Project 1

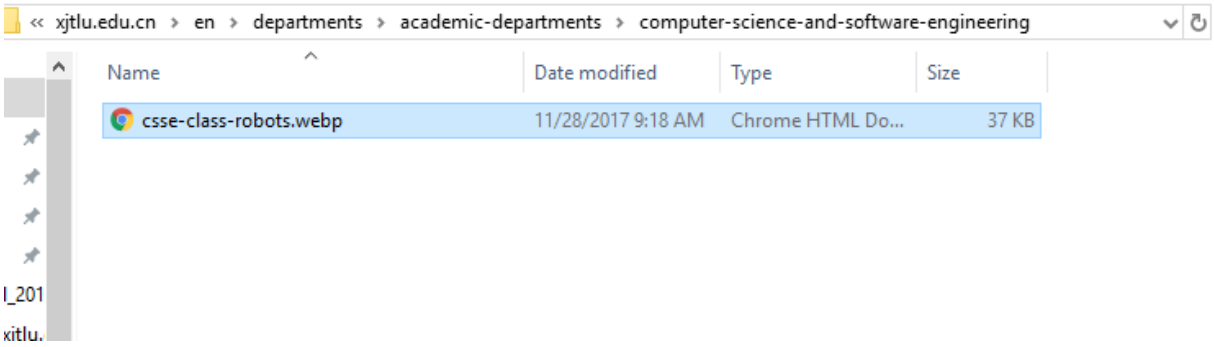
Imagecrawler

Due: November 27, 2018 11:59 PM

The goal of this project is to build an "Imagecrawler" application that can download images from websites and save them on your local computer. The program should take two command line parameters: a URL that is the starting point of the crawl, and a *depth,* which is how many pages deep your crawler should go. **The depth parameter is optional, and defaults to 5**, if it is not specified.

The way this should work is:

1. **(20 points)** Connect to the supplied URL and request the web page. Because we are studying networking, **you are NOT ALLOWED to use any of the built-in http libraries in Python (or any other library)**. You need to use TCP sockets, build the http requests yourself, and handle the replies.
    a. You should create a folder named after the website. For example, if the URL is http://www.cnn.com, your folder should be named www.cnn.com.
    b. URLs can include paths. For example: http://www.xjtlu.edu.cn/en/departments/academic-departments/computer-science-and-software-engineering/. In this case, you should make a series of directories with the same structure as the URL.



2. **(25 points)** Download all images in the page (.gif, .jpg, .jpeg, .png, .webp, case insensitive).
    a. The names of the images should be the same as they are on the remote server.
    b. Images may be on the same server, or on different servers. You should store it based on the page it appears on, not the server it exists on. (for example, http://www.cnn.com has many images that exist on different servers, but when you download them, store them all in the www.cnn.com folder.) This includes images that exist on the same server. Regardless of the folder the image exists in, it should be stored in the folder for the current URL.

3. **(25 points)** For all href links in the page, repeat steps 1 and 2, up to the depth specified by the user.
    a. Remember that links can be absolute or relative to the current server. (e.g. http://www.cnn.com is absolute, but en/departments/academic-departments/computer-science-and-software-engineering/about/learning-and-teaching is relative to the current server.
    b. The depth of a page is the number of links you followed to get to it. The original URL is depth 0, any links on that page have depth 1, any links on any depth 1 pages have depth 2, etc.
    c. Remember that links may be circular. I could link to you, and you could link back to me.
    d. Ignore style sheets, Javascript, etc.
    e. To find links in the HTML text, I suggest you look at the Python regular expression tools (re.regex) or HTMLParser. These are both standard libraries in Python.
4. **(10 points)** To speed up your application, thread your application to download in parallel
5. **(10 points)** Support https (SSL) connections. (You can use the Python libraries for SSL for this.)
6. **(10 points)** A one-page report specifying which parts of the project you implemented, known bugs, and citing any sources you take more than 10 lines of code from.

For this project you CAN NOT use 3rd party libraries. You can only use standard Python libraries that are part of the standard distribution. Also, you don't need to deal with Javascript, CSS, etc. Only plain HTML pages. To help you, I setup a test website at http://csse.xjtlu.edu.cn/classes/CSE205

All students are expected to be able to explain their code. If I ask you what the code does, and you can't explain it clearly, there will be a penalty, up to 100% of your grade for this assignment. (Meaning you can receive a zero for not being able to explain your code.)

This is an individual project so collaboration or sharing of code is not allowed. The University policy on plagiarism and collaboration will be strictly enforced, and I will run your code through plagiarism detection software. Limited internet references may be used (you can't submit a project you download from the internet), but any source you use more than 10 lines of code from should be documented in your report. Some examples of acceptable code snippets include: code for how to use TCP sockets in Python, code for how to make containers thread-safe, code for how to create threads, etc.