integer with a maximum value of $10^{(3\times b)}-1$, and the exponent field is ignored (it is treated as if it were zero). A NaN is in its preferred (canonical) representation if the bits G_6 through G_{w+4} are zero and the encoding of the payload is canonical.

b) If G_0 through G_4 are 11110 then r and $v=(-1)^s\times(+\infty)$. The values of the remaining bits in G, and T, are ignored. The two canonical representations of infinity have bits G_5 through $G_{w+4}=0$, and $T=0$.

c) For finite numbers, r is $(S, E-bias, C)$ and $v=(-1)^s\times 10^{(E-bias)}\times C$, where C is the leading significand digit or bits from the combination field, T, and where the biased exponent E is encoded in fields depends on whether the significand.

1) If the implementat
w bits of the expo
exponent and the d
through G_4 and T as

i) When the most s
d_0 is $8+G_4$, a valu
or 2.

ii) When the most sig
d_0 is $4G_2+2G_3+G_4$
are $2G_0+G_1$, a valu
G are 00000, 01000

The $p-1=3\times J$ decimal d
densely-packed decimal.

A canonical significand
Computational operations
non-canonical declets in op

2) Alternatively, if the impleme

i) If G_0 and G_1 together are
G_0 through G_{w+1} and the
encoding (including T).

ii) If G_0 and G_1 together are
biased exponent E is form
prefixing the 4 bits $(8+G_{w+1}$

The maximum value of the
corresponding decimal-encoded
as the payload of a NaN). If th
canonical and the value used for c

Computational operations generally pr
canonical significands in operands.

NOTE—Where k is a positive multiple of 32, for

$k = 1+5+w+t = 32\times\text{ceiling}((p+2)$
$w = k-t-6 = k/16+4$
$t = k-w-6 = 15\times k/16-10$
$p = 3\times t/10+1 = 9\times k/32-2$
$emax = 3\times 2^{(w-1)}$
$emin = 1-emax$
$bias = emax+p-2.$

— qNaN (q

An *encoding* map
representations of
store retrospective

3.3 Sets of flo

This subclause sp
encodings for spe
3.5, and the param

The set of finite
following integer p

— b = the ra
— p = the nu
— $emax$ = th
— $emin$ = the minimum exponent e

$emin$ shall be $1-emax$ for all formats.

The values of these parameters for each basic format are given in Table 3.2, in which each format is

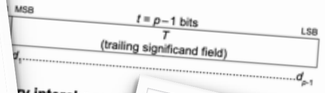
754

floating-point number has just one encoding in a binary interchange format. To make the encoding unique, in terms of the parameters in 3.3, the value of the significand m is maximized by decreasing e until either $e=emin$ or $m\geq 1$. After this process is done, if $e=emin$ and $0<m<1$, the floating-point number is subnormal. Subnormal numbers (and zero) are encoded with a reserved biased exponent value.

Representations of floating-point data in the binary interchange formats are encoded in k bits in the following three fields, shown in Figure 3.1:

$+bias$

and field digit string $T=d_1 d_2 \ldots d_{p-1}$; the leading bit of the significand,
e biased exponent E.



y interchange fo

interchange form
E shall include
clusive, to en
subnormal
and NaNs
n, and va

sNaN
$)^s\times(+$
2^{1-p}
int
ng

IEEE Std 754-2008
IEEE Standard for Floating-Point Arithmetic

3.6 Interchange format parameters

Interchange formats support the exchange of floating-point data between implementations. In each radix, the precision and range of an interchange format is defined by its size; interchange of a floating-point datum of a given size is therefore always exact with no possibility of overflow or underflow.

This standard defines binary interchange formats of widths 16, 32, 64, and 128 bits, and in general for any multiple of 32 bits of at least 128 bits. Decimal interchange formats are defined for any multiple of 32 bits of at least 32 bits.

The parameters p and $emax$ for every interchange format width are shown in Table 3.5 for binary interchange formats and in Table 3.6 for decimal interchange formats. The encodings for the interchange formats are as described in 3.4 and 3.5.2; the encoding parameters for each interchange format width are also shown in Tables 3.5 and 3.6.

