Ruby's Exception vs StandardError

"Never rescue Exception in Ruby!"

The problem with rescuing Exception is that it actually rescues every exception that inherits from Exception. That's a problem because there are some exceptions that are used internally by Ruby. They are not the application exceptions, and swallowing them will cause bad things to happen.

- SignalException::Interrupt - If you rescue this, you can't exit your app by hitting control-c.
- NoMemoryError - Wanna know what happens when your program keeps running after it uses up all the RAM?

```
begin
  # some code which raises the exception
rescue Exception => e
  # Don't do this. This will swallow every single exception. Nothing gets past it.
end
```

I believe that one don't want to swallow any of these system-level exceptions. You only want to catch all of your application level errors.

All of the exceptions that you should care about inherit from StandardError

- RuntimeError - who could forget good old RuntimeError?
- NoMethodError - raised when you try to invoke a method that doesn't exist

```
begin
 # some code which raises the exception
rescue StandardError => e
  # Only your app's exceptions are swallowed.
end
```

When you don't specify an exception class at all, ruby assumes you mean StandardError.

Custom Exceptions Should Inherit from StandardError:

It means you should always inherit from StandardError, and never from Exception.

Inheriting from Exception is bad because it breaks the expected behavior of rescue.

```
class MyError < StandardError
end

raise MyError
```