Valuing Simplicity

From computing to biology and back again

Simplicity is to be highly valued in a standard.
The cheapest, most reliable code is the code that is not there.
To be reproducible, a programming language has to be very stable.
To be stable a language has to be a shared standard.
To become a standard, a language must be simple (else who would want to learn it?)
To be simple, a language must listen to beginners (how else to simplify?)
To become a standard, a language must be expressive (else what is it good for?)
To be expressive, a language must satisfy experts (who else can judge if it works correctly?)
To be expressive enough for today's biology, a language must cover a great many disciplines…
… more than letters in the alphabet from abstract algebra to zoology.
Why? Biology is extremely diverse and the best complexity taming tools can barely keep up.
Thus doubts are in order, whether any human brain can hold the necessary expertise.
Especially, since this language must be simple to ever become a standard.
And simplicity is one of the most difficult things to achieve.
Humans hate too much complexity.
Mission is impossible.
Game Over.
End

.
.
z

Zen
is the art of
finding points that matter,
more than anything else, because from these simple points the
complexity of everything else is composed by means of combination and recombination.

COMPLEXITY POISONS LIKE SMOKE OF CIGARS
2018-August-08

Biology is a prime example. Many biologists in the past could not believe that DNA was central to storing hereditary information: just 4 boring bases? no interesting observable chemical reactions? How could this possibly store so much information? Overcoming such blindspots takes time to try many alternative ideas. Computation is another prime example: 'Turing machines' are very simple,and so is the 'lambda calculus', and so is the idea to store sets in sets in sets in sets in sets ... and so on. Yet it took many years until theoretical computer science could show that these three fundamental modes of computing are equivalent and each is in principle capable of solving any problem a single computer might solve. Throw in the message-passing 'pi-calculus' and the daunting complexity of today's concurrent computing is covered too, including networks of cell-phones or of biochemical reactions. Everything ever offered by any programming language can be seen as convenient shortcuts to diverse combinations of elementary tasks for these pioneering systems. Their goal: simplify programming and reduce the time it takes to find errors. Similarly in biology: genes from 4 bases, automatically translated into proteins from 20 amino acids, and substantial efforts to correct errors. Even modern mathematics is being revolutionized by what is known as 'category theory': it brings a combinatorial approach to composing theories, finds analogous structures in surprising places, and simplifies many assumptions that seem arbitrary otherwise and that unnecessarily complicate mathematics.

The 2018 NIH data science strategy estimates that biodata scientists work 80% of the time on finding and cleaning data. Many problems they face are recurrent, and can be solved faster by a compiler that implements a general-purpose programming language for biology that is designed by biologists for biologists. Yet, such innovation depends on thinking outside of the box and crossing disciplinary boundaries in more ways than most realize. To illustrate distances between the disciplinary cutures that need to be integrated to make this work, consider the cultural mismatch that I intentionally introduced by packaging hard facts about programming efficiency into a layout often associated with poetry.It is fast and convenient to conclude that texts with such layouts are not serious. And yet, this one is! Those who cannot see past such a layout mismatch prove how much the simplifying language described here could help them and how little they will likely contribute to make it happen. Are you still with me? If yes, you earned an invitation for lunch or elquivalent, redeemable against a printout of this text. We can then discuss the next challenge that seems hard to understand for many: Structures can only become as tall as their foundations are deep. Many like tall, yet digging deep is as counter-intuitive to rising high, as developing software infrastructure is to delivering software that meets the needs of users. Some companies have understood this, but academia seems behind that curve, as developers of *useful* software are regularly penalized for their investments since Margaret Dayhoff founded bioinformatics in