INFO-H-302
Introduction Python

Analyse et Conception par Objets

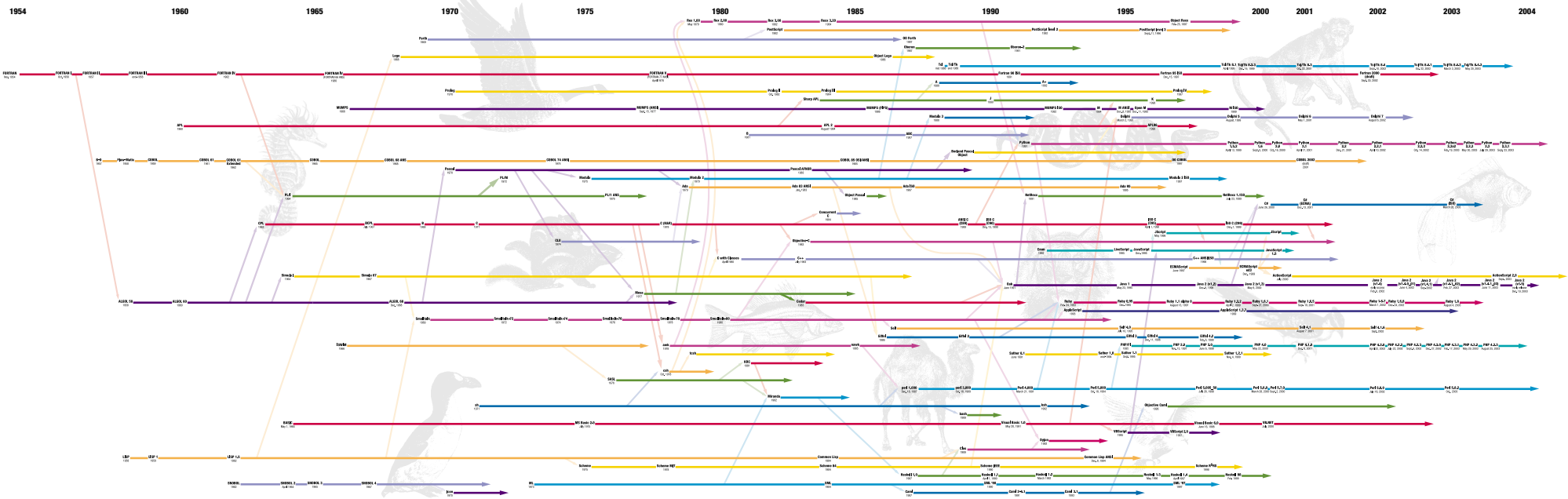
Pourquoi apprendre Python ?

Pourquoi apprendre un nouveau langage ?

- ... pourquoi pas ?
- "Controlling complexity is the essence of computer programming."
  (Brian Kernighan)
- Autre langage, autre approche de la complexité

History of Programming Languages

O'REILLY®



www.oreilly.com

For more than half of the fifty years computer programmers have been writing code, O'Reilly has provided developers with comprehensive, in-depth technical information. We've kept pace with rapidly changing technologies as new languages have emerged, developed, and matured. Whether you want to learn something new or need answers to tough technical questions, you'll find what you need in O'Reilly books and on the O'Reilly Network.

This timeline includes fifty of the more than 2500 documented programming languages. It is based on an original diagram created by Eric Levenez (www.levenez.com), augmented with suggestions from O'Reilly authors, friends, and conference attendees.

For information and discussion on this poster, go to *www.oreilly.com/go/languageposter*.

Mother Tongues

Tracing the roots of computer languages through the ages

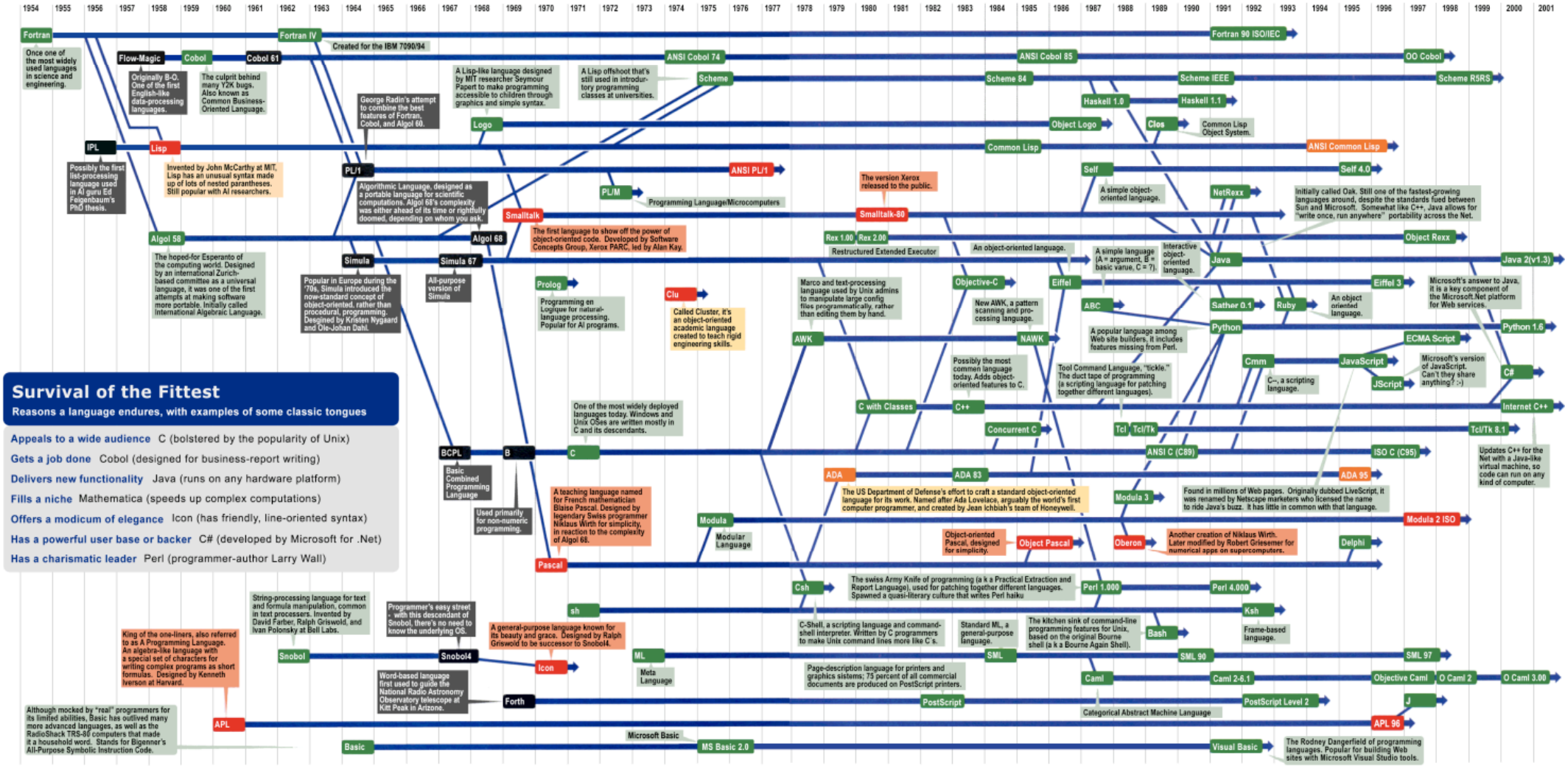
Just like half of the world's spoken tongues, most of the 2,300-plus computer programming languages are either endangered or extinct. As powerhouses C/C++, Visual Basic, Cobol, Java and other modern source codes dominate our systems, hundreds of older languages are running out of life.

An ad hoc collection of engineers-electronic lexicographers, if you will-aim to save, or at least document the lingo of classic software. They're combing the globe's 9 million developers in search of coders still fluent in these nearly forgotten lingua frangas. Among the most endangered are Ada, APL, B (the predecessor of C), Lsp, Oberon, Smalltalk, and Simula.

Code-raker Grady Booch, Rational Software's chief scientist, is working with the Computer History Musuem in Silicon Valley to record and, in some cases, maintain languages by writing new compilers so our ever-changing hardware can grok the code. Why bother? "They tell us about the state of software practice, the minds of their inventors, and the technical, social, and economic forces that shaped history at the time," Booch explains. "They'll provide the raw material for software archaeologists, historians, and developers to learn what worked, what was brilliant, and what was an utter failure." Here's a peek at the strongest branches of programming's family tree. For a nearly exhaustive rundown, check out the Language List at *HTTP://www.informatik.uni-freiburg.de/Java/misc/lang_list.html*. - Michael Mendeno

Key

- 1954 Year Introduced
- **Active:** thousands of users
- **Protected:** taught at universities; compilers available
- **Endangered:** usage dropping off
- **Extinct:** no known active users or up-to-date compilers
- **Lineage continues**

