How Not To Fetch Data Over HTTP

Let’s say you want to download a resource over HTTP, such as an Atom feed. Being a feed, you’re not just going to download it once; you’re going to download it over and over again. (Most feed readers will check for changes once an hour.) Let’s do it the quick-and-dirty way first, and then see how you can do better.

```
import urllib.request
a_url = 'http://diveintopython3.org/examples/feed.xml'
data = urllib.request.urlopen(a_url).read()          #①

print (type(data) )                                  #②
#<class 'bytes'>

print(data)
#b'<!DOCTYPE html><body style="padding:0; margin:0;">
# <html><body><iframe src="http://mcc.godaddy.com/park/pKMcpaMuM2WwoTq1LzRmYzWyqN=="
# style="visibility: visible;height: 100%; position:absolute" allowtransparency="true"
# marginheight="0" marginwidth="0" frameborder="0" width="100%"></iframe></body></html>'
```

① Downloading anything over HTTP is incredibly easy in Python; in fact, it’s a one-liner. The `urllib.request` module has a handy `urlopen()` function that takes the address of the page you want, and returns a file-like object that you can just `read()` from to get the full contents of the page. It just can’t get any easier.

② The `urlopen().read()` method always returns a bytes object, not a string. Remember, bytes are bytes; characters are an abstraction. http servers don’t deal in abstractions. If you request a resource, you get bytes. If you want it as a string, you’ll need to determine the character encoding and explicitly convert it to a string.

So what’s wrong with this? For a quick one-off during testing or development, there’s nothing wrong with it. I do it all the time. I wanted the contents of the feed, and I got the contents of the feed. The same technique works for any web page. But once you start thinking in terms of a web service that you want to access on a regular basis (e.g. requesting this feed once an hour), then you’re being inefficient, and you’re being rude.