

# Funcons-beta: Floats

The P<sub>L</sub>anCompS Project

`Funcons-beta/Values/Primitive/Floats/Floats.cbs*`

---

\*Suggestions for improvement: [plancomps@gmail.com](mailto:plancomps@gmail.com).  
Issues: <https://github.com/plancomps/CBS-beta/issues>.

## Floats

```
[ Datatype float-formats
  Funcon binary32
  Funcon binary64
  Funcon binary128
  Funcon decimal64
  Funcon decimal128
  Type floats
  Funcon float
  Funcon quiet-not-a-number
    Alias qNaN
  Funcon signaling-not-a-number
    Alias sNaN
  Funcon positive-infinity
    Alias pos-inf
  Funcon negative-infinity
    Alias neg-inf
  Funcon float-convert
  Funcon decimal-float
  Funcon float-equal
  Funcon float-is-less
  Funcon float-is-less-or-equal
  Funcon float-is-greater
  Funcon float-is-greater-or-equal
  Funcon float-negate
  Funcon float-absolute-value
  Funcon float-add
  Funcon float-subtract
  Funcon float-multiply
  Funcon float-multiply-add
  Funcon float-divide
  Funcon float-remainder
  Funcon float-sqrt
  Funcon float-integer-power
  Funcon float-float-power
  Funcon float-round-ties-to-even
  Funcon float-round-ties-to-infinity
  Funcon float-floor
  Funcon float-ceiling
  Funcon float-truncate
  Funcon float-pi
  Funcon float-e
  Funcon float-log
  Funcon float-log10
```

Floating-point numbers according to the IEEE 754 Standard (2008).

See: - <http://doi.org/10.1109/IEEESTD.2008.4610935> - [https://en.wikipedia.org/wiki/IEEE\\_754](https://en.wikipedia.org/wiki/IEEE_754)

```
Datatype float-formats ::= binary32
                        | binary64
                        | binary128
                        | decimal64
                        | decimal128
```

*Built-in Type* `floats(_ : float-formats)`

Note that for distinct formats `FF1`, `FF2`, the types `floats(FF1)` and `floats(FF2)` are not necessarily disjoint.

*Built-in Funcon* `float(FF : float-formats, _ : bounded-integers(0, 1), _ : natural-numbers, _ : integers) :  $\Rightarrow$  floats`

Each finite number is described by three integers: \* `s` = a sign (zero or one), \* `c` = a significand (or ‘coefficient’), \* `q` = an exponent. The numerical value of a finite number is  $(-1)^s * c * b^q$  where `b` is the base (2 or 10), also called radix.

The possible finite values that can be represented in a format are determined by the base `b`, the number of digits in the significand (precision `p`), and the exponent parameter `emax`: \* `c` must be an integer in the range zero through  $(b^p)-1$  (e.g., if `b`=10 and `p`=7 then `c` is 0 through 9999999); \* `q` must be an integer such that  $1 - emax \leq q + p - 1 \leq emax$  (e.g., if `p`=7 and `emax`=96 then `q` is -101 through 90).

Note that `float(FF, S, C, Q)` is not a 1-1 operation.

*Built-in Funcon* `quiet-not-a-number(FF : float-formats) : floats(FF)`

*Alias* `qNaN = quiet-not-a-number`

*Built-in Funcon* `signaling-not-a-number(FF : float-formats) : floats(FF)`

*Alias* `sNaN = signaling-not-a-number`

*Built-in Funcon* `positive-infinity(FF : float-formats) : floats(FF)`

*Alias* `pos-inf = positive-infinity`

*Built-in Funcon* `negative-infinity(FF : float-formats) : floats(FF)`

*Alias* `neg-inf = negative-infinity`

## Conversions

*Built-in Funcon* `float-convert`( $FF_1 : \text{float-formats}$ ,  $FF_2 : \text{float-formats}$ ,  $F : \text{floats}(FF_1)$ ) :  $\Rightarrow \text{floats}(FF_2)$

