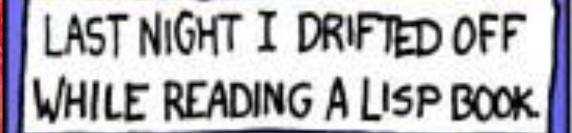
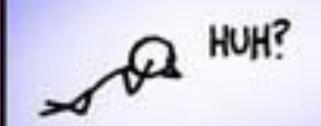


MACROS

Under the micro lens





SUDDENLY, I WAS BATHED IN A SUFFUSION OF BLUE.

AT ONCE, JUST LIKE THEY SAID, I FELT A GREAT ENLIGHTENMENT. I SAN THE NAKED STRUCTURE OF LISP CODE UNFOLD BEFORE ME.



THE PATTERNS AND METAPATTERNS DANCED.

SYNTAX FADED, AND I SWAM IN THE PURITY OF QUANTIFIED CONCEPTION. OF IDEAS MANIFEST.

TRULY, THIS WAS
THE LANGUAGE
FROM WHICH THE
GODS WROUGHT
THE UNIVERSE.





I MEAN, OSTENSIBLY, YES. HONESTLY, WE HACKED MOST OF IT TOGETHER WITH PERL.

source: https://xkcd.com/224/

MACROS

> are a way to modify a program during compile time.

- > are functions that run at compile time,
 - > unevaluated s-expressions => modified s-expression evaluated at run-time

➤ lisp = list processing

(= CODE DATA)

DIFFERENCES FROM FUNCTIONS

➤ Arguments to a function are evaluated before the function call, arguments to a macro are unevaluated s-expressions

> Functions are evaluated inside out, while macros are evaluated outside in.

➤ Macro evaluation happens at compile-time, while function execution happens at runtime.

(OR MACROS (NOT MACROS))*

Reducing syntax

➤ Compile-time optimisations

➤ Defining control structures

Designing DSLs

^{*} The use of macros is subject to a risk of increased complexity, please read the documentation carefully before investing