Table 3.5—Binary interchange format parameters

Parameter	binary16	binary32	binary64	binary128	binary{k} ($k\geq 128$)
k, storage width in bits	16	32	64	128	multiple of 32
p, precision in bits	11	24	53	113	$k-\text{round}(4\times\log2(k))+13$
$emax$, maximum exponent e	15	127	1023	16383	$2^{(k-p-1)}-1$
Encoding parameters					
bias, $E-e$	15	127	1023	16383	*emax*
sign bit	1	1	1	1	1
w, exponent field width in bits	5	8	11	15	$\text{round}(4\times\log2(k))-13$
t, trailing significand field width in bits	10	23	52	112	$k-w-1$
k, storage width in bits	16	32	64	128	$1+w+t$

The function round() in Table 3.5 rounds to the nearest integer.

For example, binary256 would have $p=237$ and $emax=262143$.

Table 3.6—Decimal interchange format parameters

Parameter	decimal32	decimal64	decimal128	decimal{k} ($k\geq 32$)
k, storage width in bits	32	64	128	multiple of 32
p, precision in digits	7	16	34	$9\times k/32-2$
emax	96	384	6144	$3\times 2^{(k/16+3)}$
Encoding parameters				
bias, $E-q$	101	398	6176	$emax+p-2$
sign bit	1	1	1	1
$w+5$, combination field width in bits	11	13	17	$k/16+9$
t, trailing significand field width in bits	20	50	110	$15\times k/16-10$
k, storage width in bits	32	64	128	$1+5+w+t$

For example, decimal256 would have $p=70$ and $emax=1572864$.

IEEE Std 754-2008
IEEE Standard for Floating-Point Arithmetic

Decoding densely-packed decimal: Table 3.3 decodes a declet, with 10 bits $\mathbf{b}_{(9)}$ to $\mathbf{b}_{(0)}$, into 3 decimal digits $\mathbf{d}_{(1)}$, $\mathbf{d}_{(2)}$, $\mathbf{d}_{(3)}$. The first column is in binary and an "x" denotes a "don't care" bit. Thus all 1024 possible 10-bit patterns shall be accepted and mapped into 1000 possible 3-digit combinations with some redundancy.

Table 3.3—Decoding 10-bit densely-packed decimal to 3 decimal digits

$\mathbf{b}_{(6)}$, $\mathbf{b}_{(7)}$, $\mathbf{b}_{(8)}$, $\mathbf{b}_{(3)}$, $\mathbf{b}_{(4)}$	$\mathbf{d}_{(1)}$	$\mathbf{d}_{(2)}$	$\mathbf{d}_{(3)}$
0 x x x x	$4b_{(9)}+2b_{(1)}+b_{(2)}$	$4b_{(3)}+2b_{(4)}+b_{(5)}$	$4b_{(7)}+2b_{(8)}+b_{(9)}$
1 0 0 x x	$4b_{(9)}+2b_{(1)}+b_{(2)}$	$8+b_{(5)}$	$8+b_{(9)}$
1 0 1 x x	$4b_{(9)}+2b_{(1)}+b_{(2)}$	$4b_{(3)}+2b_{(4)}+b_{(5)}$	$4b_{(7)}+2b_{(8)}+b_{(9)}$
1 1 0 x x	$8+b_{(2)}$	$4b_{(3)}+2b_{(4)}+b_{(5)}$	$4b_{(9)}+2b_{(1)}+b_{(9)}$
1 1 1 0 0	$8+b_{(2)}$	$8+b_{(5)}$	$4b_{(9)}+2b_{(1)}+b_{(9)}$
1 1 1 0 1	$4b_{(9)}+2b_{(1)}+b_{(2)}$	$4b_{(3)}+2b_{(4)}+b_{(5)}$	$8+b_{(9)}$
1 1 1 1 0	$4b_{(9)}+2b_{(1)}+b_{(2)}$	$8+b_{(5)}$	$8+b_{(9)}$
1 1 1 1 1	$8+b_{(2)}$	$8+b_{(5)}$	$8+b_{(9)}$

al: Table 3.4 encodes 3 decimal digits $\mathbf{d}_{(1)}$, $\mathbf{d}_{(2)}$, and $\mathbf{d}_{(3)}$, each having 4 bits
bscript $\mathbf{d}_{(1,0:3)}$, $\mathbf{d}_{(2,0:3)}$, and $\mathbf{d}_{(3,0:3)}$, where bit 0 is the most significant
h 10 bits $\mathbf{b}_{(9)}$ to $\mathbf{b}_{(0)}$. Computational operations generate only
3.4.