*Built-in Funcon* `decimal-float`( $FF : \text{float-formats}$ ,  $_ : \text{strings}$ ,  $_ : \text{strings}$ ,  $_ : \text{strings}$ ) :  $\Rightarrow \text{floats}(FF)$

`decimal-float`( $F$ , “M”, “N”, “E”) is an approximation in `floats`( $FF$ ) to the value of ‘M.N’ times 10 to the power ‘E’, where “M.N” is decimal notation (optionally-signed) for a fixed-point number and “E” is decimal notation (optionally signed) for an integer. When any argument string is invalid, the result is `quiet-not-a-number`( $F$ ).

## Comparison

*Built-in Funcon* `float-equal`( $FF : \text{float-formats}$ ,  $_ : \text{floats}(FF)$ ,  $_ : \text{floats}(FF)$ ) :  $\Rightarrow \text{booleans}$

*Built-in Funcon* `float-is-less`( $FF : \text{float-formats}$ ,  $_ : \text{floats}(FF)$ ,  $_ : \text{floats}(FF)$ ) :  $\Rightarrow \text{booleans}$

*Built-in Funcon* `float-is-less-or-equal`( $FF : \text{float-formats}$ ,  $_ : \text{floats}(FF)$ ,  $_ : \text{floats}(FF)$ ) :  $\Rightarrow \text{booleans}$

*Built-in Funcon* `float-is-greater`( $FF : \text{float-formats}$ ,  $_ : \text{floats}(FF)$ ,  $_ : \text{floats}(FF)$ ) :  $\Rightarrow \text{booleans}$

*Built-in Funcon* `float-is-greater-or-equal`( $FF : \text{float-formats}$ ,  $_ : \text{floats}(FF)$ ,  $_ : \text{floats}(FF)$ ) :  $\Rightarrow \text{booleans}$

## Arithmetic

*Built-in Funcon* `float-negate`( $FF : \text{float-formats}$ ,  $_ : \text{floats}(FF)$ ) :  $\Rightarrow \text{floats}(FF)$

*Built-in Funcon* `float-absolute-value`( $FF : \text{float-formats}$ ,  $_ : \text{floats}(FF)$ ) :  $\Rightarrow \text{floats}(FF)$

*Built-in Funcon* `float-add`( $FF : \text{float-formats}$ ,  $_ : \text{floats}(FF)$ ,  $_ : \text{floats}(FF)$ ) :  $\Rightarrow \text{floats}(FF)$

*Built-in Funcon* `float-subtract`( $FF : \text{float-formats}$ ,  $_ : \text{floats}(FF)$ ,  $_ : \text{floats}(FF)$ ) :  $\Rightarrow \text{floats}(FF)$

*Built-in Funcon* `float-multiply`( $FF : \text{float-formats}$ ,  $_ : \text{floats}(FF)$ ,  $_ : \text{floats}(FF)$ ) :  $\Rightarrow \text{floats}(FF)$

*Built-in Funcon* `float-multiply-add`( $FF : \text{float-formats}$ ,  $_ : \text{floats}(FF)$ ,  $_ : \text{floats}(FF)$ ,  $_ : \text{floats}(FF)$ ) :  $\Rightarrow \text{floats}(FF)$

*Built-in Funcon* `float-divide`( $FF : \text{float-formats}$ ,  $_ : \text{floats}(FF)$ ,  $_ : \text{floats}(FF)$ ) :  $\Rightarrow \text{floats}(FF)$

*Built-in Funcon* `float-remainder`( $FF : \text{float-formats}$ ,  $_ : \text{floats}(FF)$ ,  $_ : \text{floats}(FF)$ ) :  $\Rightarrow \text{floats}(FF)$

*Built-in Funcon* `float-sqrt`( $FF : \text{float-formats}$ ,  $_ : \text{floats}(FF)$ ) :  $\Rightarrow \text{floats}(FF)$

*Built-in Funcon* `float-integer-power`( $FF : \text{float-formats}$ ,  $_ : \text{floats}(FF)$ ,  $_ : \text{integers}$ ) :  $\Rightarrow \text{floats}(FF)$