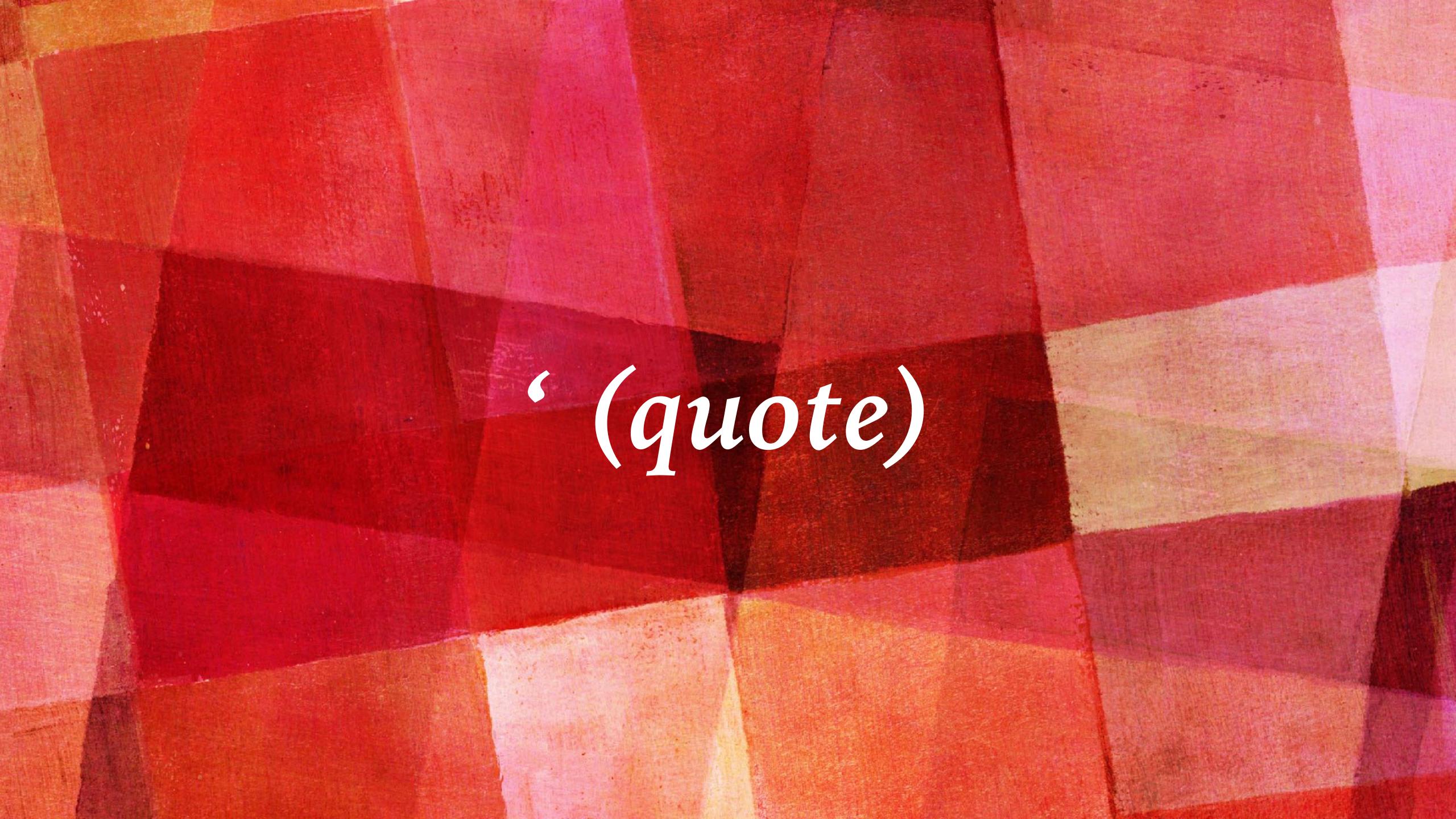
REASONS TO NOT USE MACROS

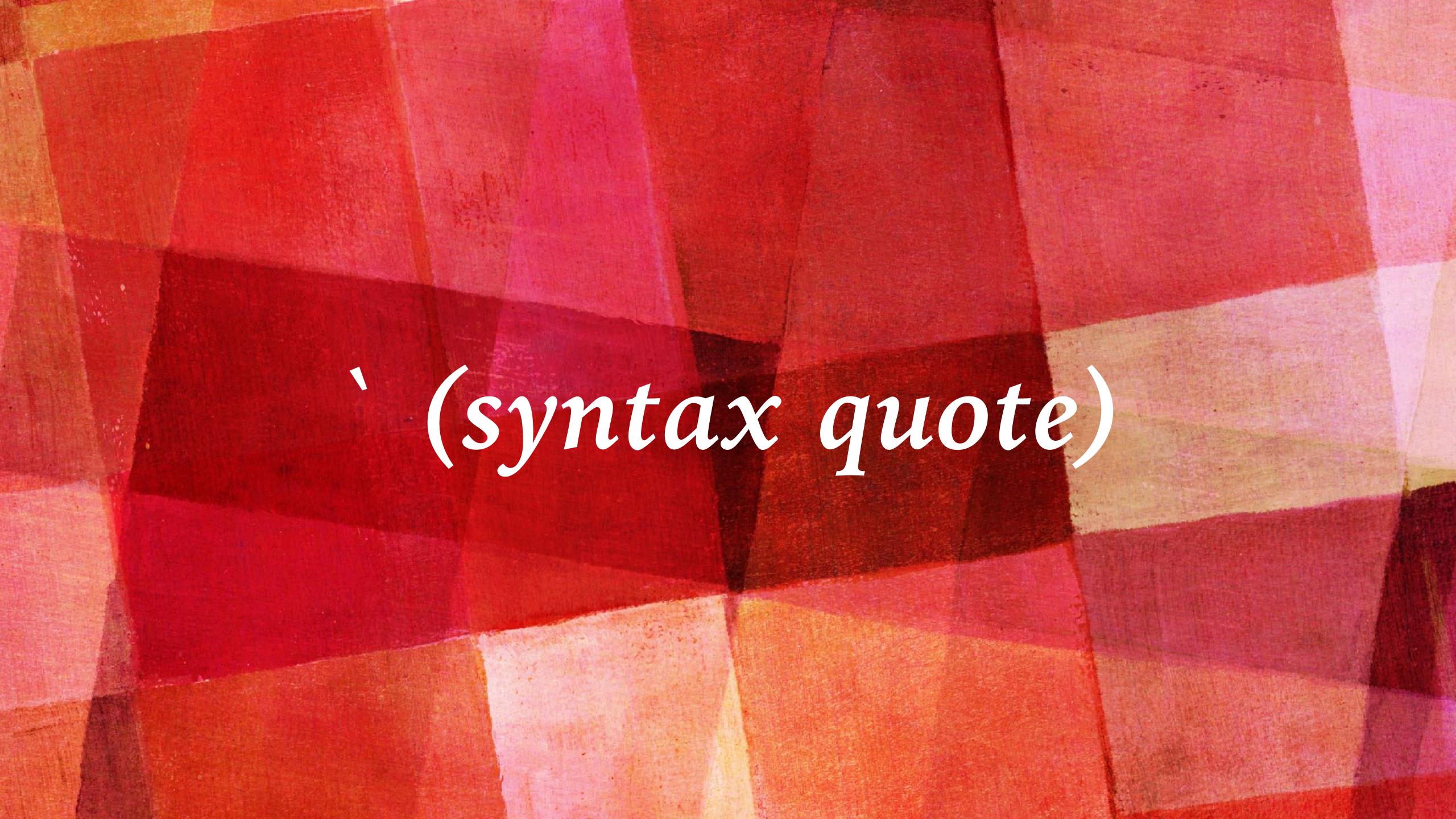
➤ Increased complexity. (Runtime vs. Compile time)

