Victoria University of Wellington
School of Engineering and Computer Science
NWEN241 : Systems Programming
Project 1 - Implementing "pywget"

Due Monday 20/4/2015 23:59
LATE DAYS APPLY

1. Overview

This lab project concerns writing such program that we call "pywget" that is a simpler version of the GNU wget command line tool. GNU Wget is a free utility for non-interactive download of files from the Web that supports a range of protocols including HTTP, HTTPS and FTP.

There are several reasons we might want to write a program to fetch web pages and store them locally as files. Web crawlers to build indexes for search engines, tools for automated downloading updated contents such as software patches, mirroring of parts of a website for backup or to "warm" a web cache.

A key feature of Wget is that it can follow links in HTML, XHTML, and CSS pages, to create local versions of remote web sites, fully recreating the directory structure of the original site. This is sometimes referred to as "recursive downloading." While doing that, Wget respects the Robot Exclusion Standard (/robots.txt). Wget can be instructed to convert the links in downloaded files to point at the local files, for offline viewing.

You will implement "pywget" as a function and you will test it using a small website that has been developed for this purpose that deliberately does not contain any copyrighted content. The website is also deliberately very simple and makes use of a restricted subset of the HTML as well as contains only two file types (html and jpg).

You can view the website here:
http://homepages.ecs.vuw.ac.nz/~ian/nwen241/index.html

Examples of what the core and completion outputs should look like when run against the website are provided on the assignment webpage.

2. Process for completing the project

Please do discuss this project with other students and on the forum but ensure that the code your write is your own.

Note that you could complete this project by either using an existing Python wget implementation or use Python's capability to execute command line tools such as wget directly. However, this would defeat the purpose of the project (learning Python).

Neither are you allowed to use code unchanged and uncited from other Python projects, make sure you tell us in the documentation for your code where you do draw upon other projects (provide the URL and a short note describing what you borrowed).

The project is broken into three parts that are loosely core, completion and challenge. You should complete core and completion before attempting the challenge. Roughly completing the core has an upper grade bound of "C+", completing the core and completion is "B+" and going beyond to attempt the challenge is required for some kind of "A".

As usual correctness is 90% of your grade and the remaining 10% are related to code quality.

Correctness is both whether your program produces the output expected when run against the test website we have provided and also how general is your solution determined by running additional tests against similar but different websites.

Code quality is adherence to standards but also understandability and code efficiency (for example, use of regular expressions or minimizing re-processing of data).

3. Things you should know before starting

Use **urlibrequest.request** to do the heavy lifting for you but pay attending to the need to encode binary strings under some circumstances because it will return binary strings.

We didn't cover reading and writing binary files in class. Images are binary files, should you wish to write to one you will need to open the target file with **"wb"** instead of just **"w"**. For this project, you can probably get around having to do this yourself but it might be useful to know about.

You will have trouble accessing any external sites from an ECS machine because of our authenticated proxy, so I suggest that you stick to sites within the victoria.ac.nz domain.

The character "~" has a special meaning in Unix (the home directory of the user) and this confuses things when you have a directory starting with a "~". Just typing **cd ~ian** will try and

change your current directory to mine (forbidden). To get around this just enclose the directory name in quotes (for example, **cd "~ian"**).

What HTML markup do I need to know? The key ones are both absolute and local links <a href> and image links <img>.

We will be using URLs a lot here and it is worth revising how they are described. *Roughly speaking* each URL is composed of a protocol part, a directory and a file name (see https://www.cites.illinois.edu/101/url101.html). For example, http://homepages.ecs.vuw.ac.nz/~ian/nwen241/index.html is an URL made up of a reference to the http protocol “http:”, the name of the server “homepages.ecs.vuw.ac.nz”, the directory path “/~ian/nwen241/” and finally the filename “index.html”.

I have included examples of what the output should look like for the core, completion and challenge versions of the lab project on the assignment page.

The example files contain a mix of absolute and local references.

4. What to Do

4.1 Core Version

The aim of this part is to implement the core functionality of the “pywget” function allowing a specified file to be downloaded locally. For example:

pywget(url="http://homepages.ecs.vuw.ac.nz/~ian/nwen241/index.html")
or
pywget(url="http://homepages.ecs.vuw.ac.nz/~ian/nwen241/images/GrumpyCat.jpg")

Will cause the following to happen:

1. The specified file is be downloaded to the current directory.
2. Downloaded files have the same name as specified in their URL.
3. Collisions are handled by inserting **.x** before the file’s extension. For example, running the program three times to download GrumpyCat.jpg will result in three copies -- GrumpyCat.jpg, GrumpyCat.1.jpg and GrumpyCat.2.jpg.

Note that you should handle errors “gracefully” by returning None and printing “Network error”. You can use **try … except** in the same way as you dealt with file errors in assignment 2.

4.2 Completion Version

The aim of this part is to extend “pywget” to allow a specified html page (“root”) and its components (both html and images) to the current directory. All absolute links in the html page will be rewritten to local links pointing the downloaded components. For example:

pywget(url="http://homepages.ecs.vuw.ac.nz/~ian/nwen241/index.html")

Will cause the following to happen:

1. The specified html file is downloaded to the current directory.
2. Each link within the root html file (**<a href>** and **<img>**) is followed and the referenced component is also downloaded to the current directory.
3. Each absolute link **(referring to a downloaded component)** within the root html file is rewritten to a local link. Other absolute links are left unmodified.

Again your solution must cope with collisions to prevent overwriting files with the same name and when links are rewritten you must be sure to take account of any adjustments to filenames done to prevent collisions.

4.3 Challenge Version

The completion version of “pywget” stops at after it has downloaded the components referenced by the root html page. A recursive version would continue by following the links within every html page referenced. To deal with potential cycles, a common approach is to limit the number of times that a link can be followed. For example, the following tells “pywget” to only follow links to a depth of 2:

pywget(url="http://homepages.ecs.vuw.ac.nz/~ian/nwen241/index.html", depth = 2)

A related problem is that the completion version of “pywget” places all the downloaded content into a single flat directory. This will quickly become unmanageable once recursive downloading is added. A better approach is the follow the structure encoded in each URL. For example, wget creates the following local directory structure when downloading **http://homepages.ecs.vuw.ac.nz/~ian/nwen241/index.html**:

```
homepages.ecs.vuw.ac.nz/
   ~ian/
       nwen241/
            index.html
```

Your solution should solve both problems.

Hint. This stackoverflow discusses how to handle directory creation:
http://stackoverflow.com/questions/273192/in-python-check-if-a-directory-exists-and-create-it-if-necessary

5. Submission

When submitting use the following naming conventions for your attempts:

1. core.py (for the core version)
2. completion.py (for the completion version)
3. challenge.py (for the challenge version)

Please also include a README.txt that includes a discussion of how far you got and what you believe works (or doesn't work). This makes marking easier!

Unlike the assignments, partial credit is possible for the project.