Lisp for Python Developers

Vsevolod Dyomkin
@vseloved
2013-01-26

Topics

* A short intro
* Some myths and facts
* What Lisp offers

Hello World

```
CL-USER> "Hello world"
"Hello world"

CL-USER> (print "Hello world")

"Hello world"
"Hello world"
```

Horner's Rule

(Is Lisp a functional language?)

```
(defun horner (coeffs x)
  (reduce (lambda (coef acc)
            (+ (* acc x) coef))
          coeffs
          :from-end t
          :initial-value 0))
```

From Rosetta Code:
http://rosettacode.org/wiki/Horner%27s_rule_for_polynomial_evaluation

Horner's Rule

(Python version)

```
def horner(coeffs, x):
  return reduce(lambda acc, c: \
                         acc * x + c,
                    reversed(coeffs),
                    0)
```

Lisp Tools

1. Implementations - multiple
2. IDEs - SLIME, SLIMV
3. quicklisp

Myths

* The syntax myth
* The libraries myth
* The community myth
* The slowness myth
* The outdatedness myth
* The myth of too much choice
* The myth of academic language

Myth | Fact

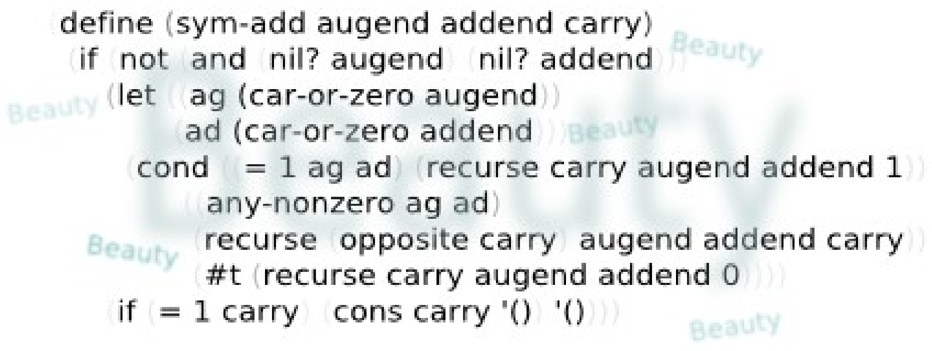
LISP = Lost In Stupid Parenthesis

Your eyes are gonna bleed!!!!

))))))))]))))))])]) — the joy of debugging ASTs

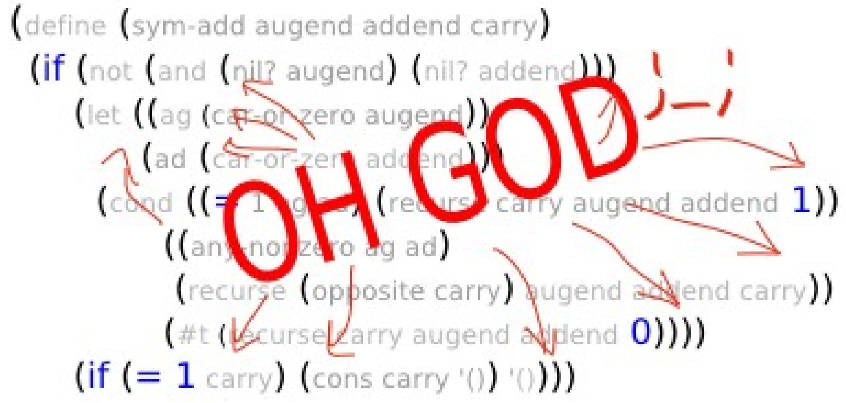
-- Armin Ronacher (https://twitter.com/mitsuhiko/status/88527153158815744)

What I see



```
(define (sym-add augend addend carry)
  (if (not (and (nil? augend) (nil? addend)))
    (let ((ag (car-or-zero augend))
          (ad (car-or-zero addend)))
      (cond ((= 1 ag ad) (recurse carry augend addend 1))
            ((any-nonzero ag ad)
             (recurse (opposite carry) augend addend carry))
            (#t (recurse carry augend addend 0))))
    (if (= 1 carry) (cons carry '()) '())))
```

What the non-Lisper sees



```
(define (sym-add augend addend carry)
  (if (not (and (nil? augend) (nil? addend)))
    (let ((ag (car-or-zero augend))
          (ad (car-or-zero addend)))
      (cond ((= 1 ag ad) (recurse carry augend addend 1))
            ((any-nonzero ag ad)
             (recurse (opposite carry) augend addend carry))
            (#t (recurse carry augend addend 0))))
    (if (= 1 carry) (cons carry '()) '())))
```

Myth	Fact
There are no Lisp libraries	1961 libraries in Quicklisp

Myth	Fact
There are no Lisp libraries	1961 libraries in Quicklisp
	27413 libraries in PyPI

```
(defun graylog (message &key level backtrace file line-no)
 (let (sock)
   (unwind-protect
        (let ((msg (salza2:compress-data
                    (babel:string-to-octets
                     (json:encode-json-to-string
                      #{:version "1.0" :facility "lisp"
                        :host (get-hostname)
                        :|short_message| message
                        :|full_message| backtrace
                        :timestamp (local-time:timestamp-to-unix
                                    (local-time:now))
                        :level level :file file :line line-no
                       })
                     :encoding :utf-8)
                    'salza2:zlib-compressor)))
          (setf sock (usocket:socket-connect *graylog-host*
                                              *graylog-port*
                                              :protocol :datagram
                                              :element-type 'ub8))
          (usocket:socket-send sock msg (length msg))))
    (usocket:socket-close sock)))
```

Myth | Fact

There's no Lisp programmers

There's no Lisp jobs

Lisp community is full of trolls

Lisp hasn't been developed for years

SBCL – more than 10 people contribute to each release, ABCL – more than 5, etc.

