**Rob Pike's 5 Rules of Programming**

**FROM:** [**http://users.ece.utexas.edu/~adnan/pike.html**](http://users.ece.utexas.edu/~adnan/pike.html)

* **Rule 1.** You can't tell where a program is going to spend its time. Bottlenecks occur in surprising places, so don't try to second guess and put in a speed hack until you've proven that's where the bottleneck is.
* **Rule 2.** Measure. Don't tune for speed until you've measured, and even then don't unless one part of the code overwhelms the rest.
* **Rule 3.** Fancy algorithms are slow when n is small, and n is usually small. Fancy algorithms have big constants. Until you know that n is frequently going to be big, don't get fancy. (Even if n does get big, use Rule 2 first.)
* **Rule 4.** Fancy algorithms are buggier than simple ones, and they're much harder to implement. Use simple algorithms as well as simple data structures.
* **Rule 5.** Data dominates. If you've chosen the right data structures and organized things well, the algorithms will almost always be self-evident. Data structures, not algorithms, are central to programming.

Pike's rules 1 and 2 restate Tony Hoare's famous maxim "Premature optimization is the root of all evil." Ken Thompson rephrased Pike's rules 3 and 4 as "When in doubt, use brute force.". Rules 3 and 4 are instances of the design philosophy KISS. Rule 5 was previously stated by Fred Brooks in The Mythical Man-Month. Rule 5 is often shortened to "write stupid code that uses smart objects".