Python Sucks!

David Jones
drj@ravenbrook.com

drj11.wordpress.com

Ravenbrook

(Bourne) shell

JavaScript

(Standard) ML

C

Python

Lua

BCPL

Java

(BBC/Sinclair) BASIC

Fortran

Icon

(Common) Lisp

Scheme

awk

Ravenbrook

Python 3

Occam

CLU

Forth

Groovy

Haskell

Perl

PHP

Clojure

Ruby

Smalltalk

Erlang

Ada

D

C++

Scala

Objective C

Prolog

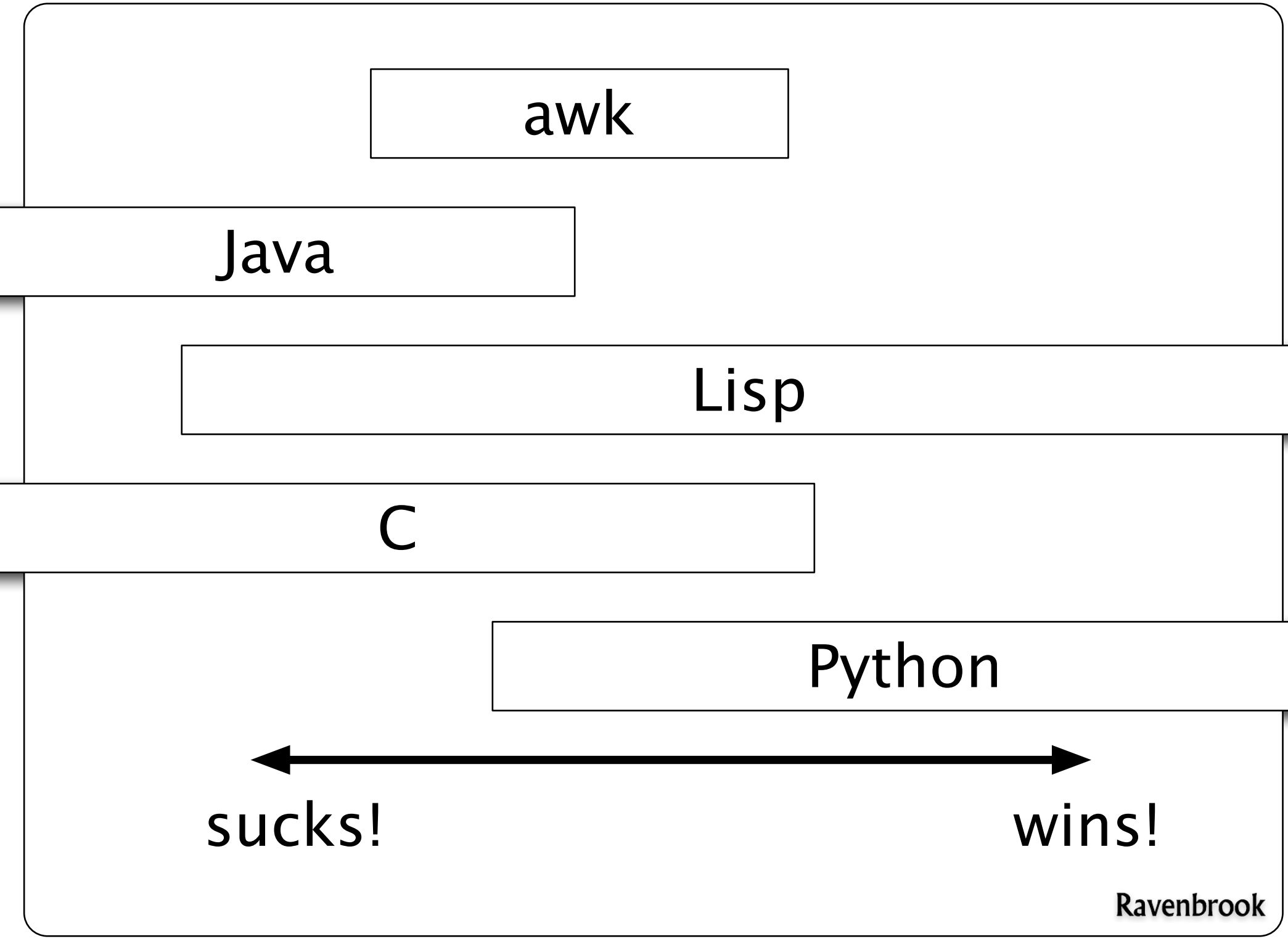
Ravenbrook

Python is
the worst
programming language
ever

Python is
the worst
programming language
ever
except
for all the others.

(After W Churchill)

```
while True:
    try:
