Useful Operations for Python Requests

There's a ton of things that we can do with Python Requests. We'll cover some of the most important features here and give you pointers for more information at the end.

First, how do we know if a request we made got a successful response? You can check out the value of ***Response.ok***, which will be **True** if the response was good, and **False** if it wasn't.

```
>>> response.ok
True

```

Now, keep in mind that this will only tell you if the web server says that the response successfully fulfilled the request. The response module can't determine if that data that you got back is the kind of data that you were expecting. You'll need to do your own checking for that!

If the boolean isn't specific enough for your needs, you can get the HTTP response code that was returned by looking at Response.status_code:

```
>>> response.status_code
200

```

Excellent! To write maintainable, stable code, you'll always want to check your responses to make sure they succeeded before trying to process them further. For example, you could do something like this:

```
response = requests.get(url)
if not response.ok:
    raise Exception("GET failed with status code {}".format(response.status_code))
```

But you don't really need to do that. Requests has us covered here, too! We can use the Response.raise_for_status() method, which will raise an **HTTPError** exception *only if* the response wasn't successful.

```
response = requests.get(url)
response.raise_for_status()
```

Up next, we'll look into the different types of HTTP request methods that we can make using this handy requests module.