10-bit densely-packed decimal

, $\mathbf{b}_{(4)}$, $\mathbf{b}_{(5)}$	$\mathbf{b}_{(6)}$	$\mathbf{b}_{(7)}$, $\mathbf{b}_{(8)}$, $\mathbf{b}_{(9)}$
$d_{(2,1:3)}$	0	$d_{(3,1:3)}$
$d_{(2,1:3)}$	1	0, 0, $d_{(3,3)}$
$d_{(3,1:2)}$, $d_{(2,3)}$	1	0, 1, $d_{(3,3)}$
1, 0, $d_{(2,3)}$	1	1, 1, $d_{(3,3)}$
$d_{(2,1:3)}$	1	1, 0, $d_{(3,3)}$
0, 1, $d_{(2,3)}$	1	1, 1, $d_{(3,3)}$
0, 0, $d_{(2,3)}$	1	1, 1, $d_{(3,3)}$
1, 1, $d_{(2,3)}$	1	1, 1, $d_{(3,3)}$

the form 01x11x111x, 10x11x111x, or 11x11x111x (where an "x"
generated in the result of a computational operation. However, as listed
as do map to values in the range 0 through 999. The bit pattern in a Na
ect how the NaN is propagated (see 6.2).

IEEE Std 754-2008
IEEE Standard for Floating-Point Arithmetic

Annex A
(informative)
Bibliography

The following documents might be helpful to the reader.

[B1] ANSI INCITS 4–1986 Information Systems—Coded Character Sets—7-bit American National Standard Code for Information Interchange (7-Bit ASCII).

[B2] Boldo, S., and Muller, J.-M., "Some functions computable with a fused-mac", *Proceedings of the 17th IEEE Symposium on Computer Arithmetic*, ISBN 0-7695-2366-8, pp. 52–58, IEEE Computer Society, 2005.

[B3] Bruguera, J. D., and Lang, T., "Floating-point Fused Multiply-Add: Reduced Latency for Floating-Point Addition", *Proceedings of the 17th IEEE Symposium on Computer Arithmetic*, ISBN 0-7695-2366-8, pp. 42–51, IEEE Computer Society, 2005.

[B4] Coonen, J. T., "Contributions to a Proposed Standard for Binary Floating-point Arithmetic", PhD thesis, University of California, Berkeley, 1984.

[B5] Cowlishaw, M. F., "Densely-Packed Decimal Encoding", *IEE Proceedings—Computers and Digital Techniques*, Vol. 149 #3, ISSN 1350-2387, pp. 102–104, IEE, London, 2002.

[B6] Cowlishaw, M. F., "Decimal Floating-Point: Algorism for Computers", *Proceedings of the 16th IEEE Symposium on Computer Arithmetic*, ISBN 0-7695-1894-X, pp. 104–111, IEEE Computer Society, 2003.

[B7] Demmel, J. W., and Li., X., "Faster numerical algorithms via exception handling", *IEEE Transactions on Computers*, 43(8): pp. 983–992, 1994.

[B8] de Dinechin, F., Ershov, A., and Gast, N., "Towards the post-ultimate libm", *Proceedings of the 17th IEEE Symposium on Computer Arithmetic*, ISBN 0-7695-2366-8, pp. 288–295, IEEE Computer Society, 2005.

[B9] de Dinechin, F., Lauter, C. Q., and Muller, J.-M., "Fast and correctly rounded logarithms in double-precision", *Theoretical Informatics and Applications*, 41, pp. 85–102, EDP Sciences, 2007.

