urllib.request

The **urllib.request** module is primarily used for opening and fetching URLs. Let's take a look at some of the things you can do with the **urlopen** function:

```
import urllib.request
url = urllib.request.urlopen('https://www.google.com/')
print (url.geturl())
#'https://www.google.com/'

print (url.info())
#<http.client.HTTPMessage object at 0x7fddc2de04e0>

header = url.info()
print (header.as_string())
#('Date: Tue, 17 Jan 2017 11:23:58 GMT\n'
# 'Expires: -1\n'
# 'Cache-Control: private, max-age=0\n'
# 'Content-Type: text/html; charset=ISO-8859-1\n'
# 'P3P: CP="This is not a P3P policy! See '
# 'https://www.google.com/support/accounts/answer/151657?hl=en for more info."\n'
# 'Server: gws\n'
# 'X-XSS-Protection: 1; mode=block\n'
# 'X-Frame-Options: SAMEORIGIN\n'
# 'Set-Cookie: '
# 'NID=80=tYjmy0JY6flsSVj7DPSSZNOuqdvqKfKHDcHsPIGu3xFv41LvH_Jg6LrUsDgkPrtM2hmZ3j9V76pS4K_cBg7
# 'expires=Sat, 24-Dec-2016 18:21:19 GMT; path=/; domain=.google.com; HttpOnly\n'
# 'Alternate-Protocol: 443:quic\n'
# 'Alt-Svc: quic=":443"; ma=2592000; v="34,33,32,31,30,29,28,27,26,25"\n'
# 'Accept-Ranges: none\n'
# 'Vary: Accept-Encoding\n'
# 'Connection: close\n'
# '\n')

print (url.getcode())
#200
```

Here we import our module and ask it to open Google's URL. Now we have an **HTTPResponse** object that we can interact with. The first thing we do is call the **geturl** method which will return the URL of the resource that was retrieved. This is useful for finding out if we followed a redirect.

Next we call **info**, which will return meta-data about the page, such as headers. Because of this, we assign that result to our **headers** variable and then call its **as_string** method. This prints out the header we received from Google. You can also get the HTTP response code by calling **getcode**, which in this case was 200, which means it worked successfully.

If you'd like to see the HTML of the page, you can call the **read** method on the url variable we created. I am not reproducing that here as the output will be quite long.

Please note that the request object defaults to a GET request unless you specify the **data** parameter. Should you pass in the data parameter, then the request object will issue a POST request instead.

Downloading a File

A typical use case for the urllib package is for downloading a file. Let's find out a couple of ways we can accomplish this task:

```
import urllib.request
url = 'http://www.blog.pythonlibrary.org/wp-content/uploads/2012/06/wxDbViewer.zip'
response = urllib.request.urlopen(url)
data = response.read()
with open('test.zip', 'wb') as fobj:
    fobj.write(data)
```

Here we just open a URL that leads us to a zip file stored on my blog. Then we read the data and write it out to disk. An alternate way to accomplish this is to use **urlretrieve**:

```
import urllib.request
url = 'http://www.blog.pythonlibrary.org/wp-content/uploads/2012/06/wxDbViewer.zip'
tmp_file, header = urllib.request.urlretrieve(url)
with open('test.zip', 'wb') as fobj:
    with open(tmp_file, 'rb') as tmp:
        fobj.write(tmp.read())
```

The urlretrieve method will copy a network object to a local file. The file it copies to is randomly named and goes into the temp directory unless you use the second parameter to urlretrieve where you can actually specify where you want the file saved. This will save you a step and make your code much

simpler:

```
import urllib.request
url = 'http://www.blog.pythonlibrary.org/wp-content/uploads/2012/06/wxDbViewer.zip'
urllib.request.urlretrieve(url, '/home/mike/Desktop/blog.zip')
#('/home/mike/Desktop/blog.zip',
# <http.client.HTTPMessage object at 0x7fddc21c2470>)
```

As you can see, it returns the location of where it saved the file and the header information from the request.

Specifying Your User Agent

When you visit a website with your browser, the browser tells the website who it is. This is called the **user-agent** string. Python's urllib identifies itself as **Python-urllib/x.y** where the x and y are major and minor version numbers of Python. Some websites won't recognize this user-agent string and will behave in strange ways or not work at all. Fortunately, it's easy for you to set up your own custom user-agent string:

```
import urllib.request
user_agent = ' Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:47.0) Gecko/20100101 Firefox/47.0'
url = 'http://www.whatsmyua.com/'
headers = {'User-Agent': user_agent}
request = urllib.request.Request(url, headers=headers)
with urllib.request.urlopen(request) as response:
    with open('/home/mdriscoll/Desktop/user_agent.html', 'wb') as out:
        out.write(response.read())
```

Here we set up our user agent to Mozilla FireFox and we set out URL to **http://www.whatsmyua.com/** which will tell us what it thinks our user-agent string is. Then we create a**Request** instance using our url and headers and pass that to **urlopen**. Finally we save the result. If you open the result file, you will see that we successfully changed our user-agent string. Feel free to try out a few different strings with this code to see how it will change.