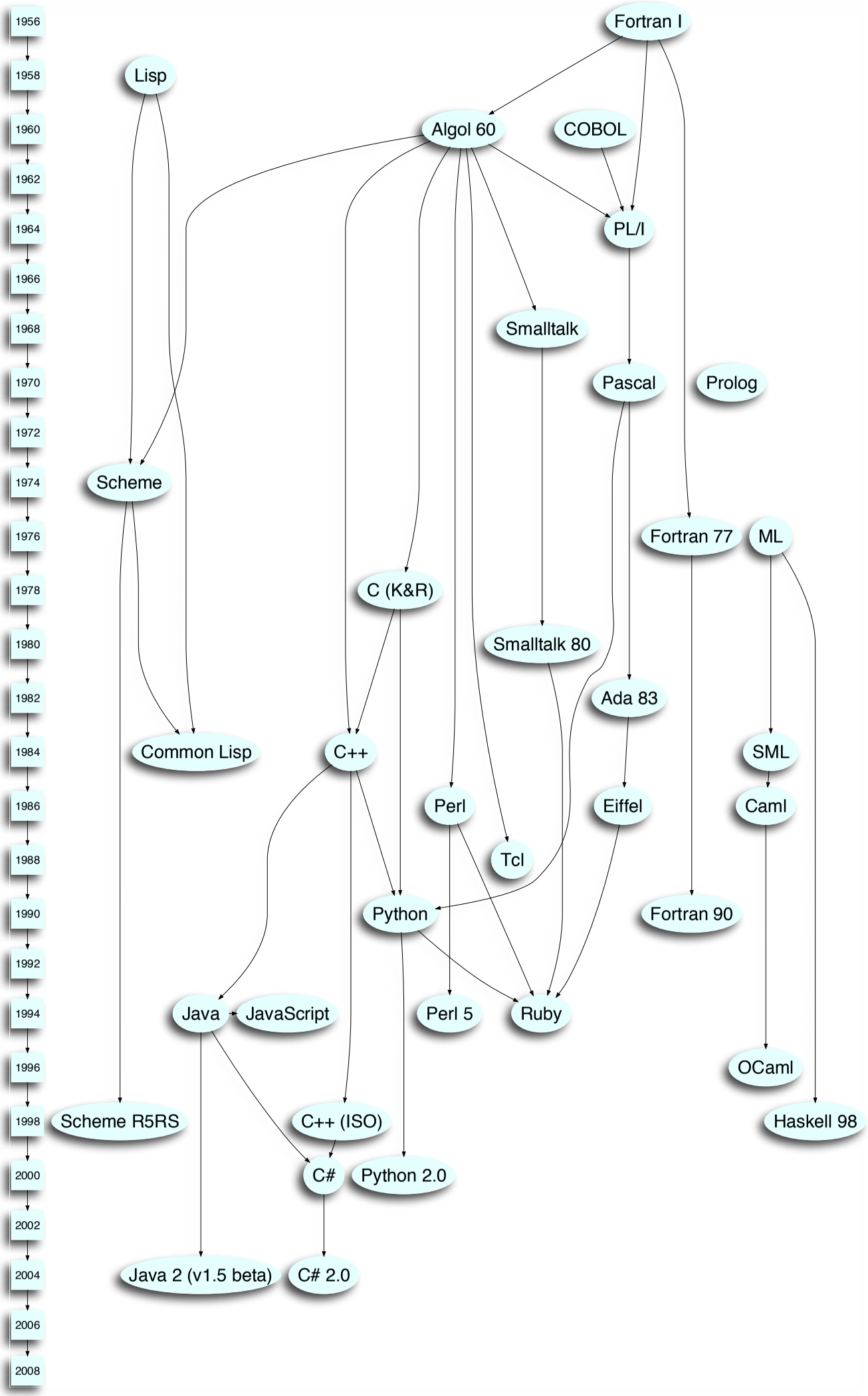


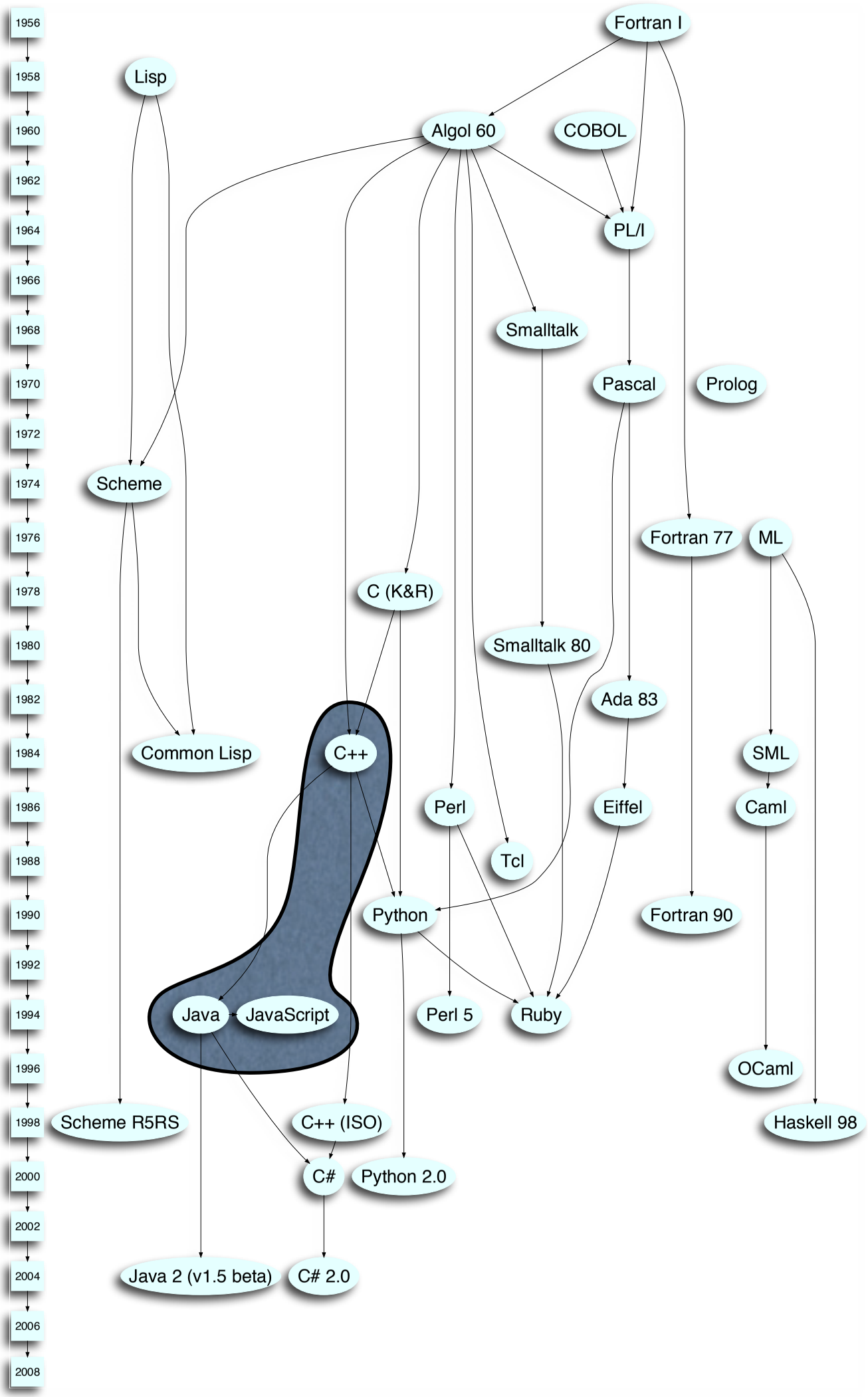
Survival of the Fittest

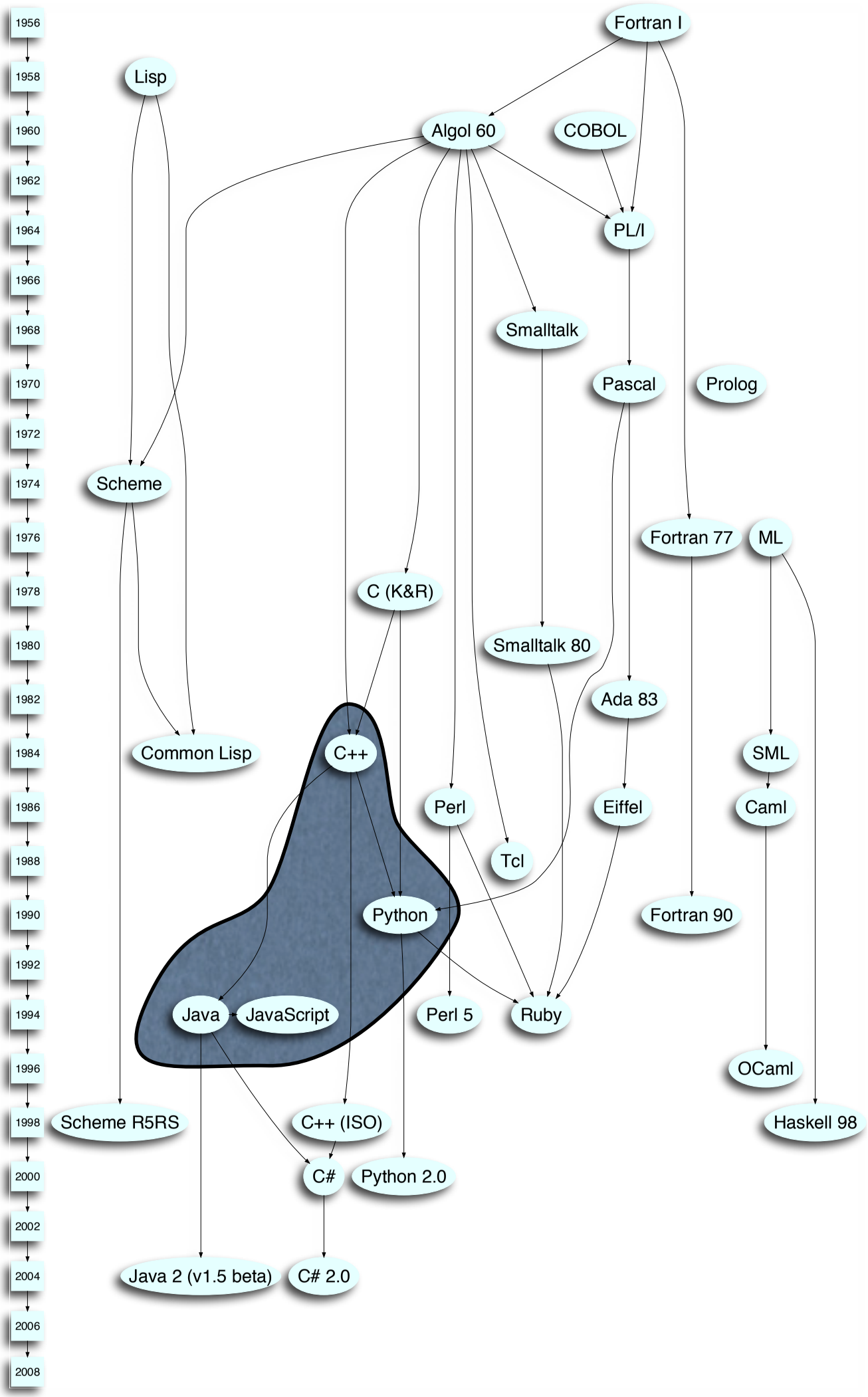
Reasons a language endures, with examples of some classic tongues

- **Appeals to a wide audience** C (bolstered by the popularity of Unix)
- **Gets a job done** Cobol (designed for business-report writing)
- **Delivers new functionality** Java (runs on any hardware platform)
- **Fills a niche** Mathematica (speeds up complex computations)
- **Offers a modicum of elegance** Icon (has friendly, line-oriented syntax)
- **Has a powerful user base or backer** C# (developed by Microsoft for .Net)
- **Has a charismatic leader** Perl (programmer-author Larry Wall)

Sources: Paul Boutin; Brent Hailpern, associate director of computer science at IBM Research; The Retrocomputing Museum; Todd Proebsting, senior researcher at Microsoft; Gio Wiederhold, computer scientist, Stanford University


