Numeric Primitives | Limits & Type Conversion

Many Numeric Types

- 8, 16, 32, 64, and 128 bit integers
  - Signed & unsigned
- ***isize*** & ***usize***
  - Pointer sized numeric types
    - ***usize*** used to index into arrays
  - Depends on architecture: 16bit, 64bit, etc
- 32bit & 64bit floating point

Min/Max: Unsigned Integer

Type	Min	Max
u8	0	255
u16	0	65535
u32	0	4294967295
u64	0	18446744073709551615
u128	0	<BIG>

Min/Max: Signed Integer

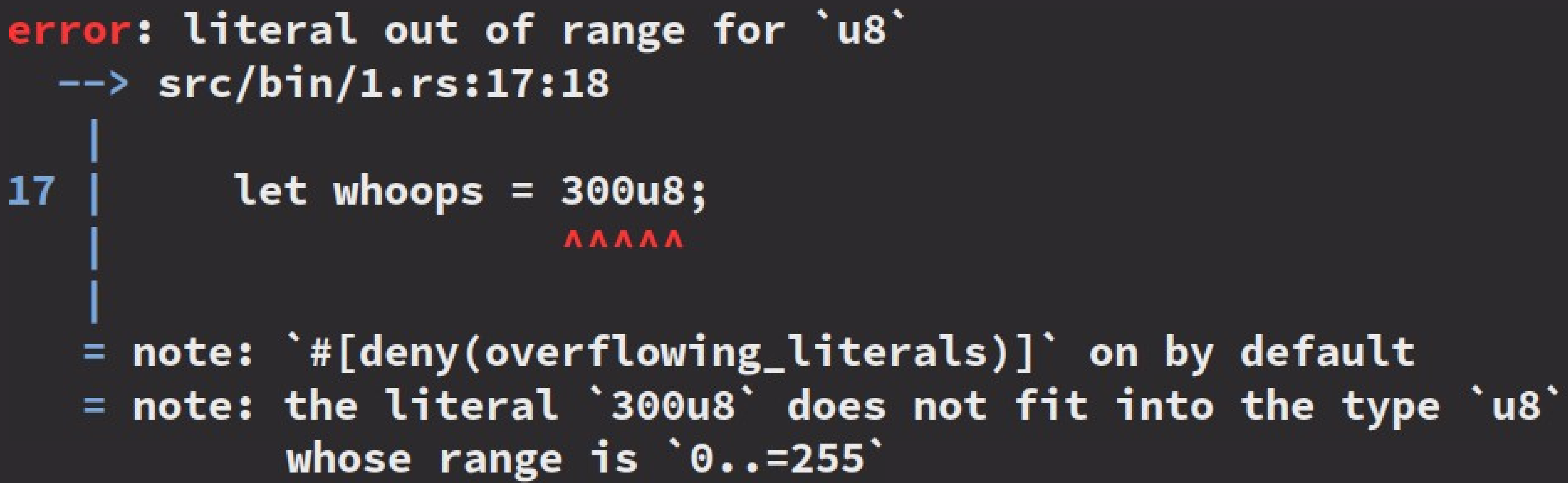
Type	Min Max
i8	-128 127
i16	-32768 32767
i32	-2147483648 2147483647
i64	-9223372036854775808 9223372036854775807
i128	-<BIG> <BIG>

Literal Numeric Annotations

```
15u8;
-12i16;
999_usize;
13_456_019u32;
17.7f32;
```

Type Safety

```rust
let whoops = 300u8;
```



```
error: literal out of range for `u8`
  --> src/bin/1.rs:17:18
   |
17 |     let whoops = 300u8;
   |                  ^^^^^
   |
   = note: `#[deny(overflowing_literals)]` on by default
   = note: the literal `300u8` does not fit into the type `u8`
           whose range is `0..=255`
```

Conversion

- Integers can be converted between types
  - ***u8*** will always fit into a ***u16***
    - Lossless conversion
  - ***u16*** cannot fit into ***u8***, but it can still be converted
    - Value will be a number in the range of the target type
- Math operations require all operands to be the same type
  - Convert to the largest type needed

Cast Syntax

```rust
let a = 15u8 as u16;
let b = a as u8 + 20u16 as u8;
```

Casting to less bits

- (Source value) – (Target max + 1)
  - Repeat until the value fits in the type
- Alternatively: (Source value) *modulus* (Target max + 1)
- This happens automatically when using ***as*** to convert

```
600u16 as u8   // =88
```

$$\begin{tikzcd} \text{Source} & \text{Target} \\ \mathbf{u16} \arrow[r] & \mathbf{u8} \\ 0..65535 & 0..255 \end{tikzcd}$$

600-256 = 344
344-256 = 88

Converting Floats To Integer

- Float to integer is a *saturating* conversion
  - The value will be clamped to the minimum or maximum of the target type
- Decimal points are truncated/dropped

```
800.5f32 as u8  // =255
 -300f32 as u8  // =0

800.5f32 as i8  // =127
 -300f32 as i8  // =-128
```

Source Target

$$\begin{tikzcd} \text{f32} \arrow[r] & \text{u8 } 0..255 \\ \text{f32} \arrow[r] & \text{i8 } -128..127 \end{tikzcd}$$

Checked Casting

```
u8::try_from(300u16)
```

Recap

- Numeric types can be cast using the ***as*** keyword
- Use ***TryFrom*** when you want to be sure the value will properly fit
- Annotations can be used with numeric literals to specify the type
  - Can use underscore (_) as a digit separator
- Compiler error to create a numeric literal outside of appropriate range