This year: 2 Lisp conferences in Europe for up to 100 people (ECLM & ECLS) & numerous Lisp user groups meetings

http://lisp-univ-etc.blogspot.com/search/label/lisp-hackers

Myth | Fact

Lisp is slow

Lisp is the fastest of popular dynamic languages

Lisp can be faster than Java

… or even C

http://benchmarksgame.alioth.debian.org/u32q/lisp.php

Cool Lisp Features

1. Macros
2. All the functional stuff
3. CLOS multimethods
4. Special variables
5. Condition system
6. Read macros
7. The richest calling convention
8. Etc.

Macros

This page is intentionally left blank

Condition system

```
Jedis redis = Redis.getClient(Redis.SOME_DB);
try {
    while (!Thread.interrupted())
        try {
            // do some useful work
            break;
        } catch (JedisException e) {
            Redis.returnBrokenClient(redis);
            redis = Redis.getClient(Redis.SOME_DB);
        }
    // do some other stuff
} finally {
     Redis.returnClient(redis);
}
```

Condition system

```
Jedis redis = Redis.getClient(Redis.SOME_DB);
try {
    while (!Thread.interrupted())
        try {
            // do some useful work
            break;
        } catch (JedisException e) {
            Redis.returnBrokenClient(redis);
            redis = Redis.getClient(Redis.SOME_DB);
        }
    // do some other stuff
} finally {
     Redis.returnClient(redis);
}
```

Condition system

```
(with-persistent-connection (host port)
  ;; do useful stuff
  )
```

Condition system

```
(with-persistent-connection (host port)
  ;; do useful stuff
  )

(defmacro with-persistent-connection
    ((&key (host #(127 0 0 1)) (port 6379))
     &body body)
  `(with-connection (:host ,host :port ,port)
     (handler-bind ((redis-connection-error
                     (lambda (e)
                       (warn "Reconnecting.")
                       (invoke-restart :reconnect))))
       ,@body)))
```

Condition system

```
(defmacro with-reconnect-restart (&body body)
  `(handler-case (progn ,@body)
     (usocket:connection-refused-error (e)
       (reconnect-restart-case
         (:error e :comment "Can't connect")
         ,@body))
     ((or usocket:socket-error
          stream-error end-of-file) (e)
       (reconnect-restart-case (:error e)
         ,@body))))
```

Condition system

```
(defmacro reconnect-restart-case
      ((&key error comment) &body body)
   `(if *pipelined*
        (progn ,@body)
        (restart-case (error 'redis-connection-error
                             :error ,error
                             :comment ,comment)
          (:reconnect ()
           :report "Trying to reconnect"
           (reconnect)
           ,@body))))
```

http://lisp-univ-etc.blogspot.com/2012/11/cl-redis-separation-of-concerns-in.html

Read macros

```
{ x ! x <- (loop :for i :upto 10 :collect i) }
'(0 1 2 3 4 5 6 7 8 9 10)

{x ! x <- '(1 nil 2) ! x}
'(1 2)

{x y ! x <- '(1 2 3) y <- '(5 6 7) ! (oddp x) (> y 5)}
'((3 7))

{ (+ x y) ! x <- '(1 2 3) y <- '(5 6 7) ! (oddp x) (> y 5) }
'(10)

(set-macro-character #\{ #'read-listcomp)
(set-macro-character #\} (get-macro-character #\)))
```

```lisp
(defun read-listcomp (stream char)
  "Read list comprehension { vars ! generators [! conditions]? }"
  (declare (ignore char))
  (let (rezs srcs conds state)
    (dolist (item (read-delimited-list #\} stream))
      (if (eql '! item)
          (setf state (if state :cond :src))
          (case state
            (:src       (push item srcs))
            (:cond      (push item conds))
            (otherwise (push item rezs)))))
    (setf rezs  (reverse rezs)
          srcs  (reverse srcs)
          conds (reverse conds))
    (let ((binds (mapcar #`(cons (first %) (third %))
                           (group 3 srcs))))
      `(mapcan (lambda ,(mapcar #'car binds)
                 (when (and ,@conds)
                   (list ,(if (rest rezs)
                              (cons 'list rezs)
                              (first rezs)))))
               ,@(mapcar #'cdr binds)))))
```

Lisp is Python on steroids

but w/o batteries :)

Lisp in My View

1. Rich, non-exclusive culture
2. Consistency & simplicity
3. Mature interactive environment
4. Multi-paradigm language
5. Unique features
6. Cool non-unique features
7. Performant grown-up runtimes

Lisp Resources

1. Hyperspec - http://clhs.lisp.se/Front/Contents.htm
2. Cookbook - http://cl-cookbook.sourceforge.net/
3. Cliki – http://cliki.net
4. lispdoc - http://lispdoc.com/
5. Google Styleguide - http://google-styleguide.googlecode.com/svn/trunk/lispguide.xml
6. L1sp.org - http://l1sp.org/
7. Lisp books - http://pinterest.com/vseloved/lisp-books/