1956
1958
1960
1962
1964
1966
1968
1970
1972
1974
1976
1978
1980
1982
1984
1986
1988
1990
1992
1994
1996
1998
2000
2002
2004
2006
2008
Fortran I
Lisp
Algol 60
COBOL
PL/I
Smalltalk
Pascal
Prolog
Scheme
Fortran 77
ML
C (K&R)
Smalltalk 80
Ada 83
Common Lisp
C++
SML
Perl
Eiffel
Caml
Tcl
Python
Fortran 90
Java
JavaScript
Perl 5
Ruby
OCaml
Scheme R5RS
C++ (ISO)
Haskell 98
C#
Python 2.0
Java 2 (v1.5 beta)
C# 2.0


1956
1958
1960
1962
1964
1966
1968
1970
1972
1974
1976
1978
1980
1982
1984
1986
1988
1990
1992
1994
1996
1998
2000
2002
2004
2006
2008
Fortran I
Lisp
Algol 60
COBOL
PL/I
Smalltalk
Pascal
Prolog
Scheme
Fortran 77
ML
C (K&R)
Smalltalk 80
Ada 83
Common Lisp
C++
SML
Perl
Eiffel
Caml
Tcl
Python
Fortran 90
Java
JavaScript
Perl 5
Ruby
OCaml
Scheme R5RS
C++ (ISO)
Haskell 98
C#
Python 2.0
Java 2 (v1.5 beta)
C# 2.0

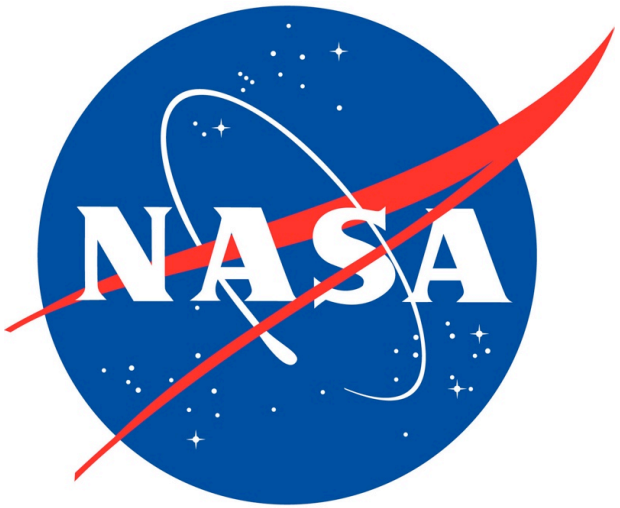

1956
1958
1960
1962
1964
1966
1968
1970
1972
1974
1976
1978
1980
1982
1984
1986
1988
1990
1992
1994
1996
1998
2000
2002
2004
2006
2008
Fortran I
Lisp
Algol 60
COBOL
PL/I
Smalltalk
Pascal
Prolog
Scheme
Fortran 77
ML
C (K&R)
Smalltalk 80
Ada 83
Common Lisp
C++
SML
Perl
Eiffel
Caml
Tcl
Python
Fortran 90
Java
JavaScript
Perl 5
Ruby
OCaml
Scheme R5RS
C++ (ISO)
Haskell 98
C#
Python 2.0
Java 2 (v1.5 beta)
C# 2.0

Pourquoi apprendre Python ?

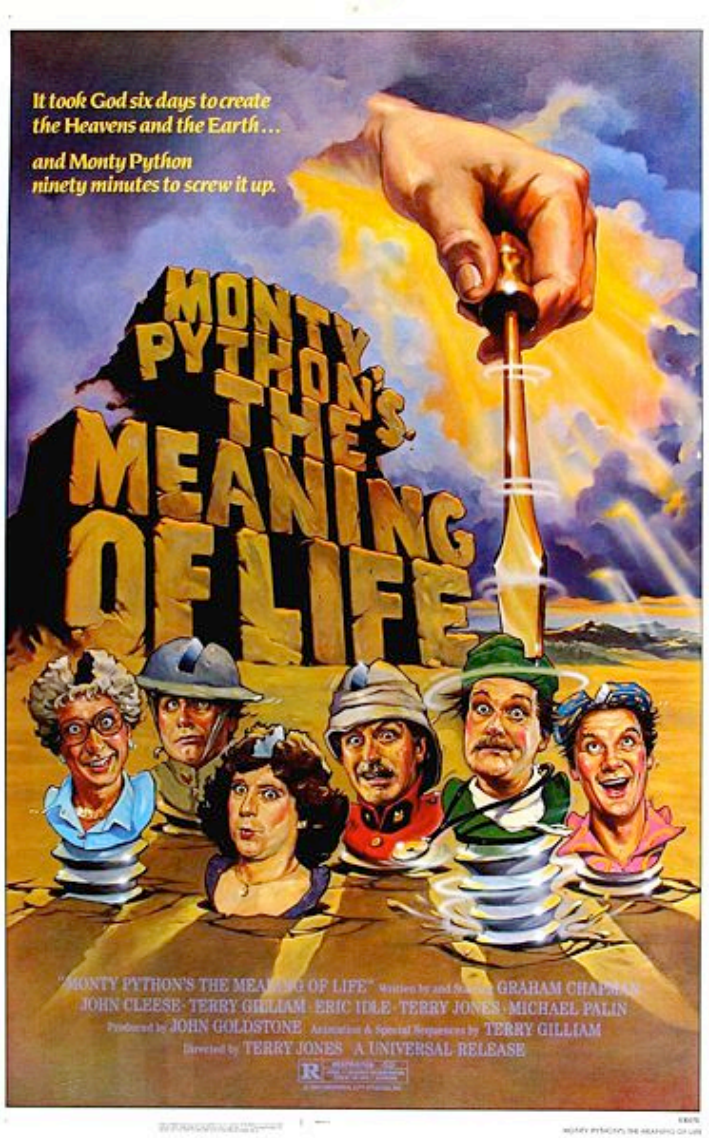
- Suffisamment proche de Java pour être facile
- Suffisamment différent pour être intéressant
- Très utilisé :
  - Développement web
  - Recherche en Mathématiques, Bioinformatique
  - Administration système