[B10] Higham, N. J., *Accuracy and Stability of Numerical Algorithms*, 2nd edition, ISBN 0-89871-521-0, Society for Industrial and Applied Mathematics (SIAM), 2002.

[B11] IEC 60559:1989, Binary floating-point arithmetic for microprocessor systems (previously designated IEC 559:1989).

[B12] ISO/IEC 9899:1999(E) Programming languages—C (C99).

[B13] Kahan, W., "Branch Cuts for Complex Elementary Functions, or Much Ado About Nothing's Sign Bit", *The State of the Art in Numerical Analysis*, (Eds. Iserles and Powell), Clarendon Press, Oxford, 1987.

[B14] Lefèvre, V., "New results on the distance between a segment and Z^2. Application to the exact rounding", *Proceedings of the 17th IEEE Symposium on Computer Arithmetic*, ISBN 0-7695-2366-8, pp. 68–75, IEEE Computer Society, 2005.

[B15] Lefèvre, V., and Muller, J.-M., "Worst cases for correct rounding of the elementary functions in double precision", *Proceedings of the 15th IEEE Symposium on Computer Arithmetic*, ISBN 0-7695-1150-3, pp. 111–118, IEEE Computer Society, 2001.

[B16] Markstein, P., *IA-64 and Elementary Functions: Speed and Precision*, ISBN 0-13-018348-2, Prentice Hall, Upper Saddle River, NJ, 2000.

[B17] Montoye, R. K., Hokenek, E., and Runyou, S. L., "Design of the IBM RISC System/6000 floating-point execution unit", *IBM Journal of Research and Development*, 34(1), pp. 59–70, 1990.

...mat, in a particular radix, by implementing one or more
...d as both a supported arithmetic format and a supported interchange format.

01001011001101001001101100011011

$$-1^{0}$$

$$\times\ 1.01101001001101101000110111_{2}$$

$$\times\ 2^{10010110_{2}\ -\ 127}$$

* numbers very close to zero as well as some special cases work differently to this

```
$ 0.1 + 0.2
0.30000000000000004
```

```
$ 0 * -1
-0
$ x == x
false
$
```

Andrew

@andrewt@mathstodon.xyz

IEEE floating point numbers are named after the noise mathematicians make when you explain how they work

Jan 31, 2024, 10:55 · · Web · 220 · 383

```
$ 0 / 0
```

```
$ 👵 == 👵
false
$
```

 = simulation1()

 = simulation2()

