Assignment 2
Due Sunday June 28 at 11:55 a.m.

This assignment is intended to provide an introduction to simple Python3 programming and to encourage you to explore the Python documentation at `https://docs.python.org/3/`.

The CheckWebsite.py Program

Websites need to be regularly checked and updated. One frequently occurring problem is a link on a webpage that no longer works. The usual cause is that the link refers to another webpage which no longer exists or whose URL has changed. A tool which checks a website for links which are broken in this manner is very helpful to the website maintainer. Your task is to produce a simple version of such a tool. Note that a website does not usually consist of a single webpage. The website will likely have many pages, each referring to other pages in the collection. The tool should explore the pages in a systematic breadth-first manner.

The output for this tool when visiting the fictitious website `http://no.such.site.no` could look like this, where the Linux command prompt is shown as '%':

```
% python3 CheckWebsite.py -maxvisits 100 http://no.such.site.no
Checking website http://no.such.site.no
Maximum number of webpage visits set to 100
* bad reference to https://nowhere.za on page http://no.such.site.no
* bad reference to http://foo.bar.com on page http://funny.site.no
Checking finished; 23 webpages checked; 2 bad references found
```

Some Requirements and Some Help:

- You are provided with a small Python program to get you started. This program is named `VisitURL.py`. It is invoked with a command line like this example:

  ```
  % python3 VisitURL.py -maxvisits 10 http://www.uvic.ca
  ```

  The supplied program parses the command line arguments and then invokes a function named `readwebpage`, passing it the URL provided on the command line. The function reads the HTML code for the webpage and returns it as a string.

  The optional `maxvisits` parameter is parsed and its value is stored in a variable named `maxhits`. If the `readwebpage` function is invoked more than `maxhits` times, the function returns an empty string without reading the webpage. (The reason for this behaviour is explained below.)

  If the URL cannot be read because of a broken link, the function returns `None` as its result.

- Your program will need to search through the HTML code for links to other webpages. Your program will search for the pattern of characters `href="xxxxxx"` where the doublequote characters are part of the pattern. The part shown as `xxxxxx` is the URL of another webpage.

  Many Python programmers automatically (i.e. without thinking) will use the facilities provided by the regular expression module (named `re`) to perform searches through strings. However, you most definitely do not to use such a powerful mechanism. You can use the Python string function named `find` to search for the substring `'href="'` and then use `find` again to search for the closing doublequote character. (Look in section 4.7.1 of the Python3 documentation for the Python

Standard Library to see how to use the `find` function and how to specify where the search can start.)

- When you have extracted a URL from a webpage, you should check whether the URL prefix is the substring `"http:"` or `"https:"`. If it begins with anything else, ignore it. (It will be a URL that our simple version of the checking tool is unable to handle.)

  A complication is that the check for the URL prefix must be a case-insensitive comparison.For example, the URL prefix could be "Http:" or "HTTP:".

  Again, look through section 4.7.1 for string methods that help with making this prefix check.

- A bug in your program could easily cause your program to explore hundreds of thousands of websites (causing a lot of internet traffic) or your program might access the same website repeatedly thousands of times a second (which would be perceived as a denial-of-service attack by the website's manager). For these reasons, you WILL leave the `maxvisits` command-line parameter set to its default value of 10 while you are developing and testing. (I.e., just omit the `-maxvisits` option from the command line.)

- The supplied code uses some language features (the *with* statement, the *try-except* statement, the *global* declaration) which have not yet been covered in the course. Of these, the *global* declaration is the only one you may need to use in your own code. The rough-and-ready rule is that if the code inside a function needs to refer to a global variable, there should be a *global* declaration for that variable inside the function. (Otherwise, Python will either create a local variable with the same name or give you an error message depending on whether your first access is reading from the variable or storing into it.)

  Feel free to read about the *global* statement in section 7.12 of the Python language reference manual or on some upcoming course slides.

An Overall Algorithm

Here is the algorithm which you are required to follow. It performs a breadth-first search of the internet starting at the initial website. (It is not quite what we want for a website checker because it will check neighbouring sites outside the desired website too; however implementing the checker properly would require more sophisticated pattern matching.)

The algorithm is based on a general approach known as a *worklist*. Its underlying data structure is a queue (easily implemented with a list in Python). Here is pseudocode for the required approach.

```
url_list := [ the_initial_url ]
while url_list is not empty do
     url := one item removed from the front of url_list
     s := readwebpage(url)
     if s is None then
          report this url as a bad url
     else
          for each “href=....” found in s do
               test the url found after the prefix
               if that url begins with ‘http’ or ‘https’ then
                    if we have not added this url to url_list before then
                         add the new url to the end of url_list
```

There is one big detail missing from this pseudocode. When you discover a bad URL, you also need to know the URL of the webpage which contained the bad URL, so that both URLs can be printed in

the program's output. How to handle that is left as an exercise to you. You might want to implement that queue as a list of URL pairs, or you might want to implement a dictionary where the key is a URL and the associated value is a list of URLs which are referenced in the webpage identified by the key. It's your choice.

Deficiencies in the Simple Website Checker

1. Our simple website checker does not restrict itself to checking webpages within the website, it can stray outside.
2. Websites whose page content is generated dynamically (usually by executing Javascript code) will not be checked at all. The Computer Science website at `http://www.csc.uvic.ca` is an example of a website where almost everything is generated dynamically. The perfect checking tool would execute the Javascript code, but that is way beyond the scope of this assignment.
3. Not all '`href=`' references refer to webpages containing HTML code. Some, for example, refer to embedded images. It would be desirable for our checking tool to report broken links to embedded images and similar, but that is also a little beyond the scope of this assignment.
4. Some `href` values are not complete URLs. They are paths relative to a previously given URL.
5. The supplied code assumes that all HTML webpages are encoded using the UTF-8 character set (i.e. ASCII plus 8 bit codes). This is not necessarily true.

Submission Instructions

- You are to submit your solution as one text file named `CheckWebsite.py` (with that exact capitalization) on the course conneX site.
- The first line of the text file must be this comment line (known as a 'shebang'):

```
#! /opt/bin/python3
```

- The second line of the text file must be a comment identifying you, the program's author. For example,

```
# Author: Guido van Rossum, V00123456
```

- Your program must execute as expected when run on a B215 Linux computer. A command similar to the following will be used to invoke your program:

```
% python3 CheckWebsite.py -maxvisits 20 http://www.uvic.ca
```

- The output of your program must have the exact same format as the example shown on the first page of this assignment.
- Your program must be robust. No matter what command-line arguments are provided, no matter what junk a webpage contains, it must be impossible to cause your program to crash.