

Re: Python from Wise Guy's Viewpoint

Source: <http://coding.derkeiler.com/Archive/Python/comp.lang.python/2003-10/5371.html>

From: Joachim Durchholz (joachim.durchholz_at_web.de)

Date: 10/30/03

Date: Thu, 30 Oct 2003 02:31:17 +0100

Joe Marshall wrote:

> Joachim Durchholz <joachim.durchholz@web.de> writes:
>
>>Type inference is about "as much static checking as possible with as
>>little annotations as absolutely necessary".
>>HM typing is extremely far on the "few declarations" side: a handful
>>even in large systems.
>
> I certainly don't mind as much static checking as possible. I get a
> little put off by **any** annotations that are **absolutely necessary**,
> though. My preference is that all *`lexically correct'* code be
> compilable, even if the object code ends up being the single
> instruction *`jmp error-handler'*. Of course I'd like to get a
> compilation warning in this case.

Then static typing is probably not for you.

Mainstream FPLs tend to require an occasional type declaration. And you'll have to know about the type machinery to interpret the type errors that the compiler is going to spit at you – never mind that these errors will always indicate a bug (unless one of those rare explicit type annotations is involved, in which case it could be a bug or a defective type annotation).

>>It sounds unbelievable, but it really works.
>
> I believe you. I have trouble swallowing claims like *`It is never
> wrong, always completes, and the resulting code never has a run-time
> error.'* or *`You will never need to run the kind of code it doesn't allow.'*

This kind of claim comes is usually just a misunderstanding.

For example, the above claim indeed holds for HM typing – for the right definitions of "never wrong" and "never has an error".

HM typing "is never wrong and never has a run-time error" in the following sense: the algorithm will never allow an ill-typed program to pass, and there will never be a type error at run-time. However, people tend to overlook the "type" bit in the "type error" term, at which point

the discussion quickly degenerates into discourses of general correctness. Adding to the confusion is the often-reported experience of functional programmers, that annotating your code with static type declarations can be a very efficient way to finding design errors soon.

The type correctness claims are backed by hard theory; the design improvement claims are of a social nature and cannot be proved (they could be verified by field studies at best).