Google™


ZOPE
Community


NASA

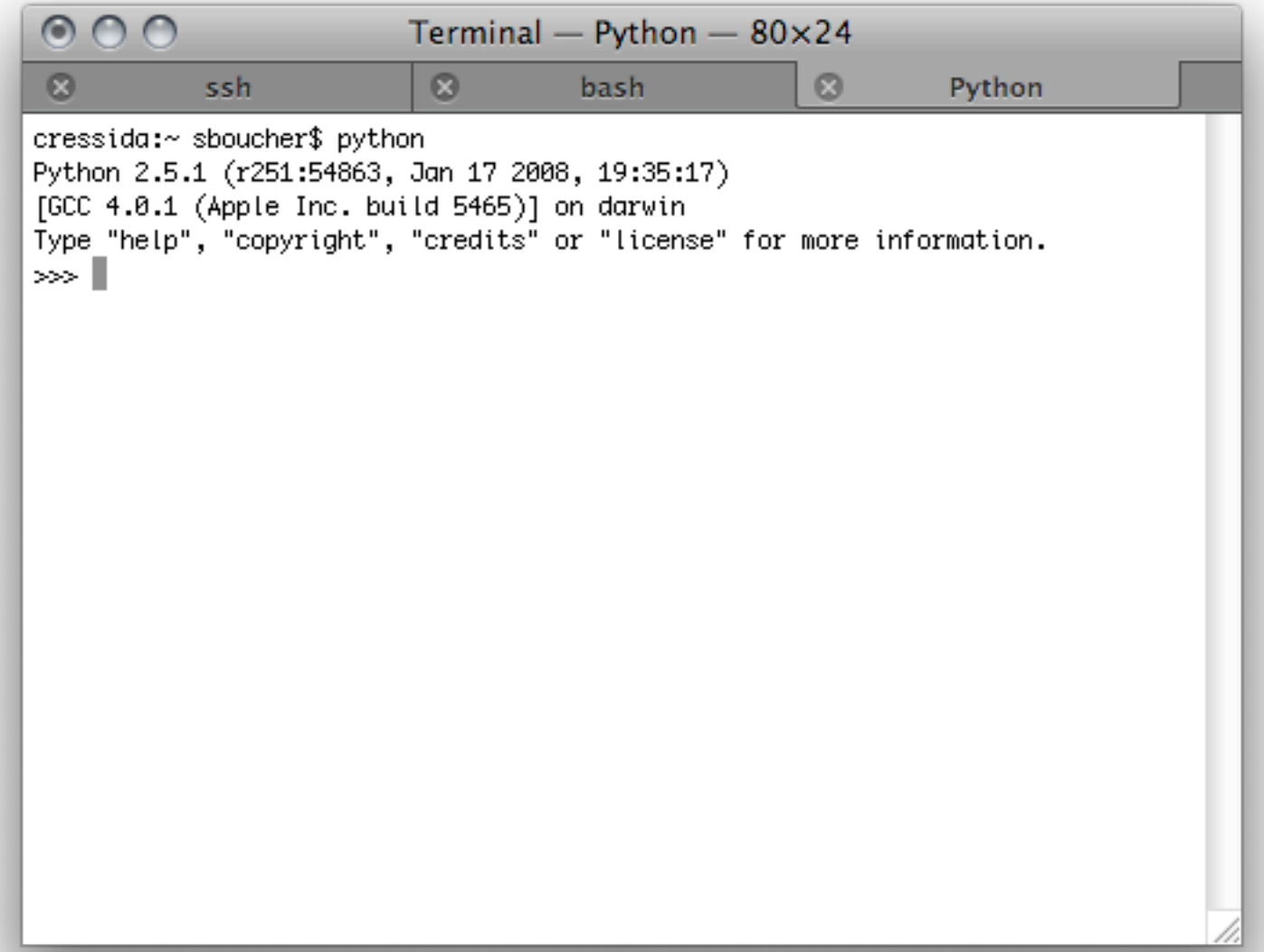

INDUSTRIAL LIGHT+MAGIC
A LUCASFILM LTD. COMPANY



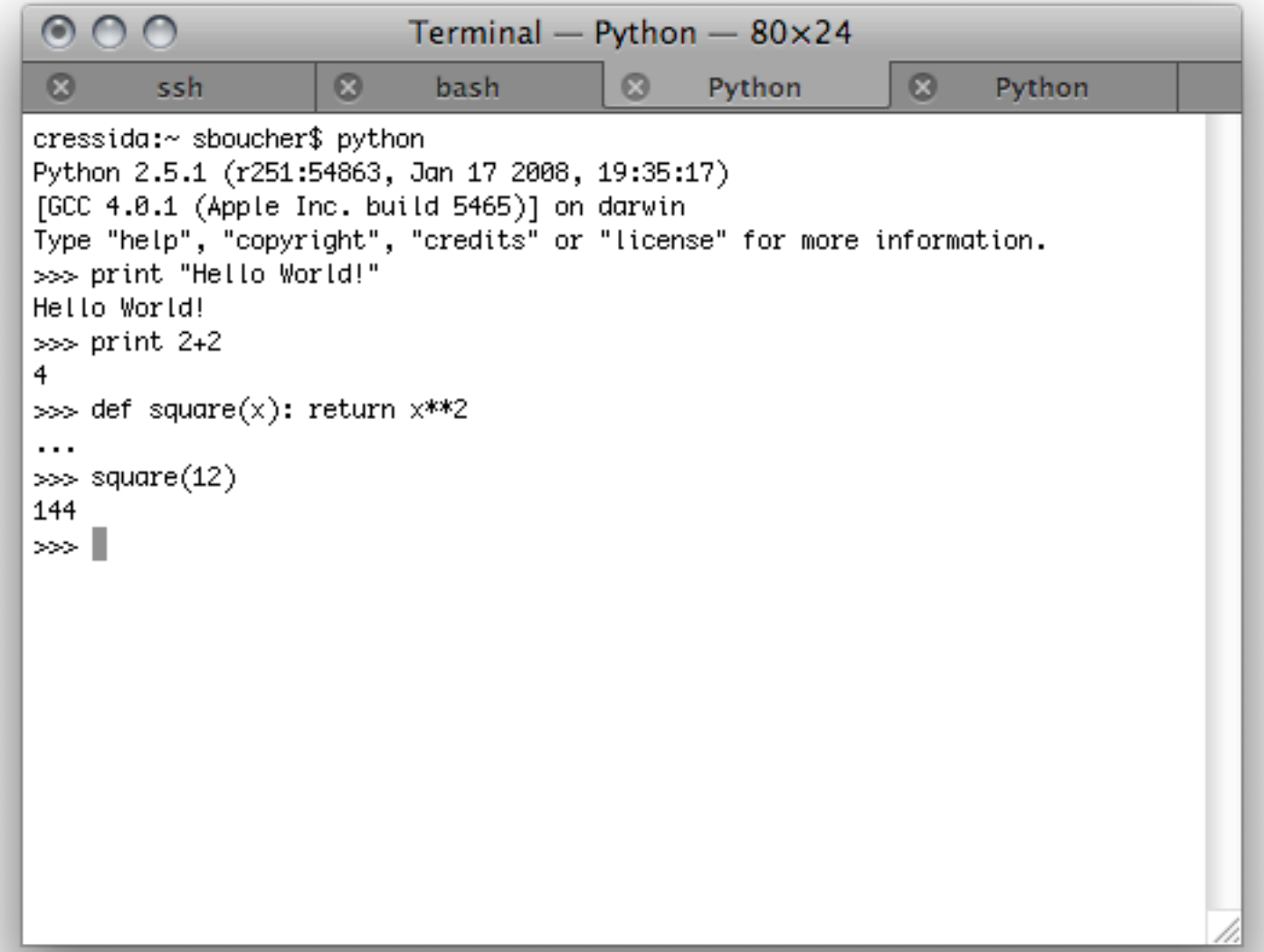
Python

- Interprété
- Orienté-Objet
- Strong typing
- Dynamic typing

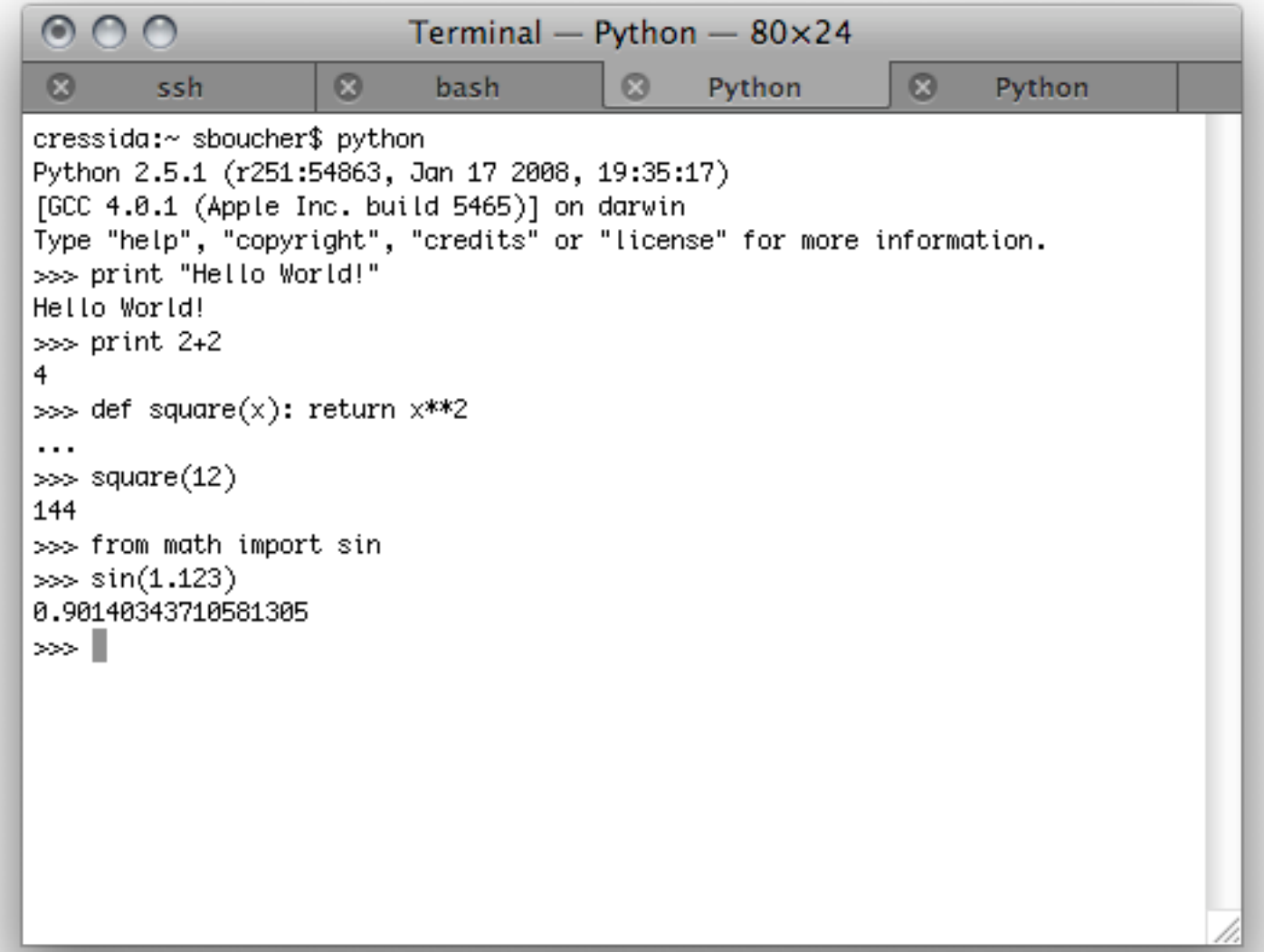
Interprété


Terminal — Python — 80×24
ssh
bash
Python
cressida:~ sboucher$ python
Python 2.5.1 (r251:54863, Jan 17 2008, 19:35:17)
[GCC 4.0.1 (Apple Inc. build 5465)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>>

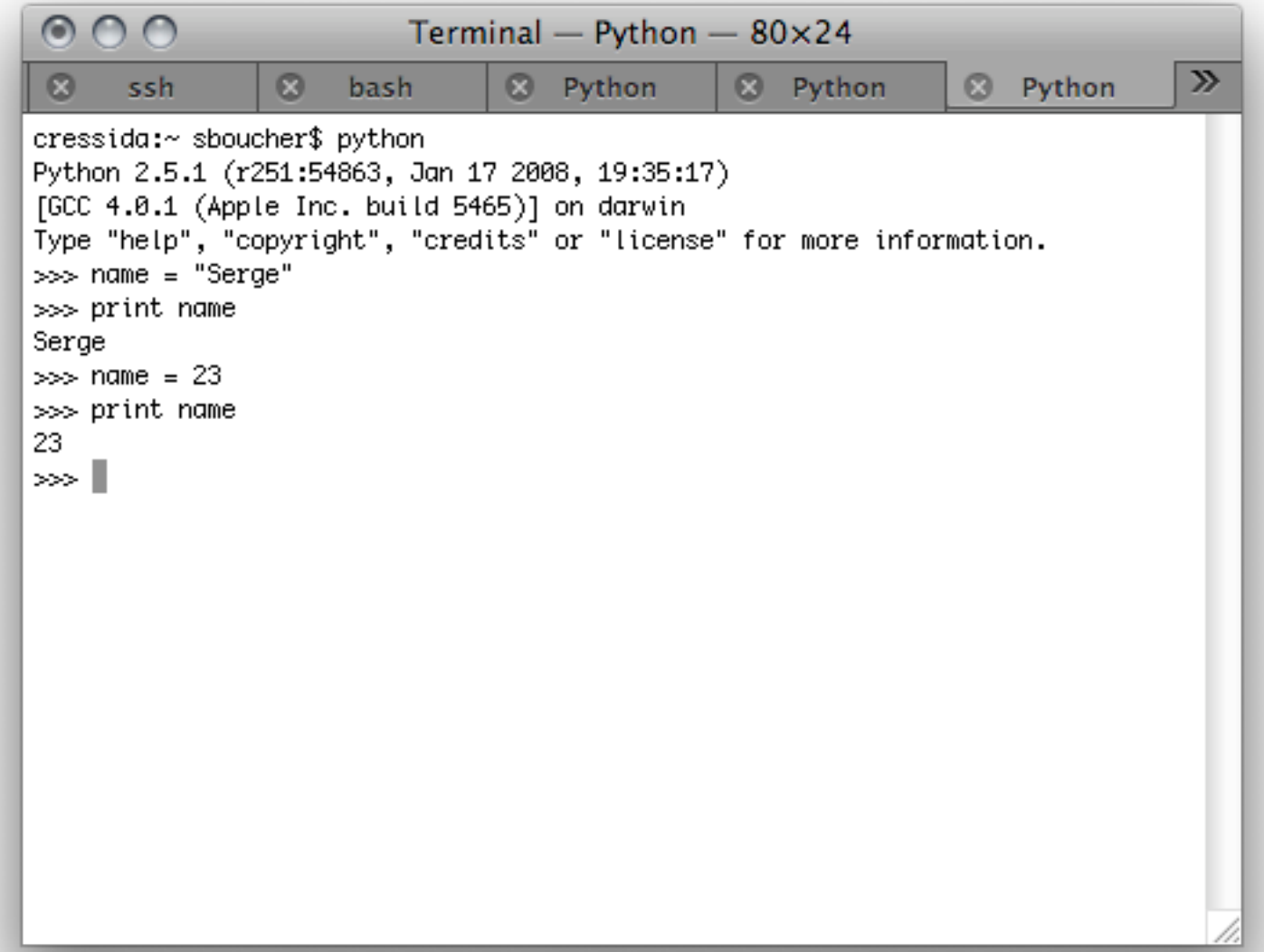
Interprété


Terminal — Python — 80×24
ssh
bash
Python
Python
cressida:~ sboucher$ python
Python 2.5.1 (r251:54863, Jan 17 2008, 19:35:17)
[GCC 4.0.1 (Apple Inc. build 5465)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> print "Hello World!"
Hello World!
>>> print 2+2
4
>>> def square(x): return x**2
...
>>> square(12)
144
>>>

