

python.

a presentation to the Rice University CS Club

November 30, 2006

Daniel Sandler

<http://www.cs.rice.edu/~dsandler/python/>

1.

2.

3.

4.

1.

A young mind is corrupted.

Me, circa 2000



(very busy)

Many tasks, many languages

C++

(work)

Scheme

(fun)

perl

(utility belt)

OOP,
popular

pretty,
powerful

useful

Many tasks, many languages

C++

(work)

OOP,
popular

painful
development
cycle

Scheme

(fun)

pretty,
powerful

unreadable
by others

perl

(utility belt)

useful

unreadable
by *anyone*

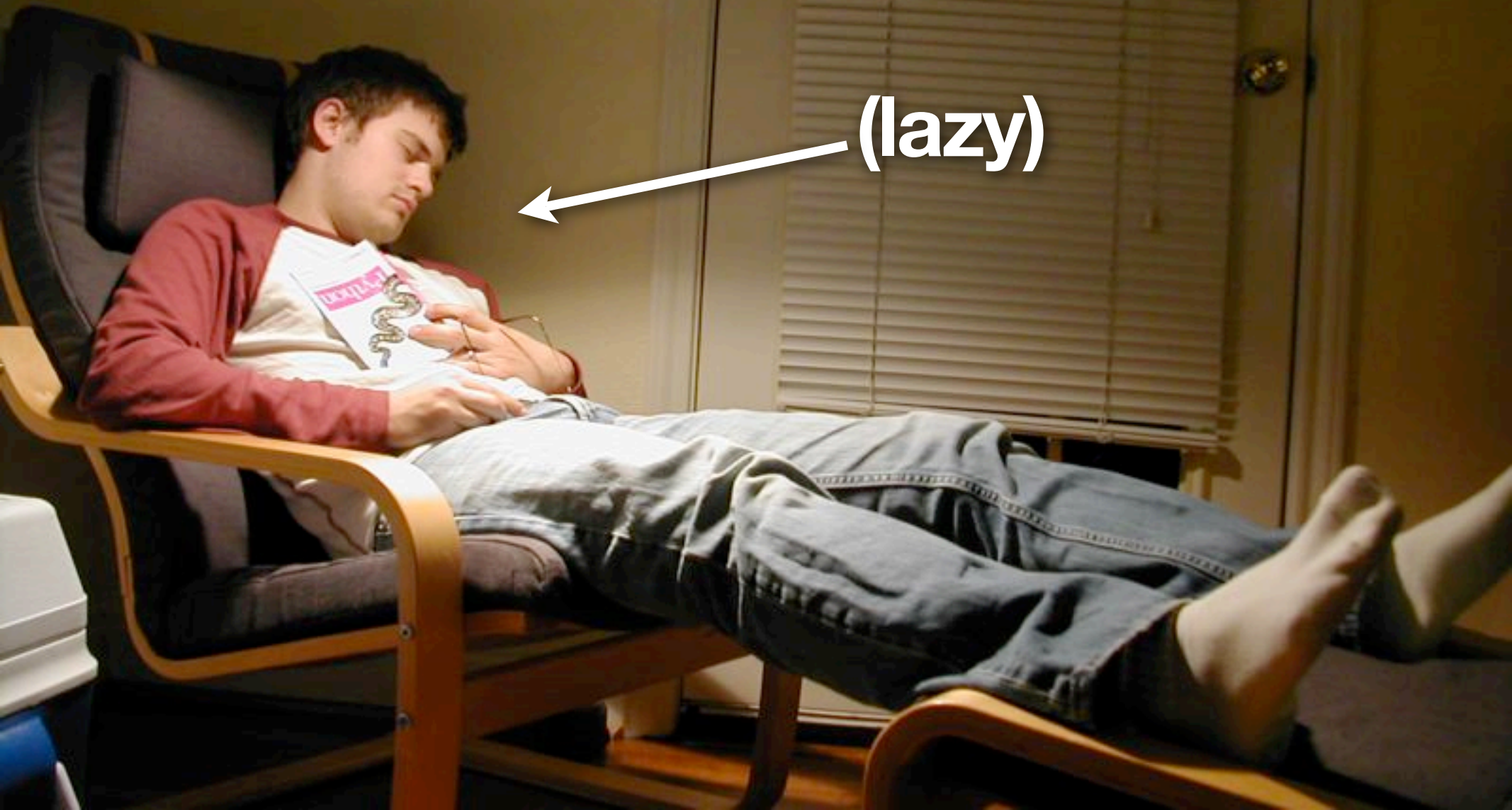
then,
I discovered
a language
called

python!

I am lazy.

Me, circa 2006

(lazy)



I am lazy.

- I am too lazy to wait for the compiler
- I am too lazy to switch between programming languages all the time
- I am too lazy to look up documentation
- I am too lazy to try to decipher line noise

Stuff I need to do:



- Sketch new ideas quickly
- Hack together analysis tools in the field
- Turn prototypes into polished research code
- Build web applications for course & departmental use

2.

**Seriously, tell us about
python already.**

python is...

- an interpreted programming language
- object-oriented
- dynamically typed
- blah
- blah
- blah

python is...

- handy
- smart
- fun
- helpful
- pretty

python is...

- handy
- smart
- fun
- helpful
- pretty

[houston craigslist](#) > [hackers seeking languages](#) > anyone out there?

anyone out there?

Reply to: pers-5038@craigslist.org

Date: 2006-11-30, 5:30PM CST

I'm looking for a programming language. Must be **handy, smart, pretty, fun, helpful**, and like late nights with coffee and conversation. By "conversation" I'm pretty much thinking "read-eval-print-loop." Tidy indentation a must.

Location: Rice University

It's NOT ok to contact this poster with programming languages that look like line noise or that require five lines to print "hello world"

5038

handy

