Exceptions in Ruby

We'll need to use a new structure that is specific to handling exceptions, `begin...rescue`.

```
num = 0

begin
  puts "dividing 10 by #{num}"
  ans = 10 / num
  puts "the answer is #{ans}"
rescue
  puts "There was an error with that division."
end

puts "--------"
puts "finished"
```

The output of the above code is:

```
dividing 10 by 0
There was an error with that division.
--------
finished
```

The behavior of `begin...rescue` is this: The code in the `begin` block will execute until an exception is reached. Once an exception is reached, the execution will immediately jump to `rescue`. This behavior is evident by the fact that the code above doesn't print "the answer is ", because the exception is reached on the line before.

Here are a few more common error types that are native to ruby:

- `ArgumentError`
- `NameError`
- `NoMethodError`
- `IndexError`
- `TypeError`

This is by no means an exhaustive list, but these are the common ones.

Raising Exceptions

Say we wrote this method:

```
def format_name(first, last)
  first.capitalize + " " + last.capitalize
end

format_name("grace", "HOPPER") # => "Grace Hopper"
```

It's obvious how this method *should* be used. That is, we ought to pass in two strings when calling `format_name`. But this is still prone to misuse:

```
format_name(42, true) # => NoMethodError: undefined method `capitalize' for 4
```

Since we want to **raise** an exception when the arguments are not strings, we'll need a quick aside on how to check data type:

```
"hello".instance_of?(String)  # => true
42.instance_of?(String)       # => false
```

Simple enough! Let's use this to rewrite `format_name`:

```
def format_name(first, last)
  if !(first.instance_of?(String) && last.instance_of?(String))
    raise "arguments must be strings"
  end

  first.capitalize + " " + last.capitalize
end

format_name("grace", "hopper") # => "Grace Hopper"
format_name(42, true)          # => RuntimeError: arguments must be strings
```

In the code above we see `raise`, this is how we can make exceptions manually.

Bring it all together

Since our `format_name` method can raise an exception, we can also handle it with `begin...rescue`.

```
def format_name(first, last)
  if !(first.instance_of?(String) && last.instance_of?(String))
    raise "arguments must be strings"
  end

  first.capitalize + " " + last.capitalize
end

first_name = 42
last_name = true
begin
  puts format_name(first_name, last_name)
rescue
  # do stuff to rescue the "arguments must be strings" exception...
  puts "there was an exception :("
end
```

An important distinction to note is that `raise` is how we bring up an exception, whereas `begin...rescue` is how we react to that exception.