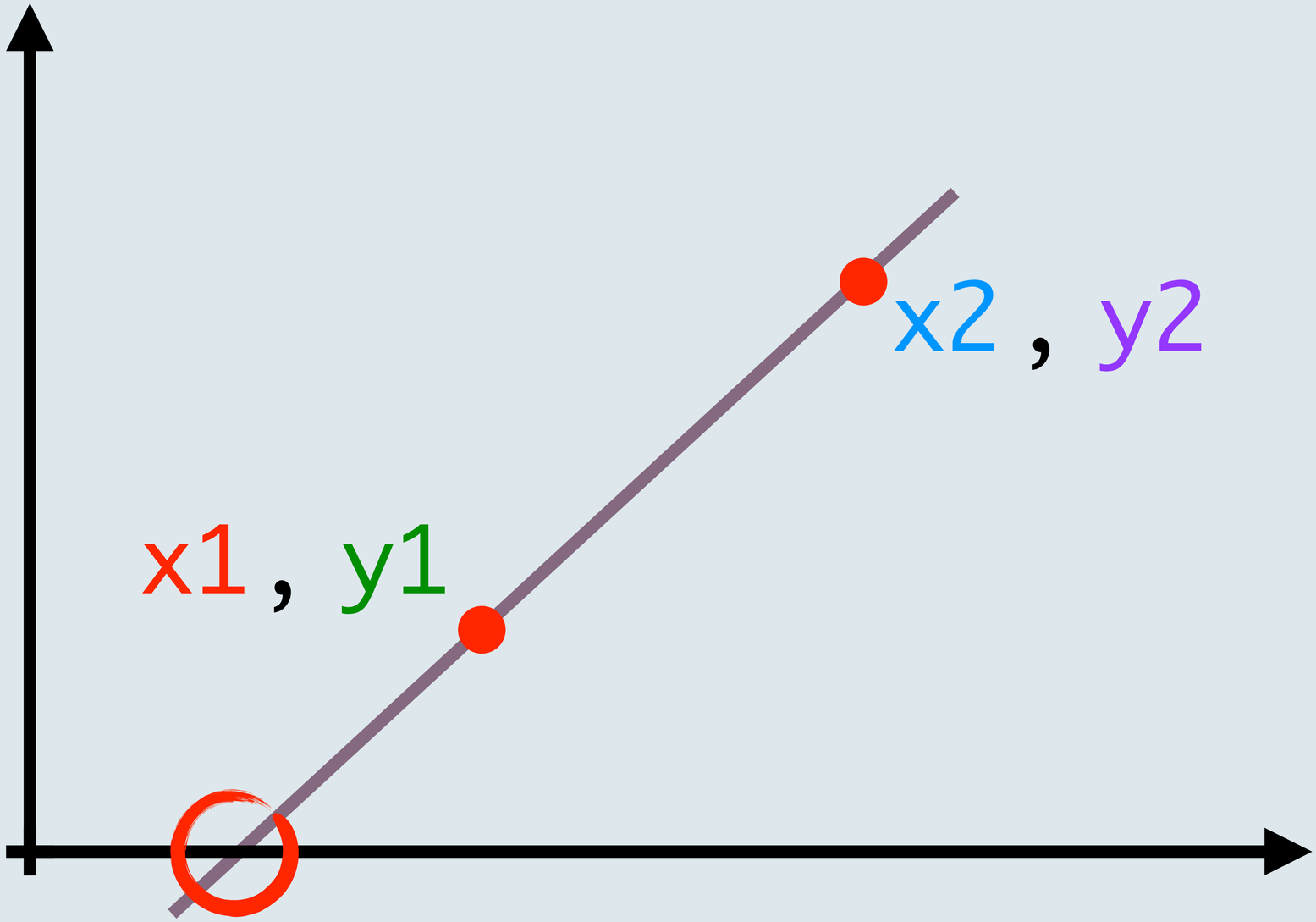
if  ==  :

    launchSpacecraft()