- built-in libraries, everything I might need
 - data structures, concurrency
 - object serialization
 - cross-platform gui
 - sockets/smtp/mime/http/imap/xml/rpc/etc.
 - posix stuff, compression
 - regexps, unicode
 - cgi/httpd
- other modules: databases, scientific computing, image processing, crypto, ...

smart

- python is not just a “scripting language”
- pick your favorite programming paradigm
 - very fancy OOP
 - functional programming: not just supported, but natural

smart

- python is not just a “scripting language”
- pick your favorite programming paradigm
 - very fancy OOP
 - functional programming not just supported, but natural

lambda

fun

- every Python is a DrPython
 - the “read-eval-print loop” (REPL)
 - the only calculator you’ll ever need
 - experiment with live objects
 - (no waiting for the compiler)

helpful

- Every object, class, function, module is self-documenting
 - “Carry your documentation with you”
 - (It’s actually a property on the object)
- Essential, don’t-leave-home-without-them functions to use in the REPL:

help(*foo*)

dir(*foo*)

pretty

```
public class Hello {  
    public static void main(String[] args) {  
        System.out.println("Hello, world");  
    }  
}
```

pretty

```
public class Hello {  
    public static void main(String[] args) {  
        System.out.println("Hello, world");  
    }  
}
```

pretty

```
print ("Hello, world")
```

pretty (2)

- Ever seen this? *It should be illegal.*

```
if (x )
{ x = 2*x
; } else{
while(x>0)
    x -= 2;
    x = Math.sqrt(x);
}
```


pretty (2)

- Ever seen this? *It should be illegal.*

```
if (x) {  
    x = 2*x;  
} else {  
    while(x>0) {  
        x -= 2;  
    }  
    x = Math.sqrt(4*x);  
}
```

pretty (2)

- In Python, it *is* illegal.

```
if x:
    x = 2*x
else:
    while x>0:
        x -= 2

    x = math.sqrt(x)
```

- Indentation defines code blocks.
- Why? Code should be obvious and easy to read.

