Section Review

Learn to Code with Ruby

Exceptions

- An **exception** is an error in your program. Ruby "raises" an exception when it is unable to parse or execute your code.

- We've seen various errors throughout the course: **TypeError** when a type is invalid, **NameError** when a name does not exist, **NoMethodError** when a method does not exist on an object

- Ruby's default behavior is to terminate the program if an exception is not handled.

- To **rescue** an error means to intercept the error when it occurs and dictate the flow of the program.

The begin and rescue Keywords

- The **begin** keyword marks the start of a section of code where something could go wrong.

- A method body implicitly declares a **begin** section for the code.

- The **rescue** keyword without a value rescues *all* exceptions.

- The **rescue** keyword can also rescue *specific* exceptions. We can then react differently to different issues.

- Use the **=> e** syntax to get access to the error object (made from an exception class). We can assign the error whatever name we want.

- All error objects have helpful methods like **message** and **backtrace** to debug the issue.

The retry and ensure Keywords

- The **retry** keyword re-executes the code in the **begin** block (or the start of the method if **begin** is not present).

- The **ensure** keyword runs clean-up code after a **begin** block (if the operation was successful OR if an error occurred).

- When using **retry**, make sure to fix the problem in the **rescue** clause or we may trigger an infinite loop.

The raise Keyword

- The **raise** keyword raises/triggers a manual exception. We are telling Ruby "something is wrong here even if it's not apparent to you".

- We can provide **raise** with a custom error message. Ruby will use a **RuntimeError** object by default.

- Define custom error classes by inheriting from **StandardError**. Then, provide the error class after **raise** to trigger that kind of exception. We can still provide a custom error message.