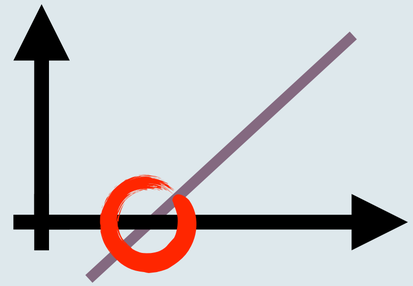
$  + 1

$ 0 * 

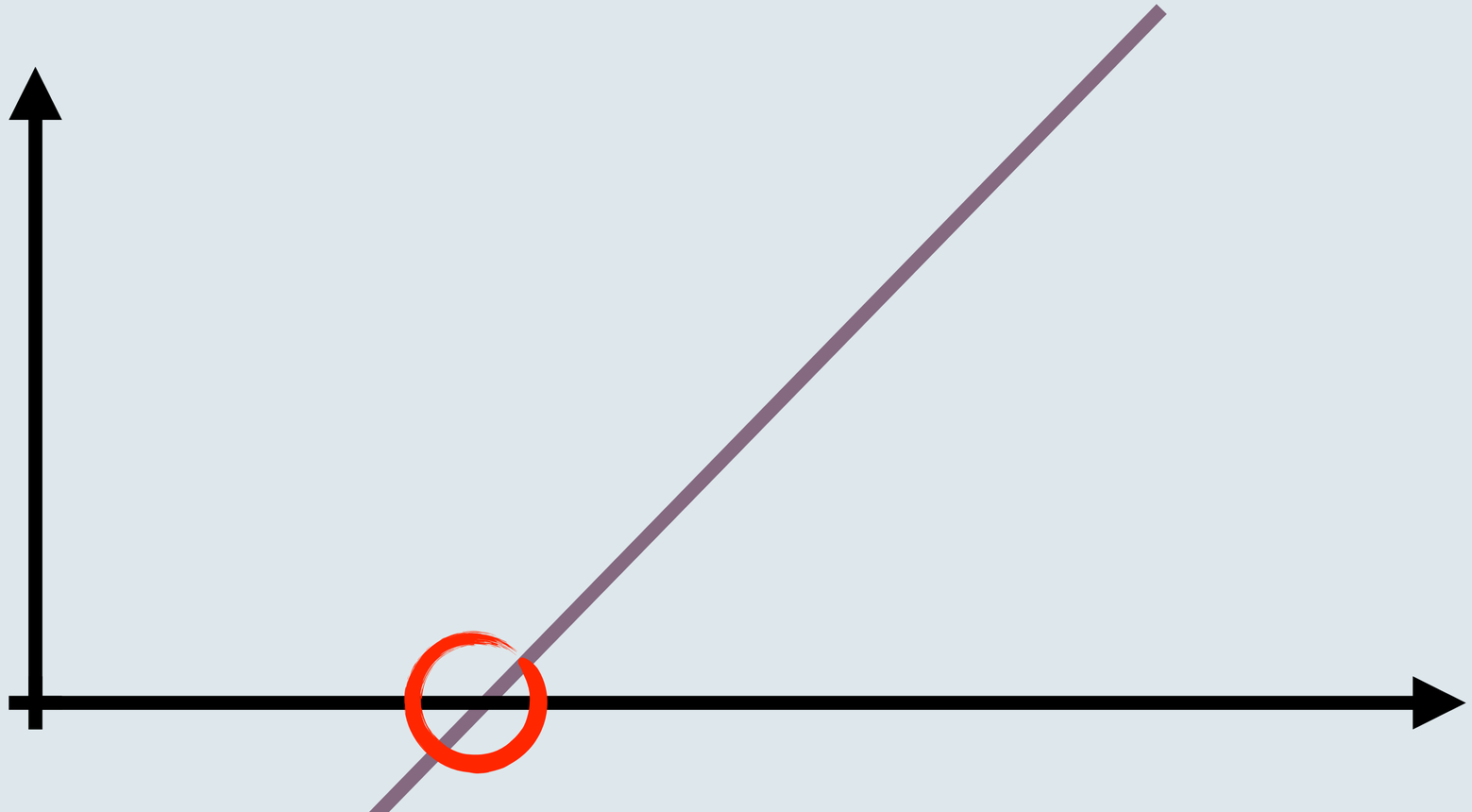
```
$ ∞ / 10000
∞
$ 1 / ∞
0
$
```


x2 , y2
x1 , y1

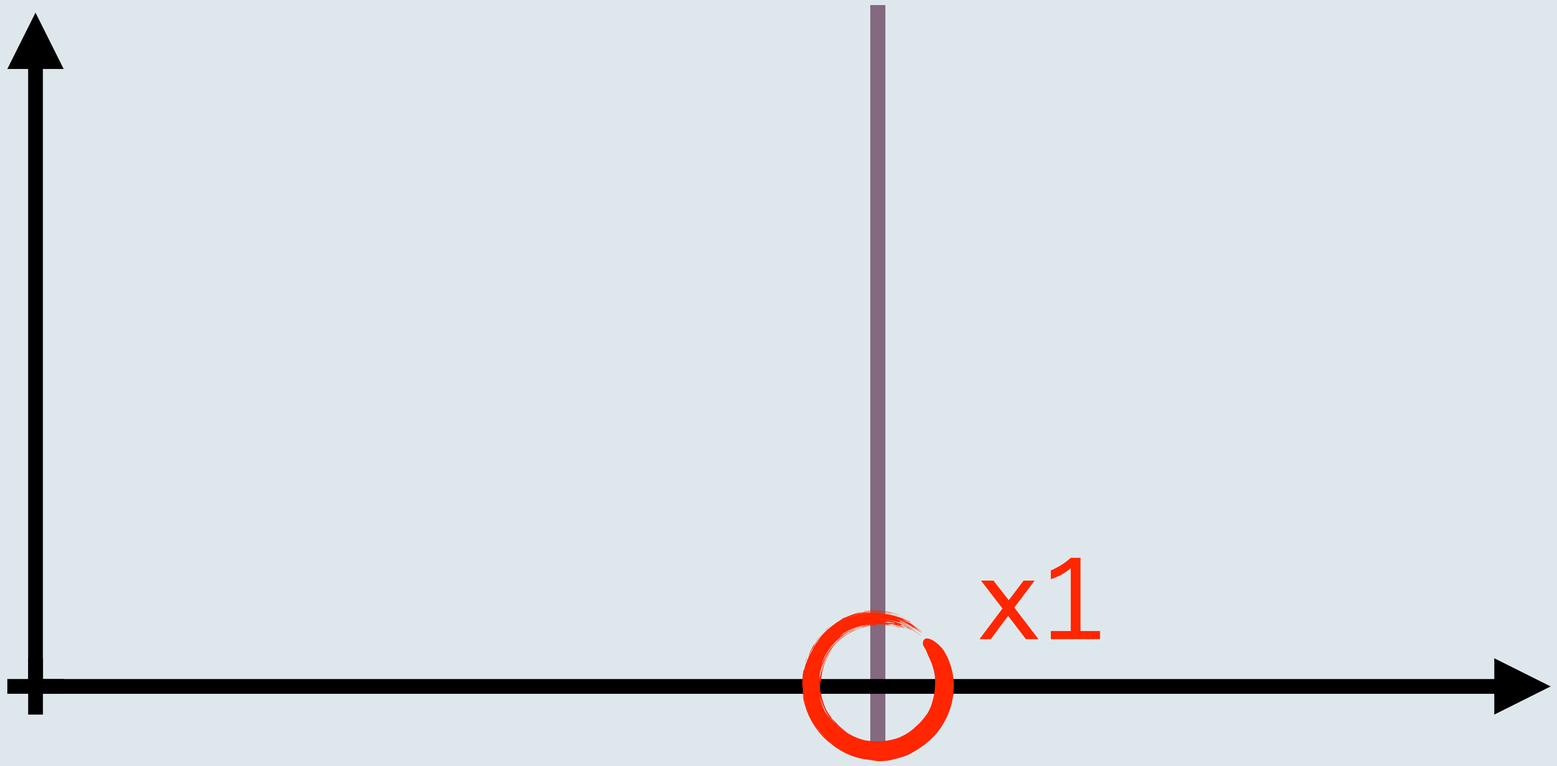
```
def intercept(x1, y1, x2, y2):
    if x1 == x2:
        return x1
    gradient = (y2 - y1) / (x2 - x1)
    if gradient == 0:
        return None
    return x1 - (y1 / gradient)
```



```
function intercept(x1, y1, x2, y2) {
    gradient = (y2 - y1) / (x2 - x1)
    return x1 - (y1 / gradient)
}
```



```
function intercept(x1, y1, x2, y2) {
  gradient = (y2 - y1) / 0
  return x1 - (y1 / ∞)
}
```



```
function intercept(x1, y1, x2, y2) {
  gradient = 0
  return x1 - (y1 / 0)
}
```

```
$ -1 / 0
-∞
$ 1 / -∞
```

```
£ 1 / -∞
0
£ 1 / (1 / -∞)
+∞
£
```

```
$ -1 / 0
-∞
$ 1 / -∞
-0
$
```

```
$ -0 == 0
true
$ (1 / -0) == (1 / 0)
false: -∞ ≠ ∞
$
```

∞ Positive Infinity

A mind-bogglingly large number

-∞ Negative Infinity

A mind-bogglingly large number

0 Zero

Basically but like, maybe slightly positive

-0 Minus Zero

Basically but gun to my head I'd guess it was slightly negative

Not a Number

No idea, sorry

¯_  _/¯

Good luck, babe

follow me on mathstodon I guess

@andrewt@mathstodon.xyz

Andrew
@andrewt@mathstodon.xyz

IEEE floating point numbers are named after the noise mathematicians make when you explain how they work

Jan 31, 2024, 10:55 · · Web · 220 · 383

If I think of any other resources to link to they'll be here, otherwise this is probably a QR code for Rick Astley or something: