PyLisp

A Lisp compiler targeting Python

Example

- `(+ 1 2)`
  `→ 3`
- `(defun square (x) (* x x))`
  `→ <function ...>`
- `(square 12)`
  `→ 144`

Generated code

- `(defun square (x) (* x x))`

```
def _(_Lx):
    _Lx = [_Lx]
    _ = f_L_2a(_Lx[0], _Lx[0])
    return _
_.__name__ = 'square'
f_Lsquare = _
```

PyLisp

- Python 2 / 3
- Macros
- Read-write captured variables
- Peephole optimizer
- ~400 lines in total (250.py + 150.lisp)
- Works fine with PyPy

Technicalities - assignments

- In Lisp there are only expressions
- In Python assignment is a statement and not an expression
- Compilation requires statement generation

Technicalities
r/w closed over variables

- In Python 2.5 closed over variables are read-only
- Python 3 added "nonlocal"
- Workaround for 2.5 is to use one-element lists instead of simple variables
- A code walking pass could optimize code to add wrapping only to captured variables or nonlocal declarations

Technicalities - optimizer

- Generated code is potentially very verbose
- Any expression may require statements (e.g. evaluation of a parameter of a function call could include an assignment)
- The regexp-based peephole optimizer detects simple cases where this is not needed

Technicalities - quoted values

- Lisp has a quote operator that is more than a "literal"
- Quoted objects are stored in a global array because identity (and not simply equality) must be preserved