Type inference

```
18 fn main() {
19     let mut route = Vec::new();
20     route.push(Meters(2000.0).into());
21     route.push(Miles(2.0));
22     route.push(From::from(Meters(3000.0)));
23
24     println!("{:?}", route);
25 }
```

```
[Miles(1.2427424), Miles(2), Miles(1.8641136)]
```

Naming convention

Methods prefixed with:

- `as_` - takes self by reference and return a reference; should be cheap; e.g.: `as_str()`, `as_slice()`
- `to_` - takes self by reference and returns new value; may be expensive; e.g.: `to_owned()`, `to_bytes()`
- `into_` - takes self by value (move/consume) and return new value; it may or may not be cheap; e.g: `into()`, `into_iter()`, `into_vec()`
- `try_` - methods that may fail/return `Result`; e.g.: `try_unwrap()`, `try_into()`