

Using Jython to Prototype and Extend Java-based Systems

Dale Parson, Dylan Schwesinger
and Thea Steele, Kutztown University

PACISE 2011

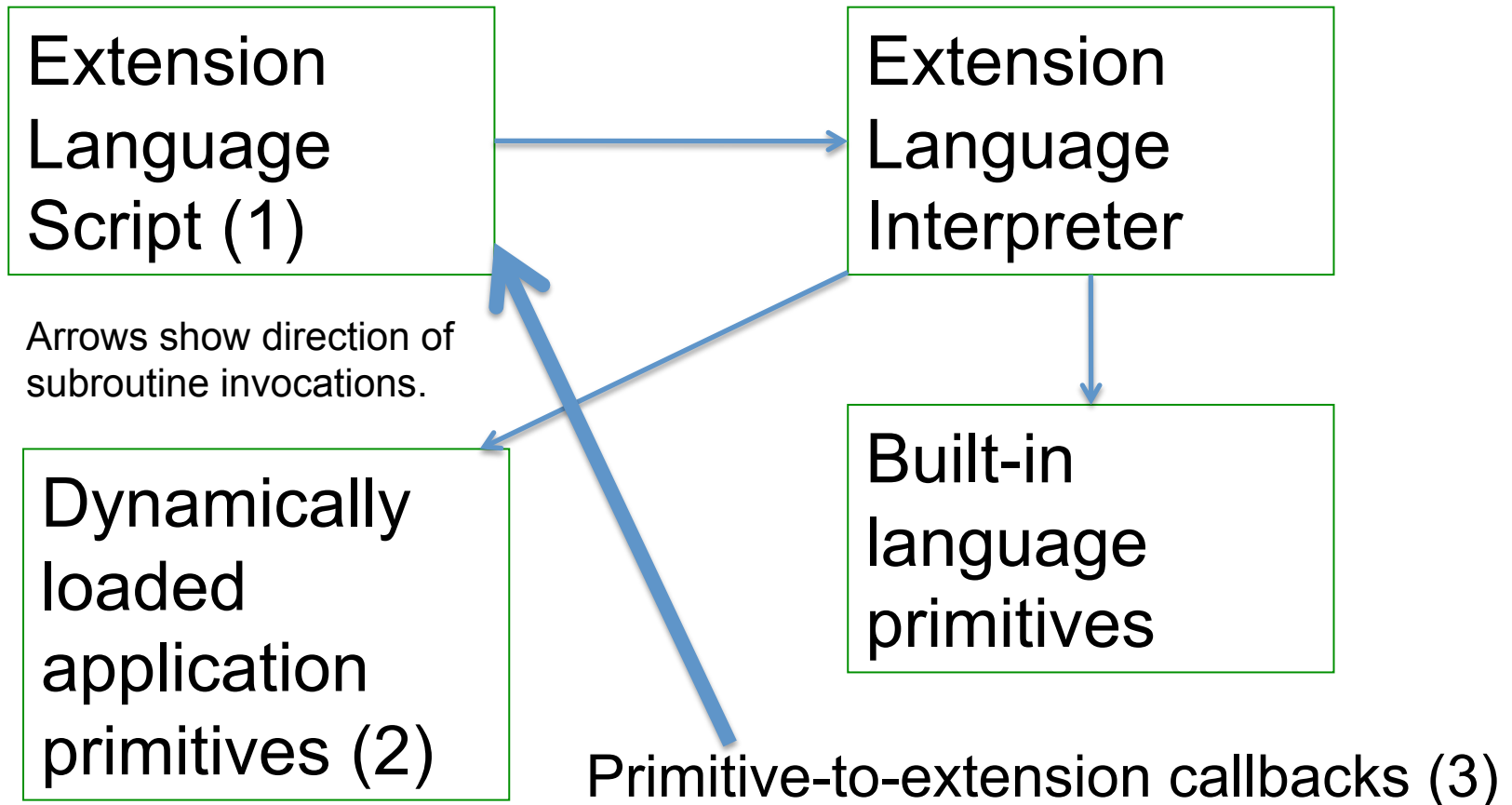
Python and Jython

- Jython is an open source implementation of an interpreter for Python. Python:
 - has classes, inheritance and polymorphism, making it a good candidate for specifying and prototyping **object-oriented** Java systems and components.
 - has source-level generic container types (sequences, sets and maps), functional programming constructs (first-class functions, closures, generators, higher order functions), and run-time interpretation, making it a good candidate for **rapid prototyping**.

