

# scaler

---

## 1. Introduction

This crate provides a convenient `Formatter` to scale, round, and display numbers.

Scaling describes the usage of [decimal](#) or [binary unit prefixes](#) to increase readability; though no scaling and scientific notation are also supported. Rounding can be done either to a specified magnitude or to a number of significant digits.

## 2. Table of Contents

1. Introduction .....	1
2. Table of Contents .....	1
3. Usage .....	2
3.1. Example .....	2
3.2. Example .....	2
3.3. Example .....	2
3.4. Example .....	2

## 3. Usage

1. Execute `Formatter::new` to create a new `Formatter` with default settings.
2. Adjust separators, rounding, scaling, and sign behaviour as necessary using the setters.
3. Format numbers with `Formatter::format`.

### 3.1. Example

```
<> Rust
let f: scaler::Formatter = scaler::Formatter::new()
    .set_rounding(scaler::Rounding::SignificantDigits(2)); // general display
assert_eq!(f.format(&123), "120");
assert_eq!(f.format(&4.56), "4,6");
```

### 3.2. Example

```
<> Rust
let f: scaler::Formatter = scaler::Formatter::new(); // calculation results
assert_eq!(f.format(&456789), "456,8 k");
assert_eq!(f.format(&0.1), "100,0 m");
```

### 3.3. Example

```
<> Rust
let f: scaler::Formatter = scaler::Formatter::new()
    .set_scaling(scaler::Scaling::None)
    .set_rounding(scaler::Rounding::Magnitude(0)); // absolute values
assert_eq!(f.format(&0.1), "0");
assert_eq!(f.format(&1), "1");
assert_eq!(f.format(&1000), "1.000");
```

### 3.4. Example

```
<> Rust
let f: scaler::Formatter = scaler::Formatter::new()
    .set_scaling(scaler::Scaling::Binary(true)); // data sizes
assert_eq!(f.format(&0.1), "1,600 * 2^(-4)");
assert_eq!(f.format(&1023), "1.023");
assert_eq!(f.format(&1024), "1,000 Ki");
```