Section Review

Learn to Code with Ruby

Inheritance

- **Inheritance** allows a class (the subclass) to obtain or “inherit" all the methods from another class (the superclass).
- Ruby supports a single inheritance model. A class can only inherit from one other class. However, that relationship can extend many levels
- Use the **< symbol** to make one class inherit from another.

Helpful Methods

- The **superclass** class method returns the superclass that a class inherits from.
- The **ancestors** class method returns an array of the complete inheritance hierarchy (all superclasses along with modules/mixins, which we'll talk about later).
- The **instance_of?** instance method checks if an object is instantiated from a given class.
- The **is_a?** instance method checks if a superclass exists anywhere in the object's inheritance hierarchy.
- The **methods** instance method returns an array of the object's methods as symbols.

Defining Methods in the Subclass

- A **subclass** can define **exclusive instance methods**.
- A **subclass** can override instance methods defined in the superclass. Define an instance method with the same name.
- We can override specific methods like **to_s** and **==** to hook into core object functionality (like string conversion or equality comparisons)

The super Keyword

- The **super** keyword invokes the method with the same name in the superclass.
- Without parentheses, **super** will pass all of the subclass method's arguments to the superclass method.
- With parentheses and no argument, **super** will pass no arguments to the superclass method.
- With parentheses and arguments, **super** will pass those specific arguments to the superclass method.

Duck Typing

- Be more concerned with an object's methods (what it can do) than its class (what it's made from).
- Ask what *role* the object can play in the program.