Fundamentals ▌ Traits

Traits

- A way to specify that some functionality exists
- Used to standardize functionality across multiple different types
  - Standardization permits functions to operate on multiple different types
    - Code deduplication

Example

```rust
trait Noise {
    fn make_noise(&self);
}


fn hello(noisy: impl Noise) {
    noisy.make_noise();
}

fn main() {
    hello(Person {});
    hello(Dog {});
}
```

```rust
struct Person;
impl Noise for Person {
    fn make_noise(&self) {
        println!("hello");
    }
}

struct Dog;
impl Noise for Dog {
    fn make_noise(&self) {
        println!("woof");
    }
}
```

Example

```rust
trait Racer {
    fn go(&self);
    fn is_ready(&self) -> bool;
    fn checkpoint(&self, position: i32);
}
```

Recap

- Traits define similar functionality for different types
- Trait functions are just regular functions
  - Can accept arguments and return values
- Use *impl Trait* as a function argument to pass data via trait