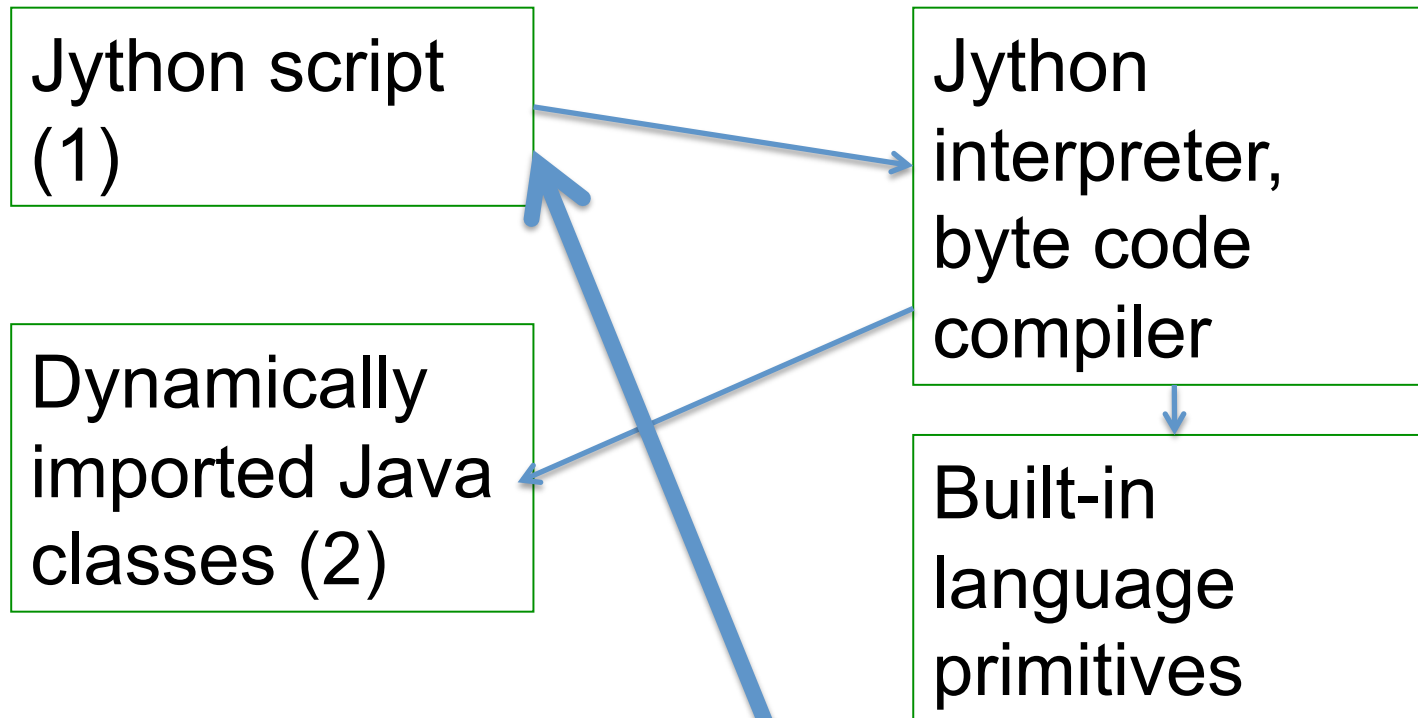
Jython and Java

- Jython is written in Java to compile Python source to byte code for the Java Virtual Machine (JVM).
- Jython can **import** any compiled Java class and use its objects as Python objects.
 - Jython can use the **substantial Java library** code base without any need to write wrapper code.
- Jython also comes with a substantial native Python library similar to C-based Python.
- A Java application can use a Jython interpreter as a Java object, requiring very little wrapper code.

