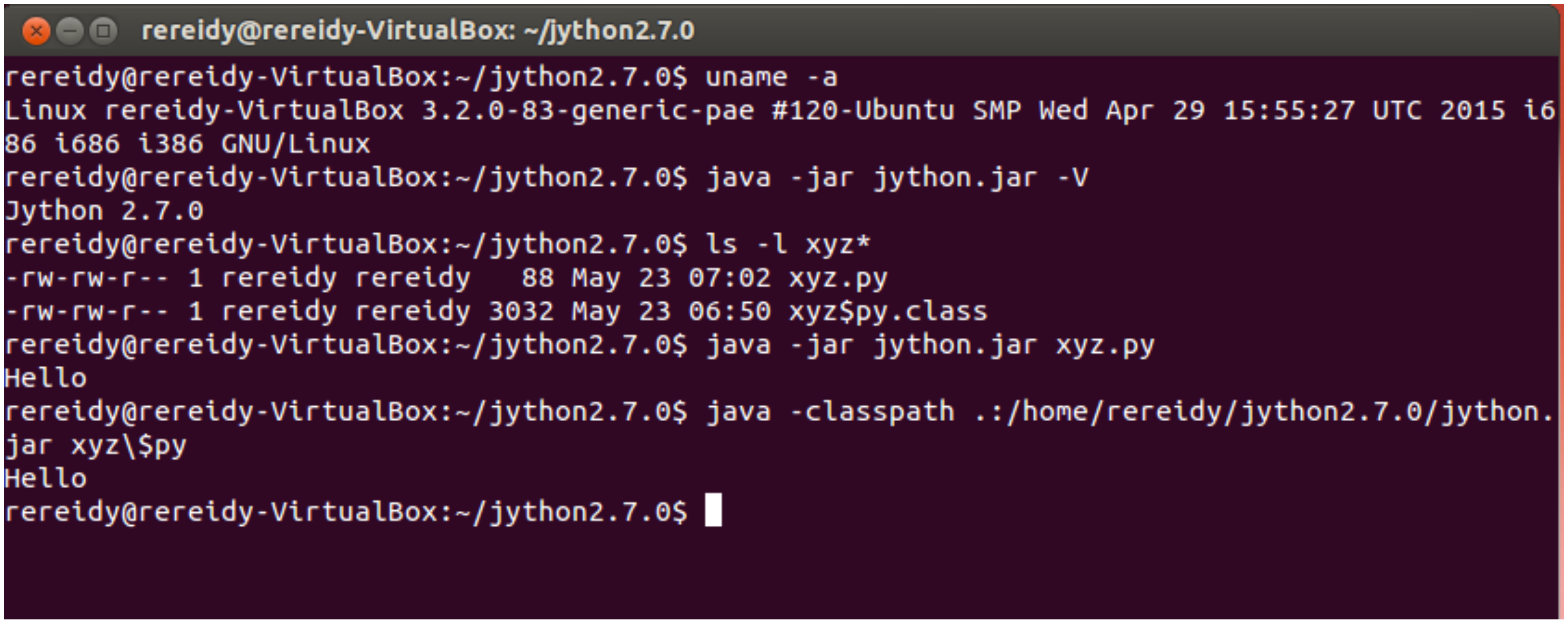
```
cerro-colorado:jython2.7.0 rereidy$ file xyz\$py.class
xyz$py.class: compiled Java class data, version 49.0 (Java 1.5)
cerro-colorado:jython2.7.0 rereidy$ java -classpath .:/Users/rereidy/jython/jython2.5.3/jython.jar xyz\$py
Hello
cerro-colorado:jython2.7.0 rereidy$
```

```
cerro-colorado:jython2.7.0 rereidy$ java -jar jython.jar xyz.py
Hello
cerro-colorado:jython2.7.0 rereidy$
```

Distribute application

- Copy Jython source to host server
- Copy compiled Jython *$py.class* files to server
  - Java class files are portable bytecode

Code developed on Mac OS X Deployed to Ubuntu Linux



```
rereidy@rereidy-VirtualBox:~/jython2.7.0$ uname -a
Linux rereidy-VirtualBox 3.2.0-83-generic-pae #120-Ubuntu SMP Wed Apr 29 15:55:27 UTC 2015 i6
86 i686 i386 GNU/Linux
rereidy@rereidy-VirtualBox:~/jython2.7.0$ java -jar jython.jar -V
Jython 2.7.0
rereidy@rereidy-VirtualBox:~/jython2.7.0$ ls -l xyz*
-rw-rw-r-- 1 rereidy rereidy   88 May 23 07:02 xyz.py
-rw-rw-r-- 1 rereidy rereidy 3032 May 23 06:50 xyz$py.class
rereidy@rereidy-VirtualBox:~/jython2.7.0$ java -jar jython.jar xyz.py
Hello
rereidy@rereidy-VirtualBox:~/jython2.7.0$ java -classpath .:/home/rereidy/jython2.7.0/jython.
jar xyz\$py
Hello
rereidy@rereidy-VirtualBox:~/jython2.7.0$ 
```

Quiz

1. What doe the following Java data types equate to in Python?

   a. char
   b. byte[] (byte array)
   c. java.lang.Object

A. A char is a Python String of length 1; a byte array (byte[]) is a Python String; a java.lang.Object is a String

Q & A

Exercises

1. Create a program to convert Java primitive types (integer, long, double, float, string) to Python types.

   Use the isinstance() method to validate the conversion has been performed.

2. Create a class which accepts a Python string in its constructor. The class should convert the Python string to a Java string and return it using a separate method (e.g. convert()).

3. Create a Jython module file and compile it.

   Verify the resulting file is a Java class and run it as a Java class.