        type, data = self.chunk()
    except ValueError, e:
        raise ChunkError(e.args[0])
```


awk
Java
Lisp
C
Python
sucks!
wins!
Ravenbrook

Lambda

λ

Lambda

Python 2.6.2 documentation:

> Note that functions created with lambda forms cannot contain statements.

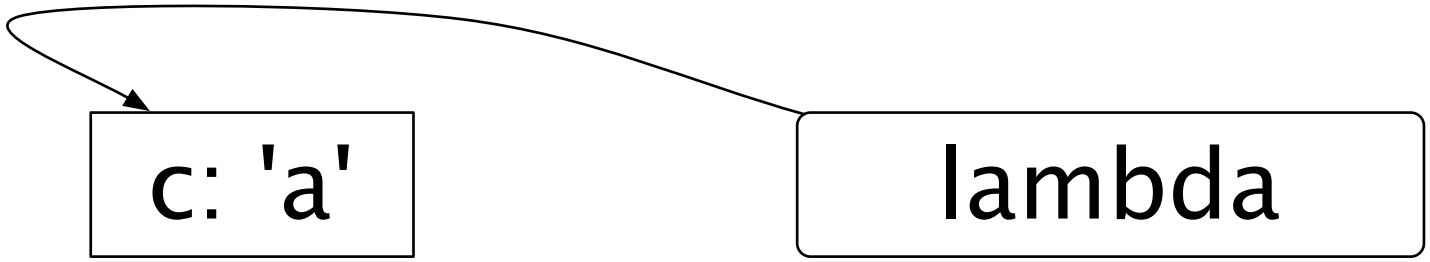
Bindings: Fail

```
contains = {}
for c in 'aeiou':
  contains[c] = lambda s: c in s
```

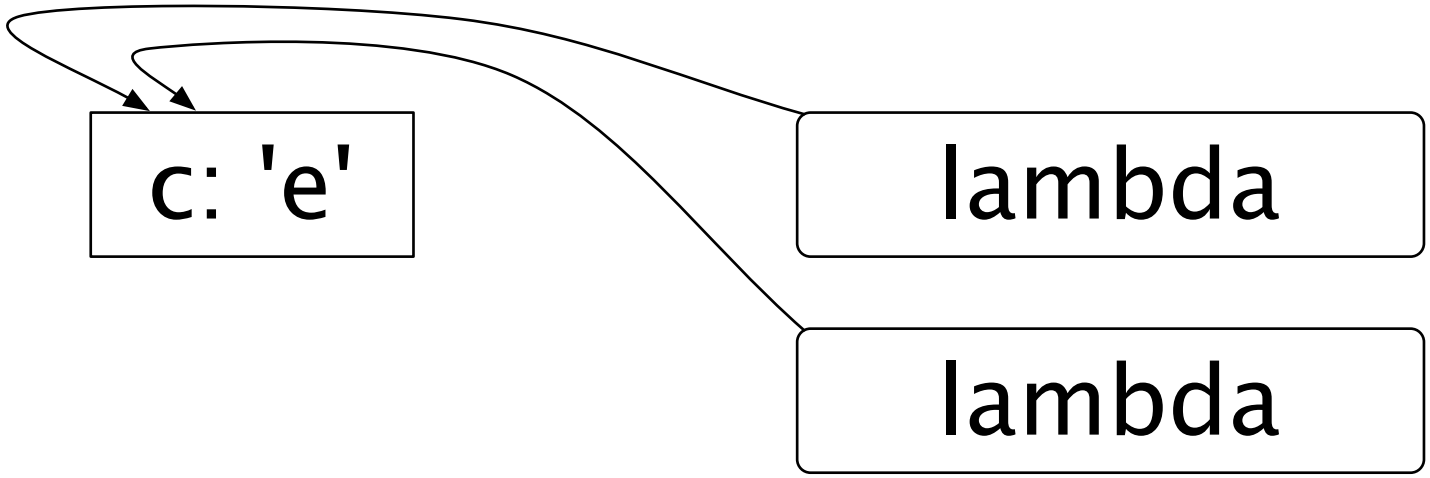
```
contains['u']('food') ⇒ False
contains['u']('zuul') ⇒ True

contains['a']('bar')  ⇒ False
contains['a']('zuul') ⇒ True
```