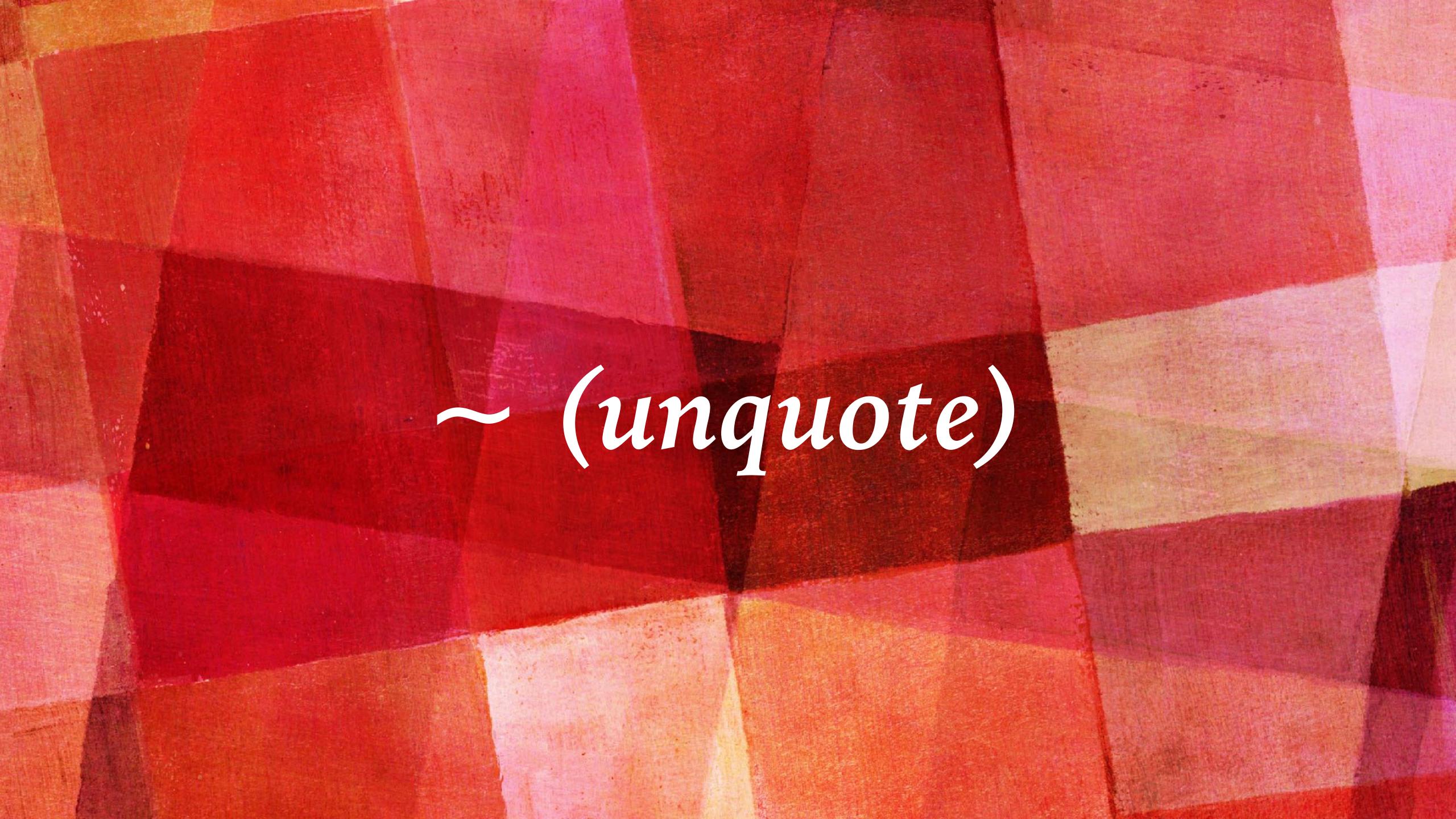
➤ Macros are not first-class in Clojure.

