"Steven D'Aprano" <steve at pearwood.info> wrote

> *are, more or less, analogous. In principle you could design a*
> *compiler to try guessing what you probably meant when faced with*
> *syntax errors:*

And in fact there are several such compilers for languages like C.
The Digital Equipment VAX VMS had an IDE with such a compiler
called LSE (Language Sensitive Environment) which could detect
and correct many common C syntax errors such as missing semi-colons
or using = instead of == etc. It was of course not perfect because
there
are many such xcases where its imp[ossibler to be certain if its a
mistake or a deliberate but unusual construct. But it got it right 9
times
out of 10...

Languages like ADA and Pascal with much tighter syntax rules
are even easier to correct. Open languages like Lisp and Forth
are more limited in their opportunities.

The LSE tool could be set to automatically correct or present a list
of potential corrections which the user then stepped through at the
end selecting the corrections required. LSE was quite expensive
(about $30,000 for a server license I think, compared to $2000 for
the vanilla C compiler) but it saved a lot of time on our project.

Automatically detecting/correcting semantic errors is sadly not
possible. (I have seen one compiler - for CORAL I think it was?)
which attempted to do so, but it could only spit out a list of
possible errors with what it thought were the probability of error.
And it was wrong at least as often as it was right...

Alan G.