Section Review

Learn to Code with Ruby

The Hash Object

- A **hash** is a mutable object for storing unordered associations between objects.
- A **hash** holds 0 or more key-value pairs. A **key** is a unique identifier for a value. Values can have duplicates.
- Use the **hash rocket (=>)** to assign a value to a key. Separate each key-value pair with a comma.
- We can access a value by its key using square brackets. By default, Ruby returns **nil** if the key does not exist.
- We can add a key-value pair after hash initialization by using the square bracket syntax and an equal sign.

The Symbol Object

- The **symbol** is an immutable object used primarily for naming. Think of it like a lightweight string.
- It is common to use symbols as hash keys, so much so that Ruby has a simplified **key: value** syntax.

Iterating over a Hash

- We can iterate over a hash with the **each** method. Assume that the order of key-value pairs is *not* guaranteed.
- The block will receive both the key and the value from each key-value pair.
- The **select** and **reject** methods perform similar filtering operations (like on an array). Once again, we can use either the key or the value to select which key-value pairs are kept or discarded.

Various Hash Methods

- The **delete** method removes a pair by its key. It also returns the value.
- The **merge** method combines two hashes together. When there are duplicate keys, Ruby will prefer the value in the argument hash.

The Hash.new Method

- The **Hash.new** method is another way to create a hash. Its argument represents the value to return for a non-existent key.
- Ruby will reuse the same object each time we reference a non-existent key.
- If we provide a block, Ruby will re-run it for each non-existent key. In that scenario, we need to make sure to manually write the key to hash with its proper value.