Base types and compound types

- Languages typically provide a small number of “built-in” types and ways to build compound types out of simpler ones:
  - Base types examples: int, real, char, string, bool
  - Type builder examples: tuples, lists, records (won’t cover)
- Base types clutter a language definition; better to make them libraries when possible.
  - ML does this to a remarkable extent (e.g., type bool can be defined directly and if/else expressions are implemented as “syntactic sugar” built on top of the case expression)

Compound-type flavors

- Conceptually, just a few ways to build compound types:
  - “Each-of ”: A t contains a t1 *and* a t2
  - “One-of ”: A t contains a t1 *or* a t2
  - “Self-reference”: The definition of t may refer to t
- Examples:
  - tuple: int * bool
  - bool: either true or false
  - list: int list list
- Remarkable: A lot of data can be described this way.
- (optional jargon: product types, sum types, recursive types)