➤ Data > Functions > Macros

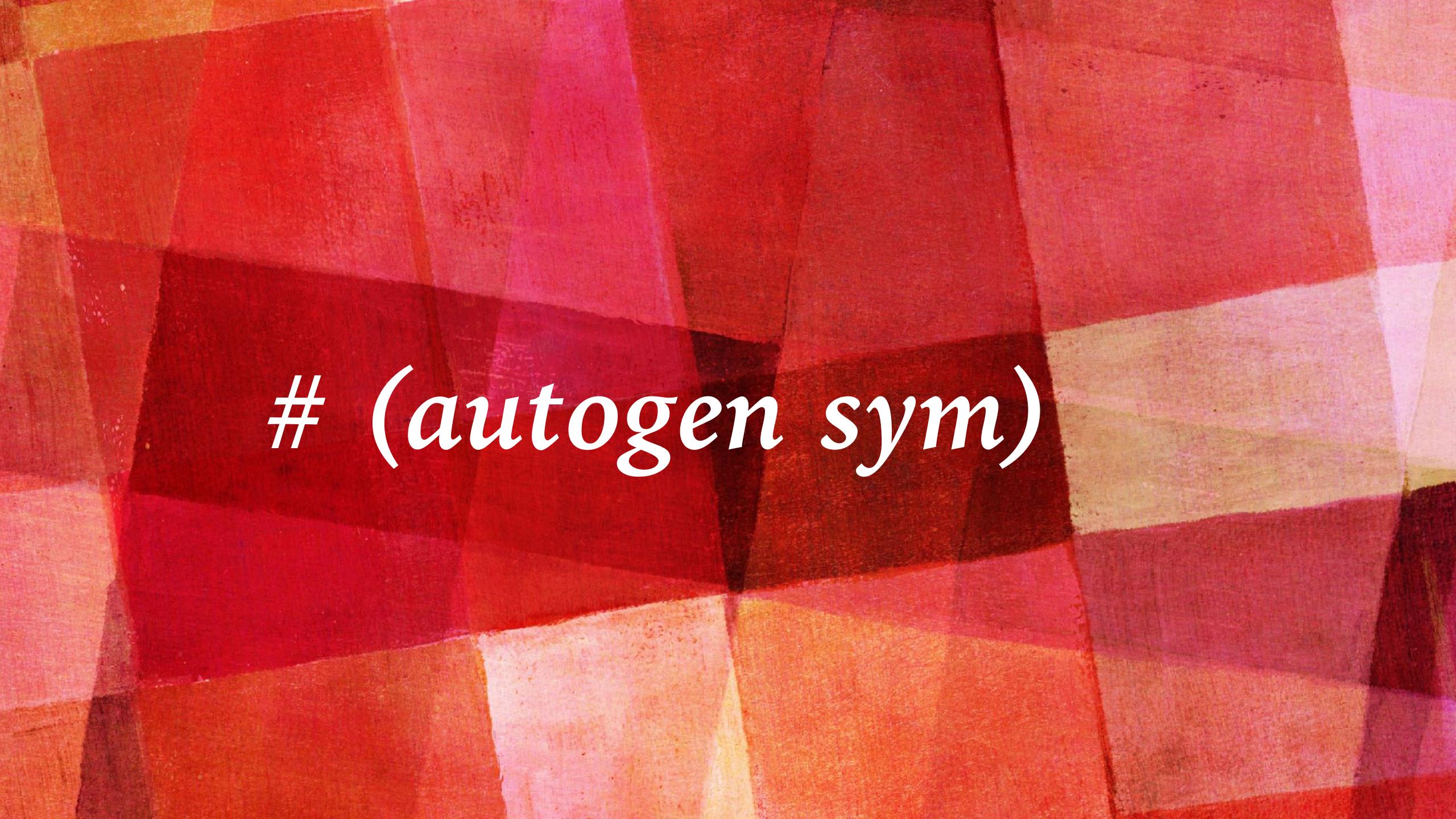












EXAMPLES

THANK YOU