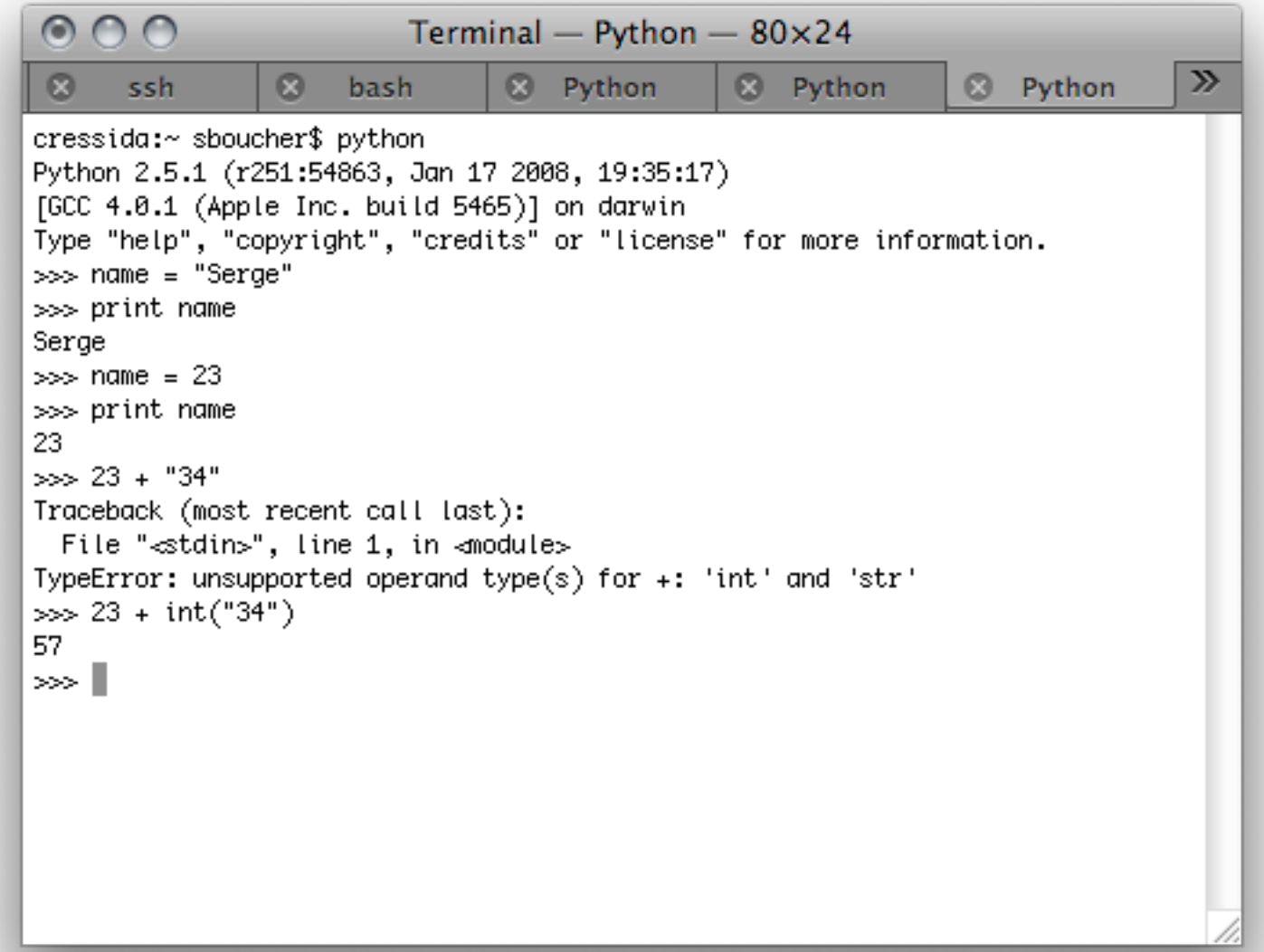
Interprété


Terminal — Python — 80×24
ssh
bash
Python
Python
cressida:~ sboucher$ python
Python 2.5.1 (r251:54863, Jan 17 2008, 19:35:17)
[GCC 4.0.1 (Apple Inc. build 5465)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> print "Hello World!"
Hello World!
>>> print 2+2
4
>>> def square(x): return x**2
...
>>> square(12)
144
>>> from math import sin
>>> sin(1.123)
0.90140343710581305
>>>

Typage dynamique



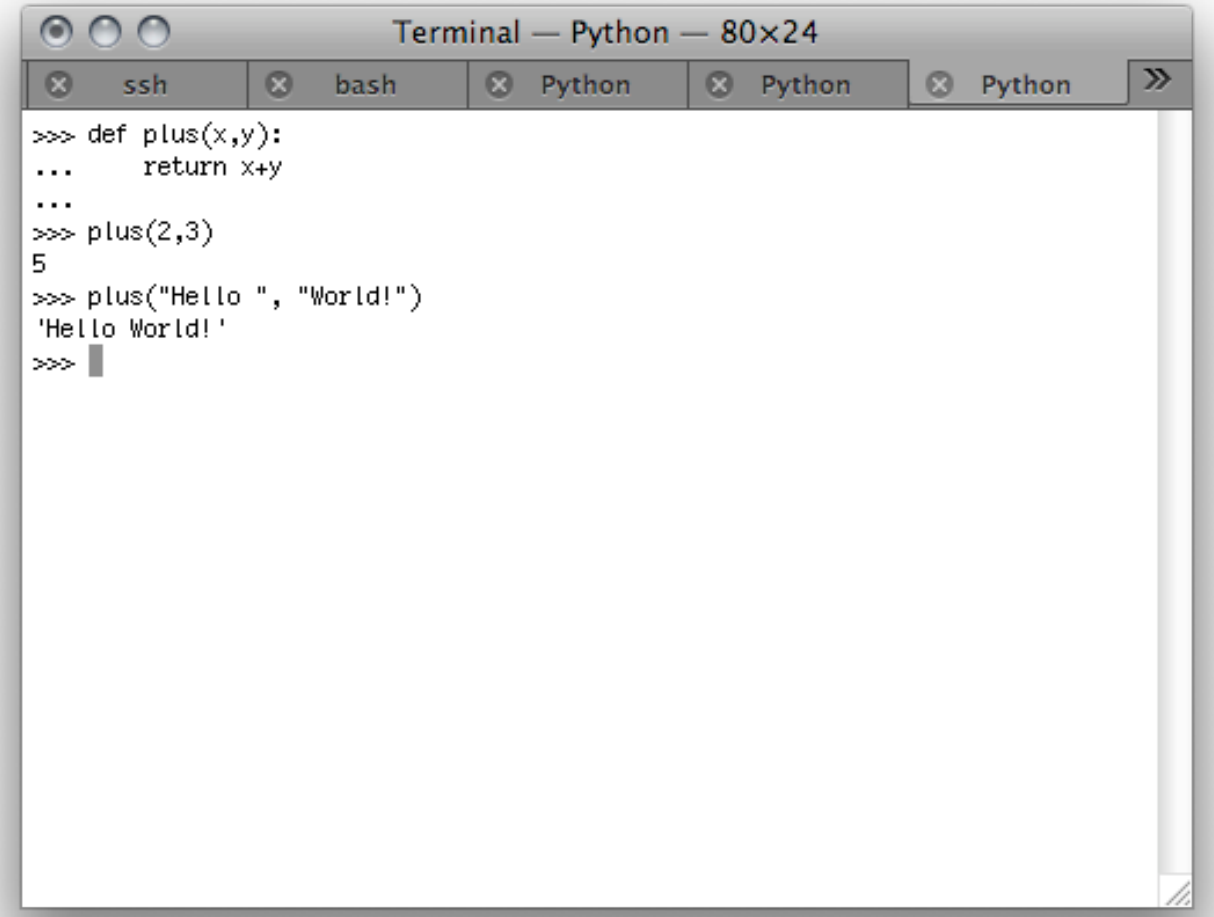
Typage fort


Terminal — Python — 80×24
ssh
bash
Python
Python
Python
cressida:~ sboucher$ python
Python 2.5.1 (r251:54863, Jan 17 2008, 19:35:17)
[GCC 4.0.1 (Apple Inc. build 5465)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> name = "Serge"
>>> print name
Serge
>>> name = 23
>>> print name
23
>>> 23 + "34"
Traceback (most recent call last):
File "<stdin>", line 1, in <module>
TypeError: unsupported operand type(s) for +: 'int' and 'str'
>>> 23 + int("34")
57
>>>

“Duck-Typing”