The perfect date, indeed

- **handy:** built-in libraries ftw
- **smart:** a real programming language
- **fun:** play around, develop quickly
- **helpful:** documentation is always there
- **pretty:** ugly code is hard to write

3.

The (clean) Python
phrasebook.

Simple types

- 1, -2.5, 0xDA51D, 3+5j, 9999999999L
- 'spam', "King Arthur's spam", '''multi-line spam'''
- True, False
- None

Compound types

- List (mutable sequence)
 - `[1, [2, 3], None, "eggs"]`
- Dictionary (mutable hash table)
 - `{"eggs": "The finest eggs in all Lilliput", "spam": "Spiced ham from Camelot"}`
- Tuple (immutable sequence)
 - `("eggs", 100, True)`

Naming, accessing

- `x = 1 # creates the name x and sets its value # to 1. Oh, hey, this is a comment.`
- `y = ["eggs", 2, 3]`
- `z = {"a": 1, "b": 2}`

- `x + 1 # result is 2`
- `y[1] # result is also 2`
- `z["b"] # ...still 2`

String formatting

- "Hello %s (%d years old)." % (whom, age)
- "Hello %(name)s (%(age)d years old)." % aDict

Functions, conditionals

```
def fib(n):  
    """Recursively computes the nth Fibonacci  
    number."""  
    if n == 0:  
        return 0  
    elif n == 1:  
        return 1  
    else:  
        return fib(n-1) + fib(n-2)
```

A simple program: “wc”

```
import java.lang.*;
import java.io.*;

public class wc {
    public static void main(String[] argv) throws IOException {
        open file BufferedReader input = new BufferedReader(
            new InputStreamReader(System.in));
        initialize boolean done = false;
        long count = 0;
        while (!done) {
            read String line = input.readLine();
            bail if (line == null) {
                done = true;
            } else {
                boolean inword = false;
                each char for (int i = 0; i < line.length(); i++) {
                    char ch = line.charAt(i);
                    new word, increment counter if (!inword) {
                        if (ch != ' ') {
                            inword = true;
                            count += 1;
                        }
                    } else {
                        end word if (ch == ' ') {
                            inword = false;
                        }
                    }
                }
            }
        }
        print System.out.println(count);
    }
}
```

“wc” in python (1)

```
import sys

done = False
count = 0
while not done:
    read line = sys.stdin.readline()
    bail if line == '': # EOF
        done = True
    else:
        inword = False
        every char for char in line:
            char new word if not inword:
                if not char.isspace():
                    inword = True
                    count += 1
            else:
                end word if char.isspace():
                    inword = False
        print print count
```

“wc” in python (2)

```
import sys
done = False
count = 0
while not done:
    line = sys.stdin.readline()
    if line == '': # EOF
        done = True
    else:
        count += len(line.split())
print count
```

length of any
sequence

split string
(default: into words)

“wc” in python (3)

```
import sys
count = 0
for line in sys.stdin: iterate lines in a file
    count += len(line.split())
print count
```

“wc” in python (4)

```
import sys
print len(sys.stdin.read().split())
```

read whole file into
a string

the Java version

```
import java.lang.*;
import java.io.*;

public class wc {
    public static void main(String[] argv) throws IOException {
        BufferedReader input = new BufferedReader(
            new InputStreamReader(System.in));

        boolean done = false;
        long count = 0;
        while (!done) {
            String line = input.readLine();
            if (line == null) {
                done = true;
            } else {
                boolean inword = false;
                for (int i = 0; i < line.length(); i++) {
                    char ch = line.charAt(i);
                    if (!inword) {
                        if (ch != ' ') {
                            inword = true;
                            count += 1;
                        }
                    } else {
                        if (ch == ' ') {
                            inword = false;
                        }
                    }
                }
            }
        }
        System.out.println(count);
    }
}
```

4.

It's not just me.