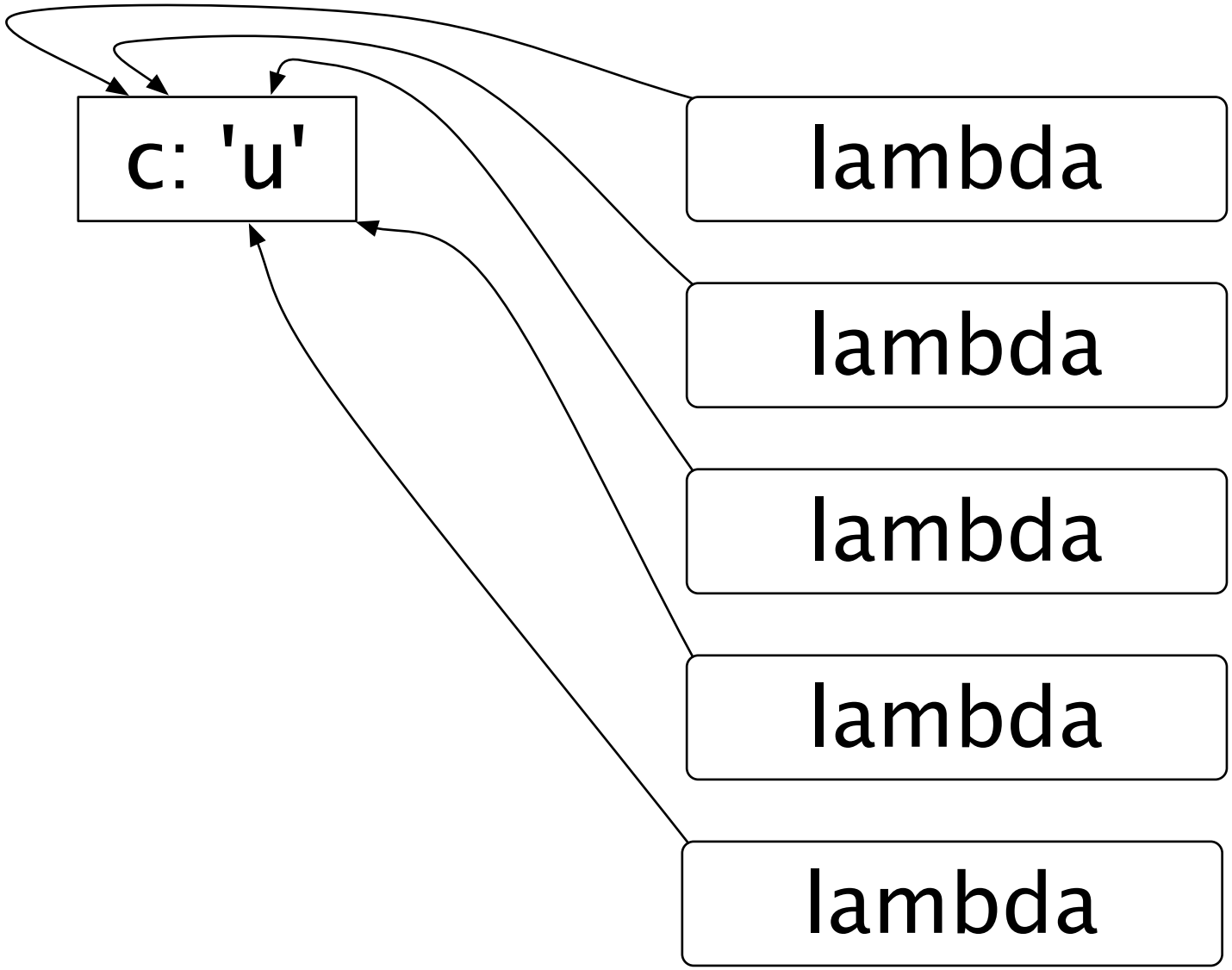
Bindings



Bindings

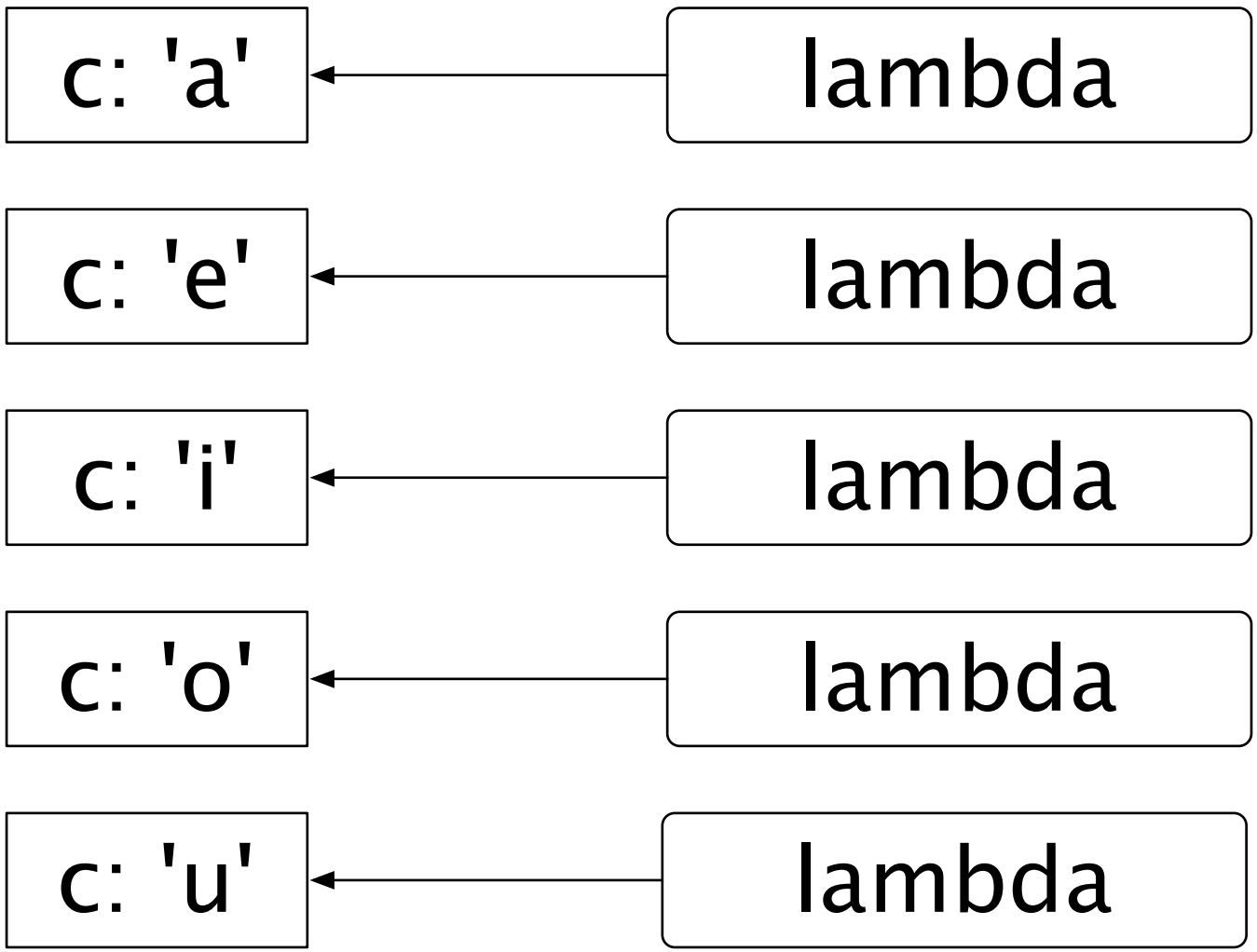


Bindings



Ravenbrook

Bindings



Ravenbrook

Bindings: Workaround

```
def f():
  ch=c
  return lambda s: ch in s

for c in 'aeiou':
  contains[c] = f()
```

Everything is an object

```
1
"strings"
lambda:'foo'
object()
import math
class bar(object):
```

Creating Objects

in Java/C++:

```
Class instance = new Class(args)
```

Creating Objects

in Python:

```
instance = Class(args)
```

```
def read(self):
  def iteridat():
      while True:
          type, data = self.chunk()
          ...
  ...
```

Ravenbrook

GIL, Threading

YOU fill in here

distutils

Easy Install

eggs

setuptools

buildout

PyPI

virtualenv

Ravenbrook

Chained Comparisons

```
assert 0 <= x < n
```

Chained Comparisons

```
a < b < c
