

Scripting vs Systems Programming Languages

- | | |
|--|--|
| <ul style="list-style-type: none">◆ Designed for gluing applications : flexibility◆ Interpreted◆ Dynamic variable creation◆ Data and code integrated : meta-programming supported◆ Dynamic typing (typeless)◆ <i>Examples: PERL, Tcl, Python, Ruby, Scheme, Visual Basic, etc</i> | <ul style="list-style-type: none">◆ Designed for building applications : efficiency◆ Compiled◆ Variable declaration◆ Data and code separated : cannot create/run code on the fly◆ Static typing◆ <i>Examples: PL/1, Ada, Java, C/C++, C#, etc</i> |
|--|--|

cs480 (Prasad)

LSysVsScript

1

(cont'd)

- ◆ Does application implement complex algorithms and data structures?
- ◆ Does application process large data sets (>10,000 items)?
- ◆ Are application functions well-defined, fixed?
If yes, consider a system programming language.
- ◆ Is the main task to connect components, legacy apps?
- ◆ Does the application manipulate a variety of things?
- ◆ Does the application have a GUI?
- ◆ Are the application's functions evolving rapidly?
- ◆ Must the application be extensible?
- ◆ Does the application do a lot of string manipulation?
If yes, consider a scripting language.

cs480 (Prasad)

LSysVsScript

2

Current Trends

- ◆ Hybrid Languages : Scripting + Systems Programming
 - Recent JVM-based Scripting Languages
 - » **Jython** : Python dialect
 - » **Clojure** : LISP dialect
 - » **Scala** : OOP +Functional Hybrid

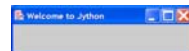
cs480 (Prasad)

LSysVsScript

3

Jython (for convenient access to Java APIs)

```
I:\tkprasad\cs480>jython
Jython 2.1 on java1.4.1_02 (JIT: null)
Type "copyright", "credits" or "license" for more information.
>>> import javax.swing as swing
>>> win = swing.JFrame("Welcome to Jython")
>>> win.size = (200, 200)
>>> win.show()
>>> ^Z
```



cs480 (Prasad)

LSysVsScript

4

Java vs Jython

```
map = new HashMap();
map.put("one", new Integer(1));
map.put("two", new Integer(2));
map.put("three", new Integer(3));
```

```
System.out.println(map.get("one"));
```

```
list = new LinkedList();
list.add(new Integer(1));
list.add(new Integer(2));
list.add(new Integer(3));
```

```
map =
{"one":1,"two":2,"three":3}
```

```
print map ["one"]
```

```
list = [1, 2, 3]
```

cs480 (Prasad)

LSysVsScript

5

(cont'd)

```
for (Iterator i; i.hasNext();)
{
    i.next();
}

List newList = ArrayList()
for (Iterator i; i.hasNext();)
{
    Object obj = i.next();
    newList.add(function(obj))
}
```

```
for i in list:
```

(* iterator *)

```
newList =
[function(i) for i in oldList]
```

(* list comprehension *)

cs480 (Prasad)

LSysVsScript

6

Defining functions and expressions in Jython (file: eg.py)

```
def fac(x):
    "factorial function"
    if x <= 1: return 1
    return long(x)*fac(x-1)
```

```
fac(5)
# 120L
print fac(5), fac(20)
print fac.__doc__
#factorial function
```

```
X = 2 + 3j
x.imag + x.conjugate()
# (5 - 3j)
```

```
list =
[pow(i,i) for i in [1,2]]
```

```
from math import *
print "PI=%f, e=%f" % (pi,e)
```

```
Y = range(1,9,1)
#[1,2,3,4,5,6,7,8]
Y[1:3]
#[2, 3]
Y[::2]
#[1, 3, 5, 7]
```

cs480 (Prasad)

LSysVsScript

7

More Jython code examples

◆ Dynamic code evaluation

```
print eval("[1,3] + range(6,10,3)")
```

```
# [1, 3, 6, 9]
```

```
x = 2 + 3j
```

```
exec "x = 5, x + x"
```

```
#(5, (4+6j))
```

cs480 (Prasad)

LSysVsScript

8

Cont'd

◆ Exceptions

```
def readfile (name):
    "return lines in file or None if unreadable"
    try:
        file = open(name, 'r')
        try: # raise IOError
            return file.readlines()
        finally:
            file.close()
    except IOError, ioe:
        print "Exception -", ioe
```

cs480 (Prasad)

LSysVsScript

9

Cont'd

◆ Functional Programming

```
apply(lambda x,y : x*y, (10, 20))
# 200
map(lambda x,y: x + y, [[1,2],[1,"a"]], [[1,"3"],[1,"b"]])
# [[1, 2, '3'], [1, 'b']]
reduce(lambda x,y: x + y, [1,2,3], 100)
# 106
filter(lambda x: x > 0, range(10,-5,-3))
# [10, 7, 4, 1]
```

cs480 (Prasad)

LSysVsScript

10

Java functionality through Jython

```
import java.lang as lang
import javax.swing as swing
import java.awt as awt

names = ["Groucho", "Chico", "Harpo"]
quotes = {"Groucho": "Say the secret word", "Chico": "Viaduct?",
         "Harpo": "HONK!"}

def buttonPressed(event):
    field.text = quotes[event.source.text]

def exit(event):
    lang.System.exit(0)
def createButton(name):
    return swing.JButton(name, preferredSize=(100,20),
        actionPerformed=buttonPressed)
```

cs480 (Prasad)

LSysVsScript

11

```
win = swing.JFrame("Welcome to Jython", size=(200,
200),windowClosing=exit)
```

```
win.contentPane.layout = awt.FlowLayout( )
field = swing.JTextField(preferredSize=(200,20))
win.contentPane.add(field)

buttons = [createButton(each) for each in names]

for eachButton in buttons:
    win.contentPane.add(eachButton)

win.pack( )
win.show( )
```



cs480 (Prasad)

LSysVsScript

12