Number types!

Signed integers

Type name	Number of bits	Min	Max
`i8` or `byte`	8	-128 (-2^7)	127 ($2^7 - 1$)
`i16` or `short`	16	-32768 (-2^{15})	32767 ($2^{15} - 1$)
`i32` or `int`	32	-2147483648 (-2^{31})	2147483647 ($2^{31} - 1$)
`i64` or `long`	64	-9223372036854775808 (-2^{63})	9223372036854775807 ($2^{63} - 1$)

Signed integers

✅ Good when

- Working with whole numbers

❌ Avoid when

- You need fractional numbers
- You don't need to represent negative numbers

⚠️ Watch out!

- Pick your range carefully, err on the large side
- Use `addExact` if you need to be careful about overflows and underflows

Unsigned integers

Type name	Number of bits	Min	Max
`u8` or `ubyte`	8	0	255 (2^8 - 1)
`u16` or `ushort`	16	0	65535 (2^{16} - 1)
`u32` or `uint`	32	0	4294967295 (2^{32} - 1)
`u64` or `ulong`	64	0	18446744073709551615 (2^{64} - 1)

Unsigned integers

✅ Good when

- Working with concepts where negative numbers make no sense (length of an array, count of occurrences)

❌ Avoid when

- When you need fractional or negative numbers

⚠️ Watch out!

- Overflows and underflows happen silently
- Underflows are a lot closer to the range of values we use day-to-day
- Look for `addExact` if you can

Floating point numbers

Floating point numbers

sign × fraction × 2^{exponent}

Type name	Exponent length	Fraction length	Sign length	Total length
single precision (`float`)	8 bits	23 bits	1 bit	32 bits
double precision (`double`)	11 bits	52 bits	1 bit	64 bits

Floating point numbers

- `0` = $1 \times 0 \times 2^{0}$
- `1` = $1 \times 1 \times 2^{0}$
- `-4` = $-1 \times 1 \times 2^{2}$ (sorta)
- `0.5` = $1 \times 1 \times 2^{-1}$
- `0.1` = ???

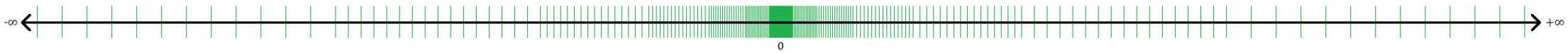
Floating point numbers

`MiniFloat`

$\text{sign} \times \text{fraction} \times 2^{\text{exponent}}$

- Fraction between 2 and -2
- Exponent between -1 and 2

Floating point numbers



Floating point numbers

✅ Good when

- Data for real-world measurements (temperature, length, mass etc)
- When you need *really fast* calculations

❌ Avoid when

- Numbers are always integers
- Cases where the exact value is important and can't be expressed by a float (money)

Floating point numbers

⚠️ Watch out!

- Sometimes (eg Javascript) you have no other options, so be careful!
- `NaN` can be unintuitive
- Sometimes you need to lookout for `-0`
- Overflows don't throw exceptions
- For very big values, floats don't behave how you'd expect

Decimals

sign × fraction × 10^{exponent}

- `0` = $1 \times 0 \times 10^{0}$
- `1` = $1 \times 1 \times 10^{0}$
- `-4` = $-1 \times 4 \times 10^{0}$
- `0.5` = $1 \times 5 \times 10^{-1}$
- `1,234,000` = $1 \times 1234 \times 10^{3}$
- `0.333333` = $1 \times 333333 \times 10^{-6}$
- ⅓ = ???
- π = ???

Rational numbers

$$\frac{numerator}{denominator}$$

- $0 = \frac{0}{1}$
- $1 = \frac{1}{1}$
- $0.5 = \frac{1}{2}$
- $0.1 = \frac{1}{10}$
- $⅓ = \frac{1}{3}$
- π = ???

Questions? Comments? Contributions?