Three Extension Language Extension Mechanisms



Three Jython Extension Mechanisms

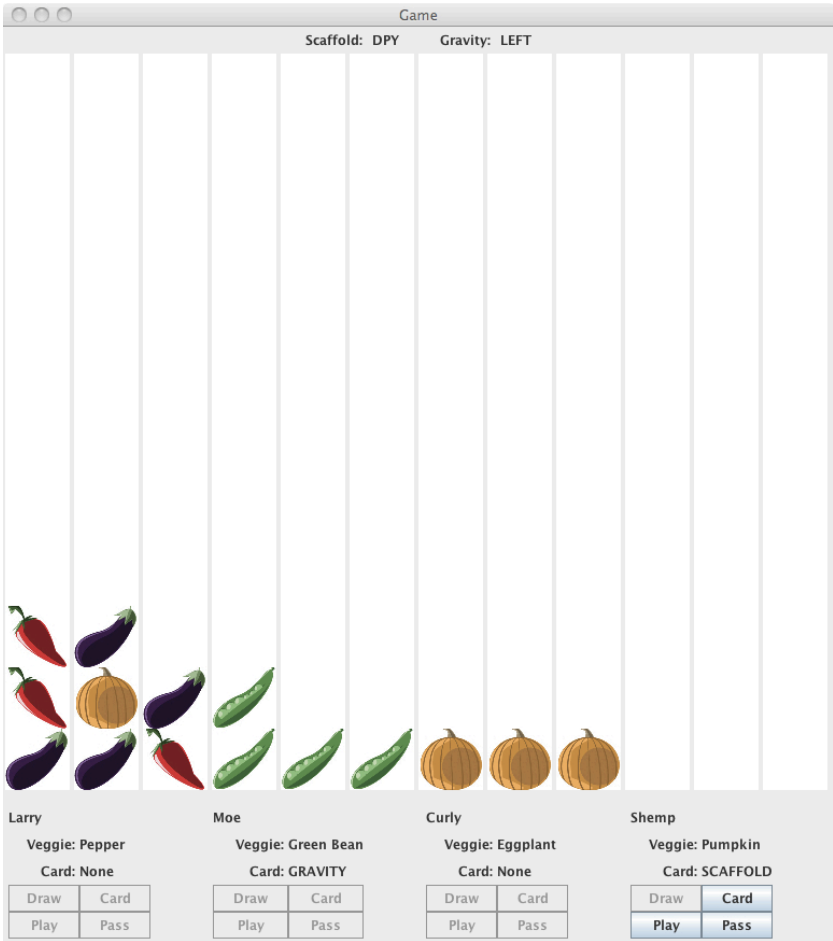


Java-to-Jython callbacks (3), e.g., Jython listeners for Java-generated events.

Jython-Java extension mechanisms

- Jython can import and use Java classes, including reflection and multithreading.
- Jython can implement Java interfaces, including event listener interfaces.
 - Interface inheritance and polymorphism.
- Jython can extend Java classes.
 - Implementation inheritance and delegation.
- Jython interpreter can work as a Java object.

A Java graphical game, all in Jython



Building the game

- Jython interpreter allowed us to build and play incrementally – it is **working pseudocode**.
 - Physical prototype was not necessary.
 - We avoided many detailed issues of Java implementation while exploring the game design space.
- Code base is compact and readable.
- We could use the full Java Swing library.
 - Others libraries such as SWT are equally accessible.
- GUI code is more concise than Java code.

Game-building pitfalls.

- Dynamic typing defers type error detection until run time.
 - Exhaustive testing is essential to catch type errors.
- Translation of Java types to Python types occurs when there is a Python counterpart.
 - For example, a Java String object becomes a Python string, requiring Python string functions, not `java.lang.String` methods. This translation does not happen for Java classes with no Python counterparts.
- Interpreted performance no problem in this project.

Jython-Java Musical Keyboard

Just Intonation MIDI Keyboard, D. Parson, NYE 2010-2011

channel	code	x0..x3	freq	l	b0,k1	b1,k5	b2,kb7	b3,ko	o	vel	vol	p	b	i	t	s
1	code	<input checked="" type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>	d 293.66	<input type="checkbox"/> latch	1	5	b7	o	-1	64	64	64	b...	4	newroot	IA...
2	code	<input checked="" type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>	d 293.66	<input checked="" type="checkbox"/> latch	1	5	b7	o	-1	64	64	64	b...	4	newroot	IA...
3	code	<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>	d 293.66	<input type="checkbox"/> latch	1	5	b7	o	-1	64	64	64	b...	4	newroot	IA...

Editor for channel 1

```
def genfunc(s, sc, ch):
    import random
    def r(lower, upper):
        return random.randint(lower,upper)

    dur = 1.0
    de = 16
    while True:
        s.b2 = dur
        yield de
        if r(0,3) == 0:
            s.b3 = dur
            yield de

        s.b1 = dur
        yield de
        s.b0 = dur
        yield de
```

Compile Cancel Silence

16 code d 293.66 latch 1 5 b7 o -1 64 64 64 b... 4 newroot IA...

Just Intonation Keyboard

- One row per MIDI (Musical Instrument Digital Interface) channel (voice), up to 16.
 - javax.sound.midi sound library used by Jython.
 - Hooks to external software & hardware synths as well.
- Buttons for keys, spinners for octave and other parameters, check&combo boxes from Swing.
- Construction of translation tables for non-standard scales uses Python functional programming.

Live Coding!