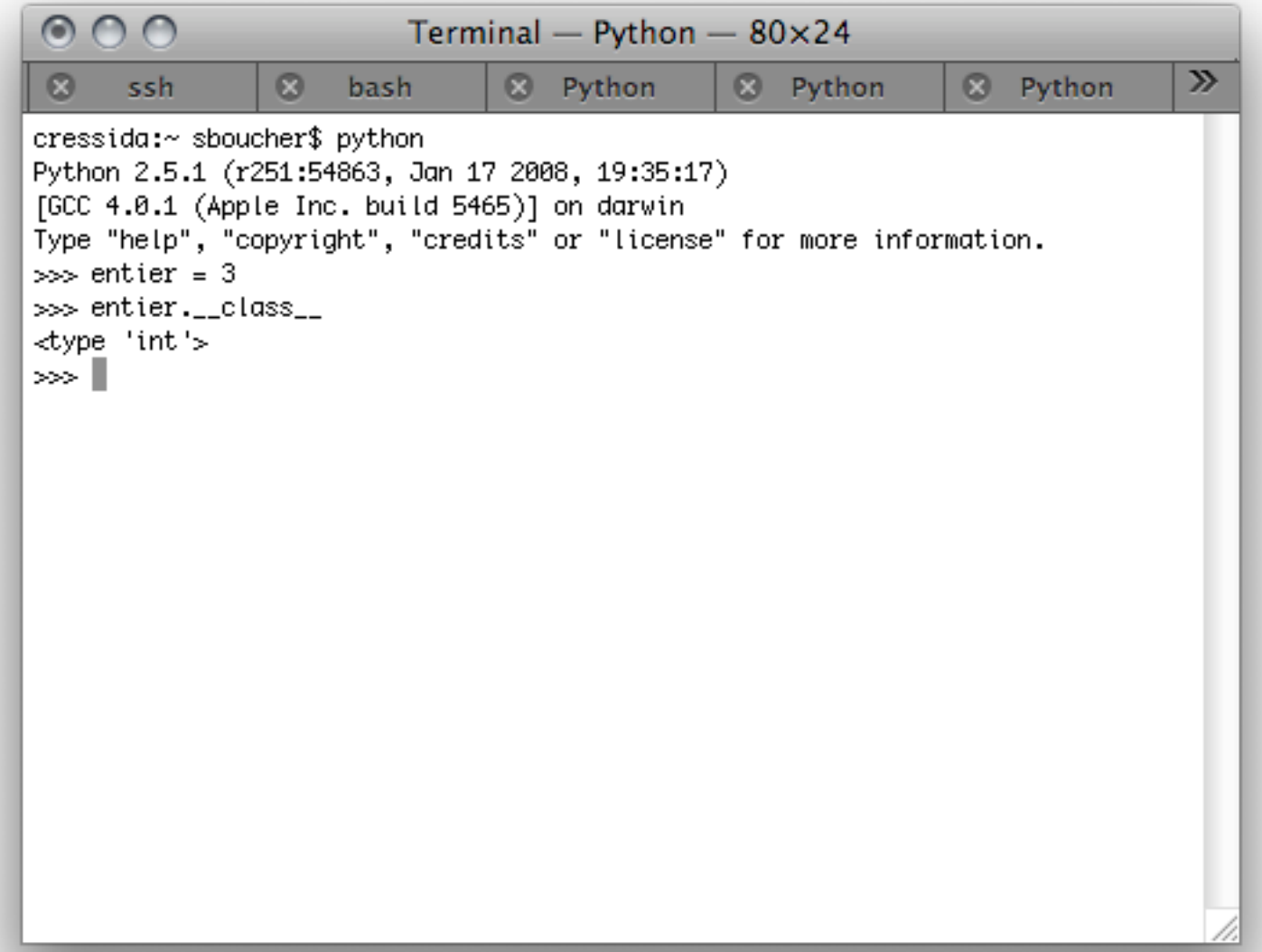
- “If it quacks like a duck and flies like a duck... it must be a duck”

“Duck-Typing”

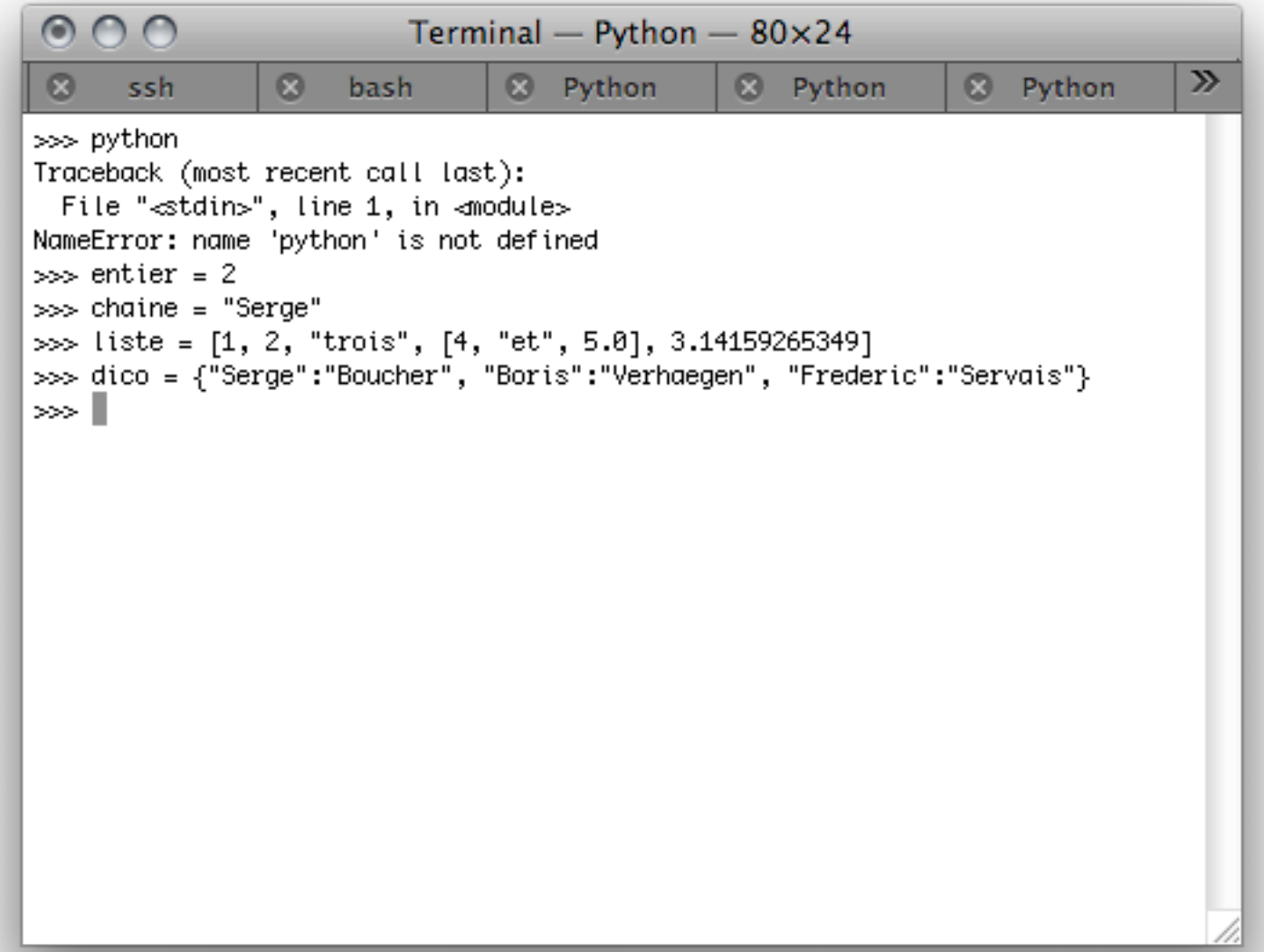


- “If it quacks like a duck and flies like a duck... it must be a duck”