Beyond the interpreter

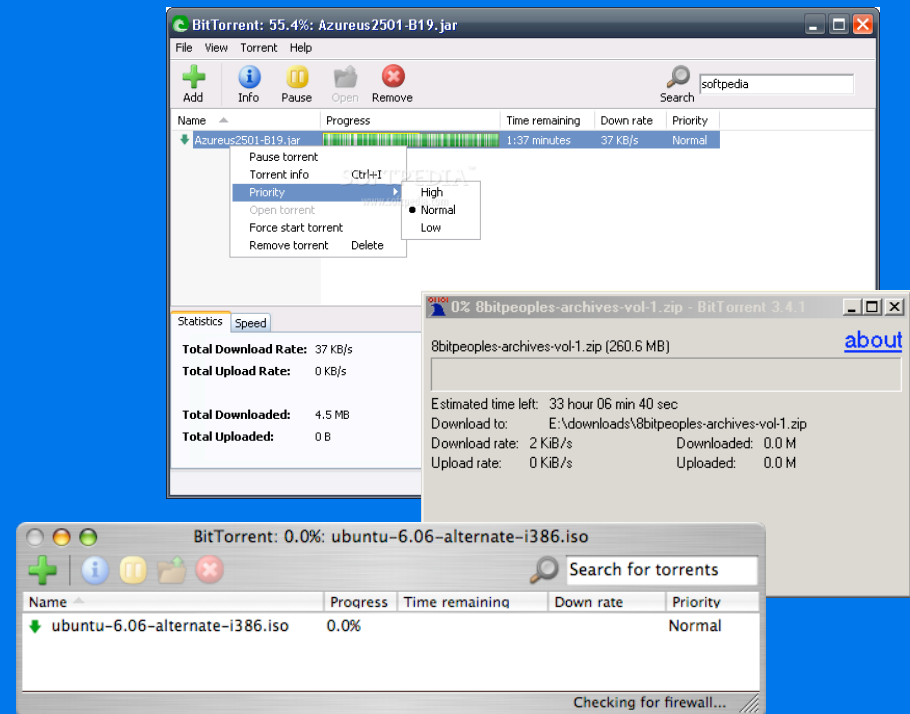
- Need speed?
 - Implement performance-critical code in C, wrap it in a Python interface for reuse
- Missing call/cc?
 - Stackless Python
- Stuck in another runtime?
 - Jython: Python syntax + JVM
 - IronPython: Python syntax + .NET CLR

Scientific computing

- Numeric python - <http://numpy.scipy.org/>
- pylab/matplotlib - <http://matplotlib.sf.net/>
- gnuplot.py, pydot
- *Next time you reach for matlab or gnuplot, try python instead*

End-user applications

- GUI toolkits:
 - Tk
 - wxWidgets
 - MFC (Windows)
 - Cocoa (OS X)
- BitTorrent



Industry

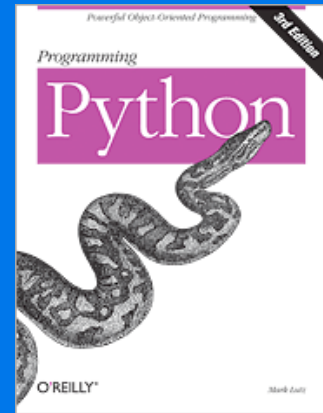
- Indexing the known universe (Google)
- Exploring the known universe (NASA)
- Recreating the known universe (ILM)
- Destroying the known universe (Eve Online)

Education

- Good for teaching
 - All the simplicity of Scheme
 - Familiar infix math
 - Widely used outside the academy
- A trend in CS education
 - MIT 6.001 headed this way
 - (Rice COMP 210 as well?)

Further reading

- *Dive Into Python*
<http://diveintopython.org/>
- *Programming Python* (O'Reilly)
- More links, plus these slides:
<http://www.cs.rice.edu/~dsandler/python/>



(updated 2006)

1.

(my python story)

2.

(about the language)

3.

(syntax)

4.

(resources)

16 tonnes

Brought to you by
the Ministry of Silly Talks