

Decimal.js

- [Emacs Lisp](#): supports integers of arbitrary size, starting with Emacs 27.1.
- [Erlang](#): the built-in `Integer` datatype implements arbitrary-precision arithmetic.
- [Go](#): the standard library package `math/big` [implements](#) arbitrary-precision integers (`Int` type), rational numbers (`Rat` type), and floating-point numbers (`Float` type)
- [Guile](#): the built-in `exact` numbers are of arbitrary precision. Example: `(expt 10 100)` produces the expected (large) result. Exact numbers also include rationals, so `(/ 3 4)` produces `3/4`. One of the languages implemented in Guile is [Scheme](#).
- [Haskell](#): the built-in `Integer` datatype implements arbitrary-precision arithmetic and the standard `Data.Ratio` module implements rational numbers.
- [Idris](#): the built-in `Integer` datatype implements arbitrary-precision arithmetic.
- [ISLISP](#): The ISO/IEC 13816:1997(E) [ISLISP](#) standard supports arbitrary precision integer numbers.
- [J](#): built-in *extended precision*
- [Java](#): Class `java.math.BigInteger` [\(integer\)](#). `java.math.BigDecimal` [Class \(decimal\)](#)

• Just one example of...

• “Arbitrary precision decimal floating point arithmetic”

- [Nim](#): `bigints` [and](#) multiple [GMP bindings](#).
- [OCaml](#): The `Num` [library](#) supports arbitrary-precision integers and rationals.
- [OpenLisp](#): supports arbitrary precision integer numbers.
- [Perl](#): The `bignum` [and](#) `bigrat` [pragmas](#) provide `BigNum` and `BigRational` support for Perl.
- [PHP](#): The `BC Math` [module](#) provides arbitrary precision mathematics.
- [PicoLisp](#): supports arbitrary precision integers.
- [Pike](#): the built-in `int` type will silently change from machine-native integer to arbitrary precision as soon as the value exceeds the former's capacity.
- [Prolog](#): ISO standard compatible Prolog systems can check the Prolog flag "bounded". Most of the major Prolog systems support arbitrary precision integer numbers.
- [Python](#): the built-in `int` (3.x) / `long` (2.x) integer type is of arbitrary precision. The `Decimal` class in the standard library module `decimal` has user definable precision and limited mathematical operations (exponentiation, square root, etc. but no trigonometric functions). The `Fraction` class in the module `fractions` implements rational numbers. More extensive arbitrary precision floating point arithmetic is available with the third-party "mpmath" and "bigfloat" packages.