- Live coding is the accepted practice of coding-as-performance by laptop musicians.
- Jython supports use of a stylized form of Python programming to improvise synthesized music.
- Python support for extensible classes, reflection, and source code interpretation (eval, exec and compile) allow using Python as a domain-specific language.
- Each graphical control becomes a symbol available for manipulation as an rvalue or an lvalue in a Python expression. Swing timers used for tempo & meter.

Jython Constructs for Live Coding

- A Jython subclass for each Java GUI control class adds a common getter/setter method pair to each. Getting / setting becomes orthogonal to the type of control.
- A Python property allows each orthogonal getter to be manipulated as an rvalue, and each setter as an lvalue, in a live code Python expression.
- Python supports addition of fields and methods to classes and individual objects after construction. Incremental construction of live coding fields, methods and properties occurs in tandem with associated, incremental GUI construction.
- Python's *exec* function allows live code to be compiled.

Work avoided in live coding

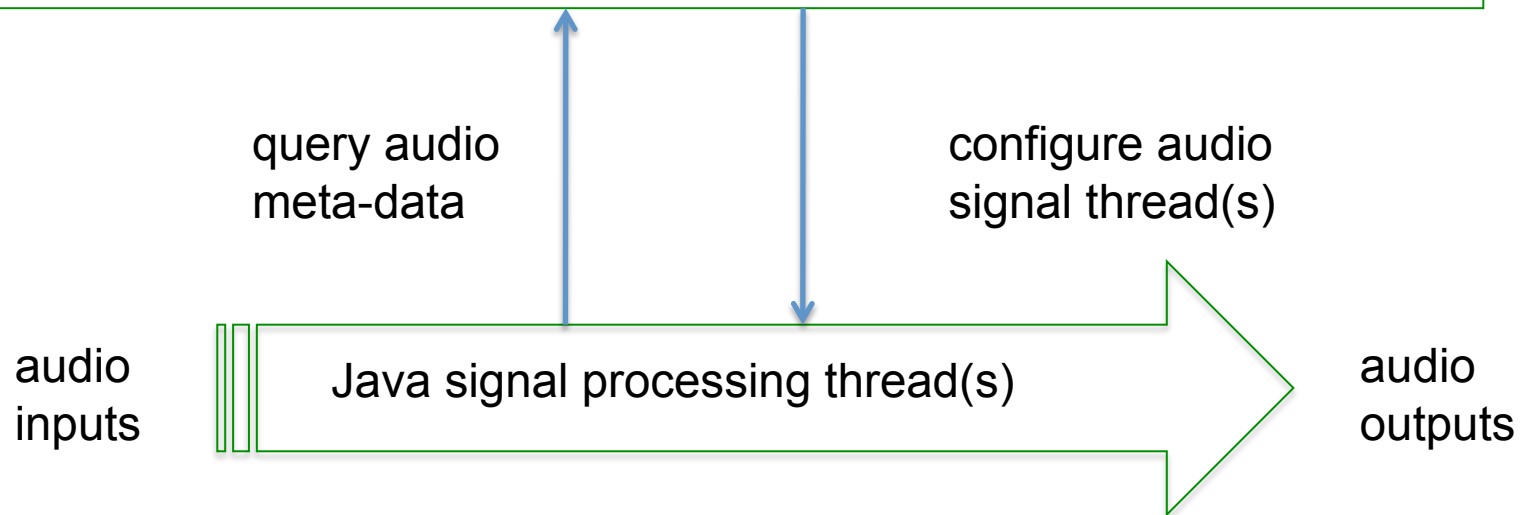
- A language such as Java would require designing a special purpose language,
- and using tools such as scanner and parser generators to compile live code to an intermediate form (VM code),
- and providing run time support for the live code VM.
- Jython eliminates the need for designing a custom language and its support tools. The domain-specific language is an incremental extension of Python.

Jython, Java and C++ performance

- Jython is considerable slow than Java, and performance can actually degrade in multithreaded Jython on a massive multiprocessor, but
- Java can run as fast or faster than optimized C++ on modern processors and multiprocessors.
 - See benchmarks reported in the paper.
- Conclusion is the Java is now fast enough to handle many time-constrained tasks (e.g., audio data flow), making Jython attractive in a two-tiered software architecture.

Two-tiered Architecture

Jython user interface and configuration with synchronized interaction with Java signal processing threads



Conclusions

- Python's object-oriented features make it a good fit for object-oriented modeling, and its functional language features and run-time compilation make it good for compact prototyping.
- Jython's seamless integration into Java make it ideal for incremental prototyping and extension of Java systems while leveraging Java & Python libraries.
- Python extension mechanisms make it ideal for incremental construction of domain languages.