*Built-in Funcon* `float-float-power`( $FF : \text{float-formats}$ ,  $_ : \text{floats}(FF)$ ,  $_ : \text{floats}(FF)$ ) :  $\Rightarrow \text{floats}(FF)$

## Rounding

*Built-in Funcon* `float-round-ties-to-even`(*FF* : float-formats, \_ : floats(*FF*)) :  $\Rightarrow$  integers

*Built-in Funcon* `float-round-ties-to-infinity`(*FF* : float-formats, \_ : floats(*FF*)) :  $\Rightarrow$  integers

*Built-in Funcon* `float-floor`(*FF* : float-formats, \_ : floats(*FF*)) :  $\Rightarrow$  integers

*Built-in Funcon* `float-ceiling`(*FF* : float-formats, \_ : floats(*FF*)) :  $\Rightarrow$  integers

*Built-in Funcon* `float-truncate`(*FF* : float-formats, \_ : floats(*FF*)) :  $\Rightarrow$  integers

## Miscellaneous

*Built-in Funcon* `float-pi`(*FF* : float-formats) :  $\Rightarrow$  floats(*FF*)

*Built-in Funcon* `float-e`(*FF* : float-formats) :  $\Rightarrow$  floats(*FF*)

*Built-in Funcon* `float-log`(*FF* : float-formats, \_ : floats(*FF*)) :  $\Rightarrow$  floats(*FF*)

*Built-in Funcon* `float-log10`(*FF* : float-formats, \_ : floats(*FF*)) :  $\Rightarrow$  floats(*FF*)

*Built-in Funcon* `float-exp`(*FF* : float-formats, \_ : floats(*FF*)) :  $\Rightarrow$  floats(*FF*)

*Built-in Funcon* `float-sin`(*FF* : float-formats, \_ : floats(*FF*)) :  $\Rightarrow$  floats(*FF*)

*Built-in Funcon* `float-cos`(*FF* : float-formats, \_ : floats(*FF*)) :  $\Rightarrow$  floats(*FF*)

*Built-in Funcon* `float-tan`(*FF* : float-formats, \_ : floats(*FF*)) :  $\Rightarrow$  floats(*FF*)

*Built-in Funcon* `float-asin`(*FF* : float-formats, \_ : floats(*FF*)) :  $\Rightarrow$  floats(*FF*)

*Built-in Funcon* `float-acos`(*FF* : float-formats, \_ : floats(*FF*)) :  $\Rightarrow$  floats(*FF*)

*Built-in Funcon* `float-atan`(*FF* : float-formats, \_ : floats(*FF*)) :  $\Rightarrow$  floats(*FF*)

*Built-in Funcon* `float-sinh`( $FF : \text{float-formats}, \_ : \text{floats}(FF)$ ) :  $\Rightarrow \text{floats}(FF)$

*Built-in Funcon* `float-cosh`( $FF : \text{float-formats}, \_ : \text{floats}(FF)$ ) :  $\Rightarrow \text{floats}(FF)$

*Built-in Funcon* `float-tanh`( $FF : \text{float-formats}, \_ : \text{floats}(FF)$ ) :  $\Rightarrow \text{floats}(FF)$

*Built-in Funcon* `float-asinh`( $FF : \text{float-formats}, \_ : \text{floats}(FF)$ ) :  $\Rightarrow \text{floats}(FF)$

*Built-in Funcon* `float-acosh`( $FF : \text{float-formats}, \_ : \text{floats}(FF)$ ) :  $\Rightarrow \text{floats}(FF)$

*Built-in Funcon* `float-atanh`( $FF : \text{float-formats}, \_ : \text{floats}(FF)$ ) :  $\Rightarrow \text{floats}(FF)$

*Built-in Funcon* `float-atan2`( $FF : \text{float-formats}, \_ : \text{floats}(FF), \_ : \text{floats}(FF)$ ) :  $\Rightarrow \text{floats}(FF)$