```

in Python, same as:

```
(a < b) and (b < c)
```

Chained Comparisons

```
a < b < c
```

in C, same as:

```
(a < b) < c
```

... not helpful!

Chained Comparisons

$$b < a > c$$

Bad bad bad!

```
0 < x in l
```

Iverson's Convention

is where

`x < y`

evaluates to

`0` or `1`

(in Python
spelled `False` and `True`)

Ravenbrook

Arithmetic with Iverson

```
color_type = 4*alpha +
  2*(not greyscale) + 1*colormap
```

Infinity

∞

NaN

IEEE 754

...

and all that

To Infinity ...

```
>>> 9e999
inf
>>> float('inf')
inf
```

```
>>> struct.unpack('>f',
  '7f800000'.decode('hex'))[0]
inf
```

Ravenbrook

and Beyond!

```
>>> struct.unpack('>f',
  '7f800001'.decode('hex'))[0]
nan
```

Floating Point Printing

```
>>> 0.1
0.10000000000000001
```

Python, Divided

Python 2.x

```
3 / 2   ⇒ 1
3.0 / 2 ⇒ 1.5
```

Python 3.x

```
3 / 2   ⇒ 1.5
3.0 / 2 ⇒ 1.5
```

Division Operators

```
3.0 // 2 ⇒ 1.0

3.0 / 2  ⇒ 1.5
```

Lisp, Divided

```
(floor 3 2)      ⇒ 1
(ceiling 3 2)    ⇒ 2
(truncate 3 2)   ⇒ 1
(truncate -3 2)  ⇒ -1
(floor -3 2)     ⇒ -2
(round 3 2)      ⇒ 2
(round 5 2)      ⇒ 2
```

Python, Fraction

```
from fractions import Fraction
```

```
1/3-1/11
```

$\Rightarrow$

?

Python, Fraction

```
>>> 1/3-1/11
0.2424242424242424
>>> _*33
7.9999999999999991
```

```
>>> Fraction(1,3)-Fraction(1,11)
Fraction(8, 33)
>>> _*33
Fraction(8, 1)
```

Ravenbrook

Python Power!

```
x**2
x**0.5
```

Decorators

```
@someclassdecorator
class foo:
    @somemethoddecorator
    def bar(self):
        pass
```

Argument List Splicing

```
zip(*l)

dict(a=1, **d)
```

Punctuation in Names

```
int?(x)
instance?(x, someclass)

l.sort!()
```