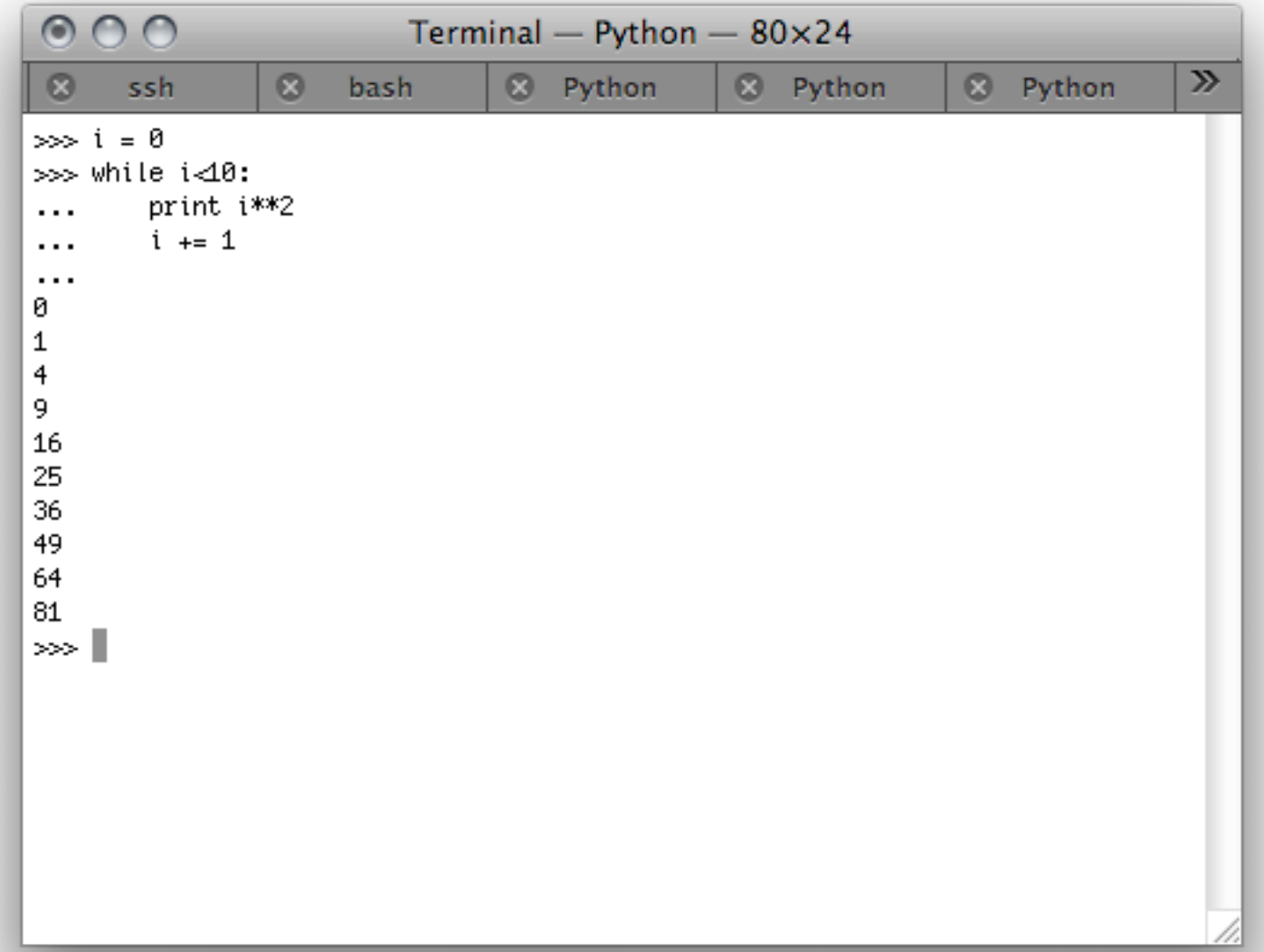
Orienté-Objet



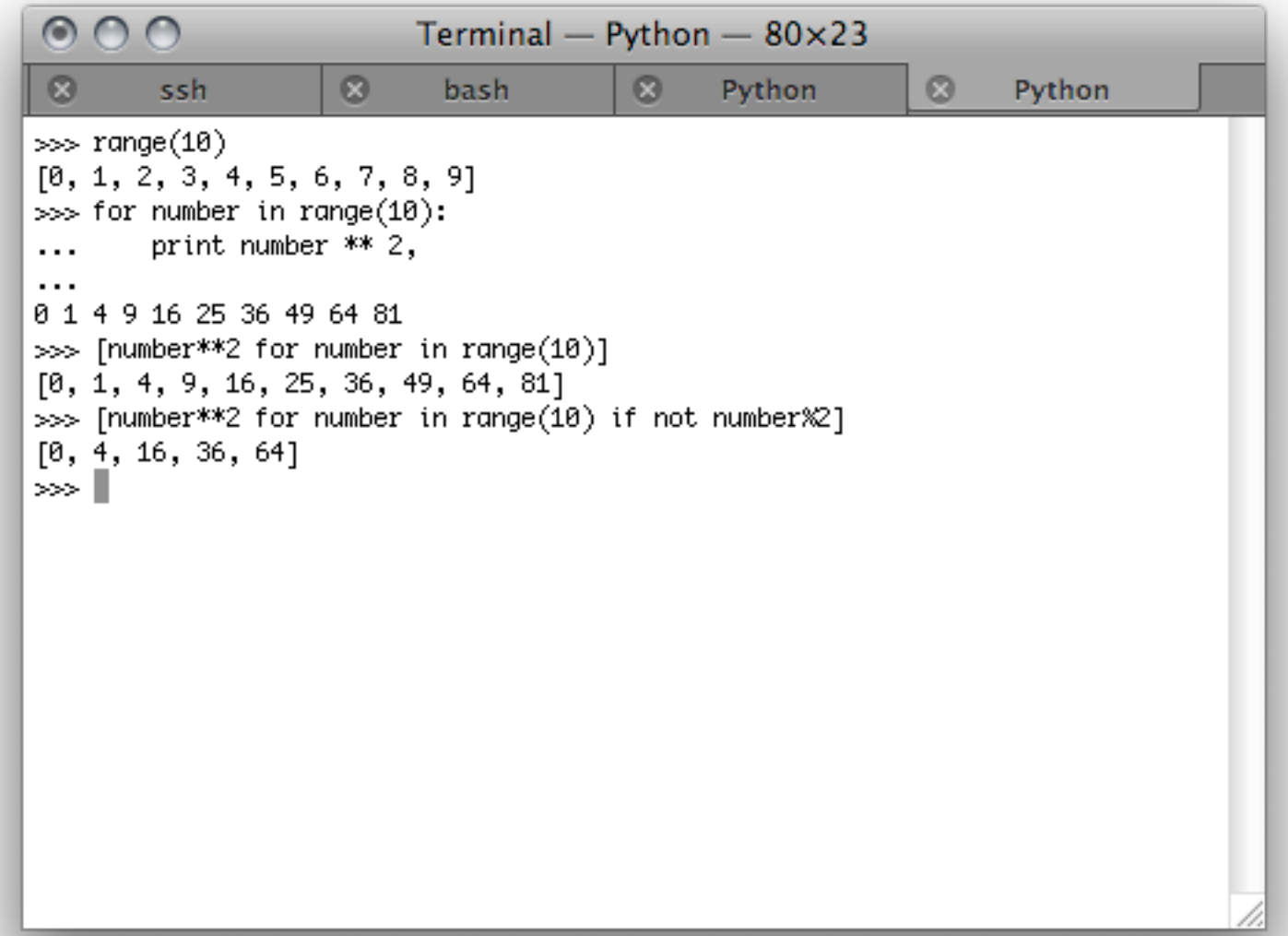
Types de données



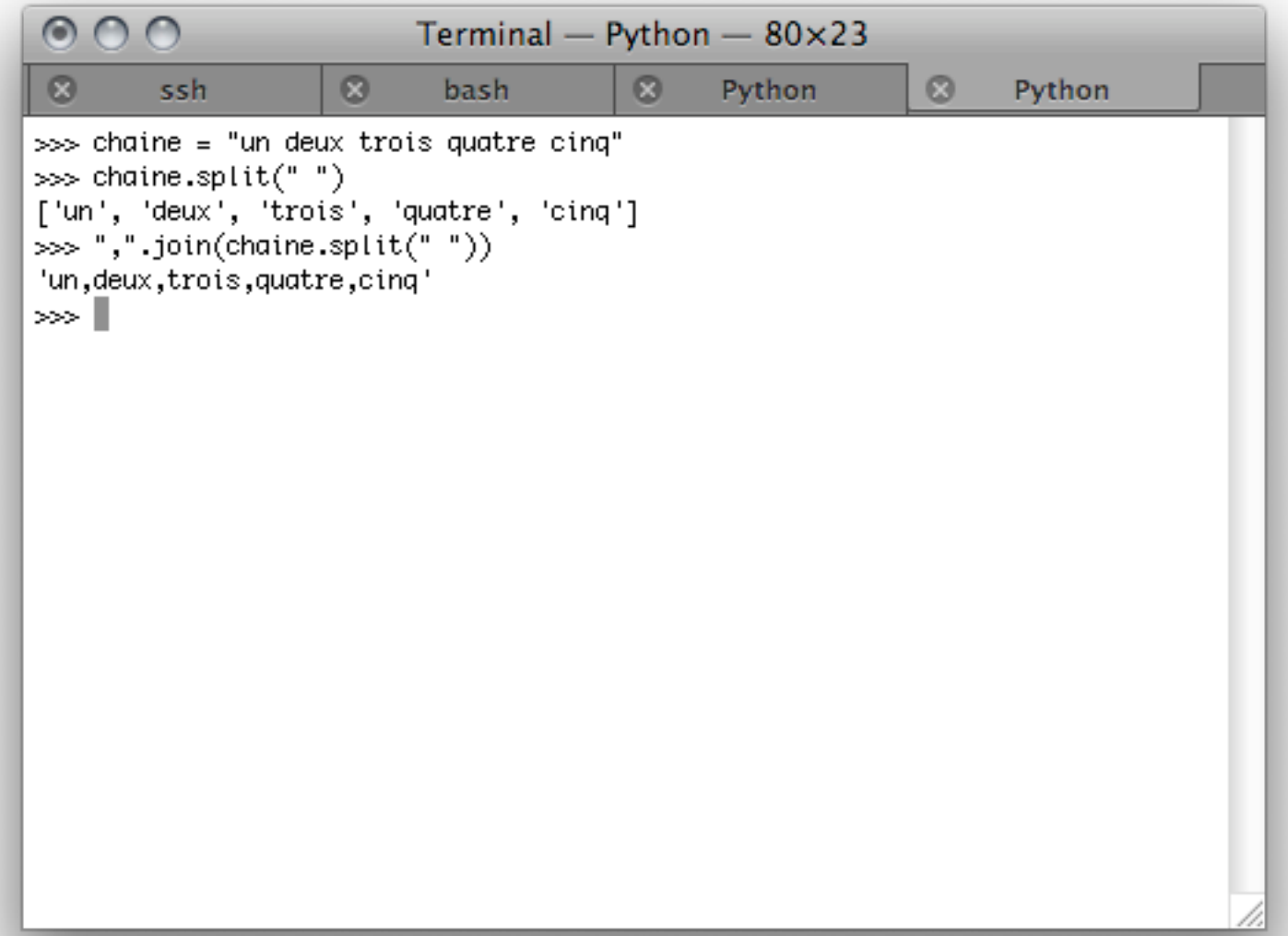
Contrôle de flux



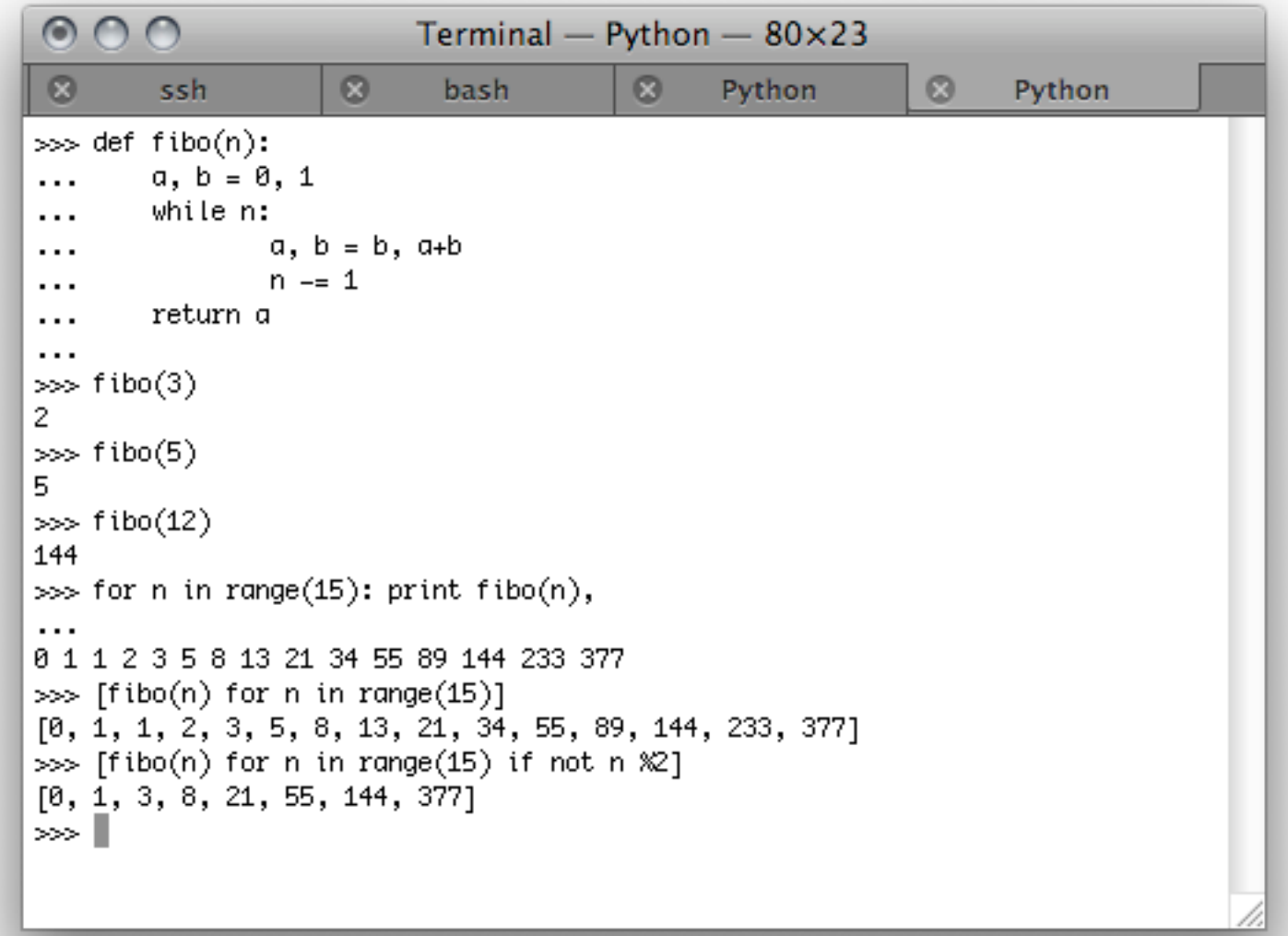
Contrôle de flux



Contrôle de flux



Fonctions


Terminal — Python — 80×23
ssh
bash
Python
Python
>>> def fibo(n):
...   a, b = 0, 1
...   while n:
...   a, b = b, a+b
...   n -= 1
...   return a
...
>>> fibo(3)
2
>>> fibo(5)
5
>>> fibo(12)
144
>>> for n in range(15): print fibo(n),
...
0 1 1 2 3 5 8 13 21 34 55 89 144 233 377
>>> [fibo(n) for n in range(15)]
[0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377]
>>> [fibo(n) for n in range(15) if not n %2]
[0, 1, 3, 8, 21, 55, 144, 377]
>>>

