Module 4: Interfaces for Abstraction
Topic 1.1: Polymorphism

Polymorphism

- Ability for an object to have different “forms” depending on the context
- Example: `Area()` function
  - Rectangle, **area = base * height**
  - Triangle, **area = 0.5 * base * height**
- **Identical** at a high level of abstraction
- **Different** at a low level of abstraction

Inheritance

- Subclass inherits the methods/data of the superclass
- Example: **Speaker** superclass
  - `Speak()` method, prints “<noise>”
- Subclasses **Cat** and **Dog**
  - Also have the `Speak()` method
- Cat and Dog are different forms of Speaker
- Remember: Go does not have inheritance

Overriding

- Subclass **redefines a method** inherited from the superclass
- Example: Speaker, Cat, Dog
  - Speaker `Speak()` prints "<noise>"
  - Cat `Speak()` prints "meow"
  - Dog `Speak()` prints "woof"
- `Speak()` is polymorphic
  - Different implementations for each class
  - Same **signature** (name, params, return)