Code should be ~~self-documenting~~ self-explaining,
using plain language without ***any*** cryptic abbreviations!

`A = 86400;`	***VERY BAD*** PROGRAMMING!
`A = 86400;  // SPD`	Ahah, a comment, clearly saying it is **SPD**!
`A = 86400;  // elhepnat's turnk`	But compilers ***do not check comments*** for errors in semantics, syntax, spelling, or whatever!
`SPD = 86400;`	Why did you call it **A** if it's actually **SPD**?
`SPD = 86400;  // scPrDy`	**Now** the code has become meaningful. Ahum.
`scPrDy = 86400;`	***Why*** did you call it **SPD** if it's actually **scPrDy**?
`scPrDy = 86400; // seconds per day`	Finally, we have something unambiguous!
`secondsPerDay = 86400;`	***WHY*** not named **secondsPerDay** in 1st instance?
`secondsPerDay = 24*60*60;`	Let the ***computer compute*** that silly number!

`secondsPerMinute = 60;`
`minutesPerHour = 60;`
`hoursPerDay = 24;`
	Any value that can be considered constant should have its own non-cryptic self-explaining name in PLAIN language without *any* abbreviation!

`secondsPerDay = secondsPerMinute*minutesPerHour*hoursPerDay;`	

This final self-explaining code needs ***NO*** comments and ***NO*** maintenance at all,
unless stupid politicians[1] would decide to modify the entire system of timekeeping.

[1] Do clever politicians exist?

I once had a colleague who had indeed learned to not use hard-coded values, but give them a name.
All *hir* programs started like:

```
one = 1;
two = 2;
three = 3;
```

Names should not tell the value, but explain meaning and/or purpose!

If it's an **elephant**, then ***call*** it an **elephant**!

Not an **elpnt** and certainly not **grffe** or **crcdl**.[2]

Pl. b.a.t. t.m. abrs. m.y.c. unr. & incph., th. msk. errrrs.!

[2] Isn't it perfectly clear to everyone that *crcdl.* means *hippopotamus* ≠ *elephant*?