Iterator

SquareIterator.java

```
5  public class SquareIterator {
6      private int max;
7      private int current=0;
8
9      public SquareIterator(int max) {
10         this.max=max;
11     }
12
13     public int next() {
14         current++;
15         return current*current;
16     }
17
18     public boolean hasNext() {
19         return max>current;
20     }
21
22     public static void main(String[] args) {
23         System.out.println("Square Iterator");
24         SquareIterator it = new SquareIterator(8);
25         while(it.hasNext()) {
26             System.out.println(it.next());
27         }
28     }
29 }
30
```

Line: 30 Column: 1 Java Tab Size: 4

Iterator

SquareIterator.java

```java
public class SquareIterator {
    private int max;
    private int current=0;

    public SquareIterator(int max) {
        this.max=max;
    }

    public int next() {
        current++;
        return current*current;
    }

    public boolean hasNext() {
        return max>current;
    }

    public static void main(String[] args) {
        System.out.println("Square Iterator");
        SquareIterator it = new SquareIterator(8);
        while(it.hasNext()) {
            System.out.println(it.next());
```

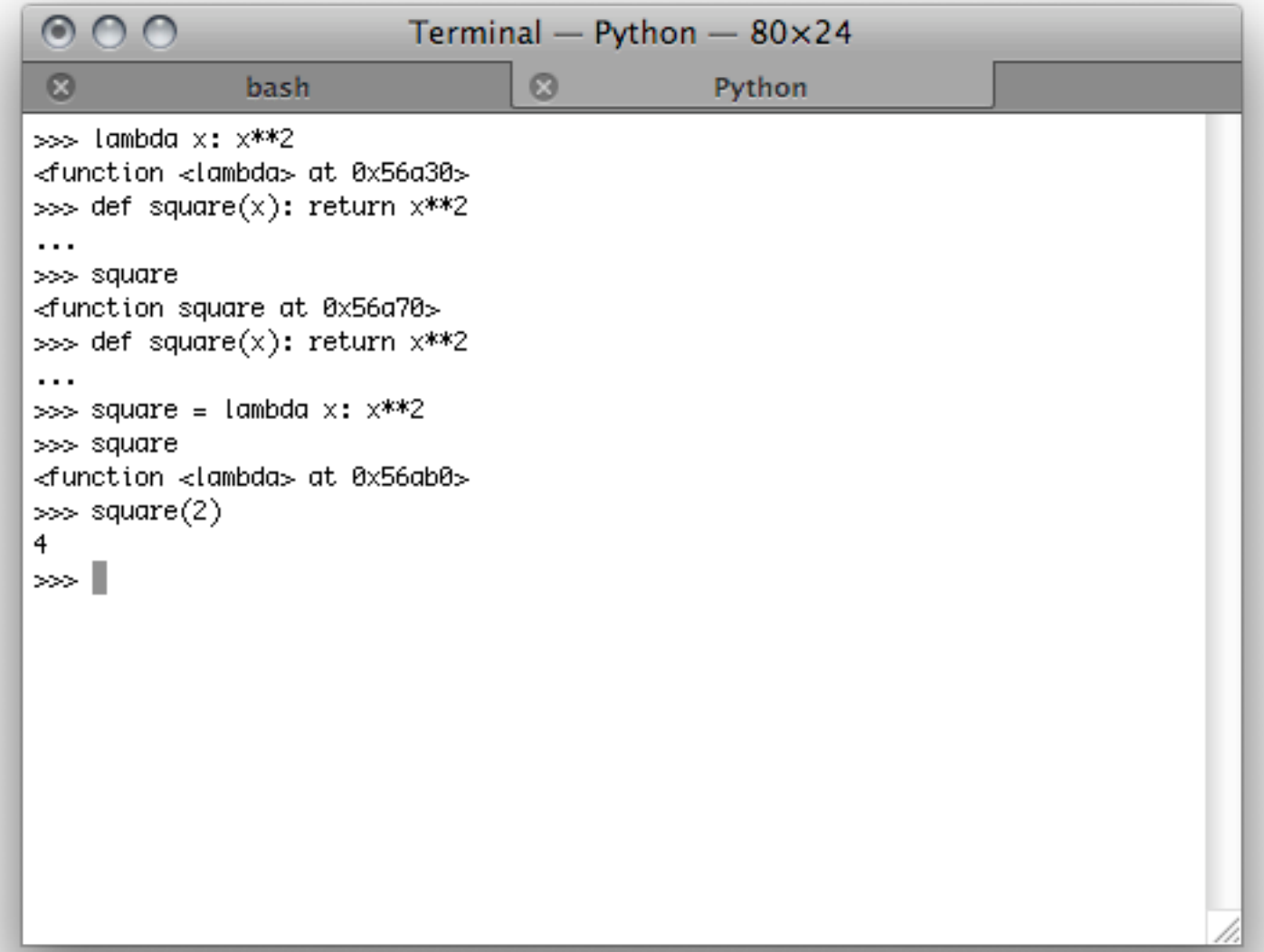
Java Tab Size: 4

untitled

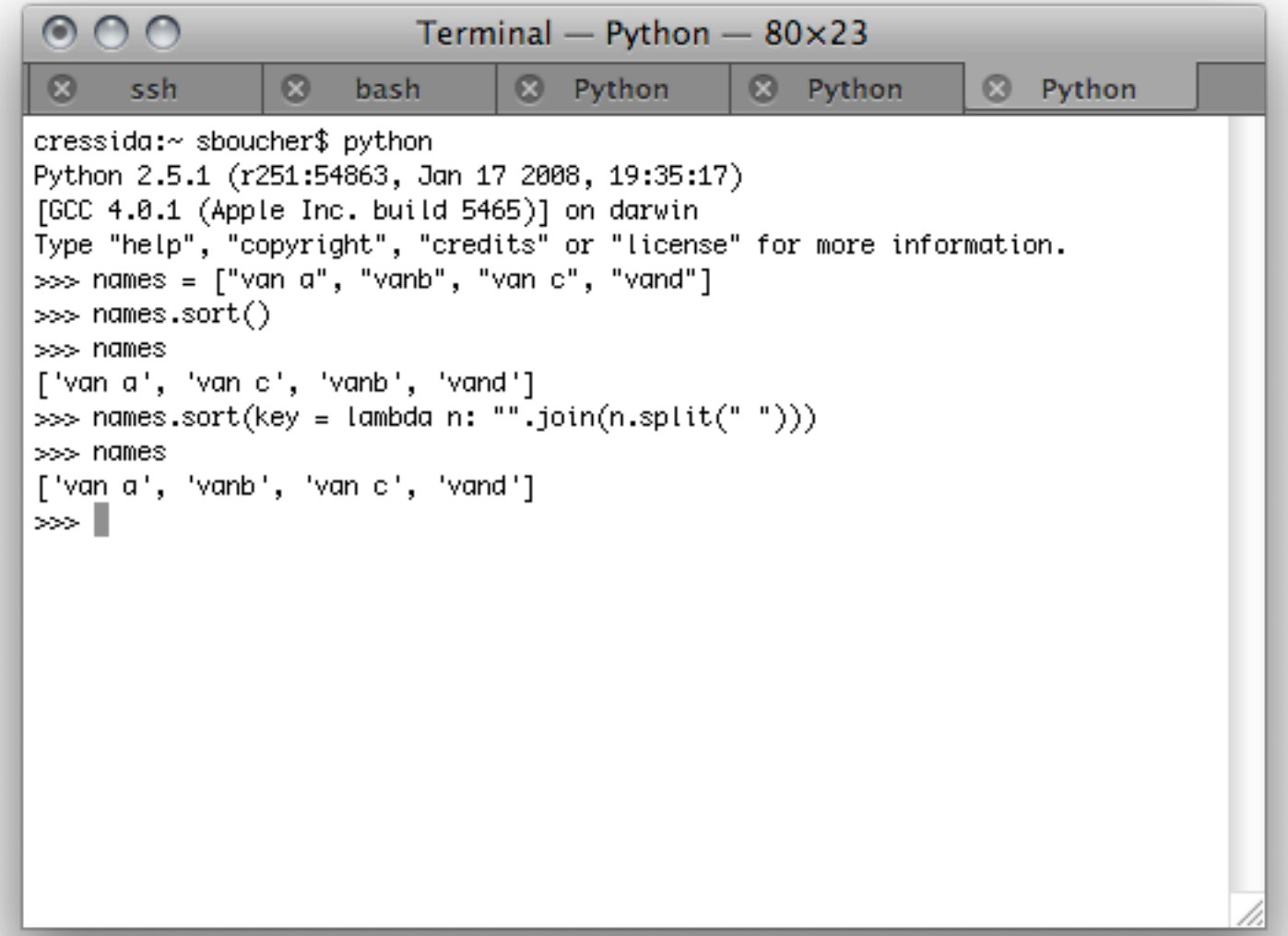
```python
def square_iterator(n):
    for i in range(1, n+1):
        yield i**2

for square in square_iterator(10): print square,
```

Line: 5 Column: 49 Python Tab Size: 4 s...

Fonctions, lambdas



Fonctions, lambdas



Plus d'information

- http://www.python.org
- http://docs.python.org/tut/
- http://www.python.org/download/