

Re: Python syntax in Lisp and Scheme

Source: <http://coding.derkeiler.com/Archive/Lisp/comp.lang.lisp/2003-10/0751.html>

From: Alexander Schmolck (a.schmolck_at_gmx.net)

Date: 10/09/03

Date: 09 Oct 2003 00:11:17 +0100

Pascal Bourguignon <spam@thalassa.informatimago.com> writes:

> *Alexander Schmolck <a.schmolck@gmx.net> writes:*
> > *Why drop your files in the first place?*
>
> *Because of gravity, and because you need to move them from the storage*
> *cabinet to the card reader.*

Yes, I thought it must be a similarly frequently encountered problem.

> > *Could you maybe provide an example (ideally one where another human reader*
> > *would not be misled by your choice of indentation)?*
>
> *Take the case of prog1, prog2 and progn2. They have about the same*
> *syntactic structure. However we indent them differently:*

Hmm, I don't think standard stuff like PROGx, IF etc. forms relevant examples (either it's always indented in that way (by common convention in CL and by necessity in some hypothetical python)) or you're dealing with a misfeature (or do you e.g. think that the millions of man-hours wasted on debating, setting up and converting between various C (or worse C++) indentation schemes are a worthy price for individual freedom of indentation-preference-self-expression?).

>
> *But what I have in mind is cases where I format the source in a tabular form:*
>
> *(case state*
> *:: state column-1 column-2 column-3 column-4*
> *:: -----*
> *(:state-1 (act-1) (act-3) (act-4))*
> *(:state-2 (act-2) (act-3) (act-4))*
> *(:state-3 (act-1) (act-2)*
> *(act-1) (act-3))*
> *:: etc*
> *)*

[Well, you'd better not drop this one (unless your emacs comes with M-x ai-indent-region :)]

A better example, only that you could write something analogous (certainly as far as use of indentation is concerned) in python (verbose rendition):

```
# state column-1 column-2 column-3 column-4
stateTable = {'state1' : (func1, func3, func4),
              'state2' : ( func3, func4),
              'state3' : (func1, func2, func3,
                          thisIsSyntacticallyOKTooBTW)}
for action in stateTable[state]: action()
```

```
>
>
> One case that happens often enough, in C, would be:
>
> if(condition-1){ statement-1; final-1; }else
> if(condition-2){ statement-2-1; statement-2-2; final-2; }else
> if(condition-3){ stat-3; final-3; }
```

```
if condition1: statement1; final1
elif condition2: statement2_1; final2
else: stat3
```

Happy?

```
> In anycase, when you'll have a macro system as powerful as the one in
> Common-Lisp, I guess that you'll see reasons enough to avoid
> structuring indentation. For an example of what I mean, have a look
> at the output of cpp (gcc -E) with the expansion of a couple of cpp
> macros.
```

Well, I explicitly disclaimed that changing lisp to structuring indentation would be necessarily a good idea.

I just think the arguments why it's a bad idea for python are pretty bogus (one I think pretty good reason to back up my conclusion is empirical observation: frequent python users, including myself don't seem to experience problems with python's use of indentation [1], despite the fact that they are familiar (and often even weaned on) languages with different structure delimiters, *including* lisp and scheme. What is more, I have seen few complaints from people who have actually written significant amounts of code in python, it's mostly those who haven't used python but were unfortunately subjected to make and fortran who are certain python must be really error-prone because of whitespace etc.).

```
> My point here being that if you like python structural indentation,
> you can easy implement it in Common-Lisp.
```

Sure, I know.

> *If you like parenthesis indentation, could you as easily have them in*
> *python?*

If you mean as an alternative syntax: Nope and that's a feature.

'as

Footnotes:

[1] I will concede that there is one wart about indentation in python: tabs are only discouraged (the standard which most people adhere to is 4 spaces per level), not disallowed.