Coursework 2 Assessment Brief

1. Module number	*SET08115*
2. Module title	*Applications Development*
3. Module leader	*Petra Leimich*
4. Tutor with responsibility for this Assessment Student's first point of contact	*As above*
5. Assessment	*Coursework 2 - Python Script Development - Coursework with Lab Demo*
6. Weighting	*50% of module assessment*
7. Size and/or time limits for assessment	*You should be able to complete this assessment within approximately 30 hours (if you have kept up with the Lectures and Labs). Lab Demo will be approx. 15 mins explaining the various scripts/demoing selected functionality.*
8. Deadline of submission	Monday 28th Nov 2016, 9am, via Moodle (see Moodle for submission Link). Your attention is drawn to the penalties for late submissions. Demo sessions to be arranged in week of submission – this will use normal lab and lecture slots where possible, but additional times may be required to accommodate all students.
9. Arrangements for submission	*Submit your code files to Moodle by the submission deadline. You must keep your own copies of your assessment deliverables and bring them to the lab demo (working code and documentation).* *Make sure that your code works on the machine that you are going to demo on!* *By submitting to moodle you confirm that the assignment is your own and that any code used from templates or online sources has been identified as such through inline comments.* *It is YOUR responsibility to ensure that the online submission has completed successfully.*
10. Assessment Regulations	All assessments are subject to the University Regulations.
11. The requirements for the assessment	See specification overleaf. Demo during specified session. Failure to attend demo will result in a fail. Resit in T3 (Summer 2017)
12. Special instructions	You must demonstrate selected scripts, and explain the code in detail. Code documentation will also be reviewed. This will be done during week 13, using scheduled classes as much as possible. Without a demo the work cannot be marked. Sign up slots for the demos will be available in the SOC office or on Petra's office door.
13. Return of work and feedback	You must keep a copy of your work. Individual oral feedback will be given during the lab demo (you are advised to take notes). A written summary of the feedback with indicative marks will be available via Moodle within three weeks of the submission date. Please note that all marks are subject to verification at T1 module board. If you have difficulty understanding your feedback or would like to discuss it, you should see Petra Leimich or email for an individual appointment p.leimich @napier.ac.uk.
14. Assessment criteria	See specification overleaf.

Assessment overview

Build a Python web page scraping/information gathering reconnaissance utility. This will retrieve web content and parse this for specified information, and then perform some forensic analysis on the contents.

The application should fetch webpage code via a specified URL and analyse the webpage:

- parse out links to information such as subpages and files,
- parse out any interesting artefacts such as email addresses and possible hashed passwords.

Use separate modularised scripts for different functions, similar to those built when completing the module labs.

Requirements

- The specific functionality requirements for each element are detailed in the marking criteria overleaf. Go through these carefully to understand the required functionality fully!!
- This coursework makes use of a number of components that you should already have completed as lab exercises. Refer to the corresponding exercise descriptions on the lab sheets for a detailed description of their functionality.
- All scripts must have appropriate header documentation, including author attribution.
- All functions must have suitable doc strings to provide help. All appropriate code sections must be commented.
- Always make good use of variables to “future-proof” your code.
- Code should be easy to read and be structured in a logical and efficient way that facilitates reuse (e.g. using separate modules and functions).
- During the demo, you will be asked to explain comments, and discuss and demonstrate various pieces of code in the assessment, including what each regular expression does and how each was created.

Test website and additional advice

There is a test webpage at the following link which can be used to test out your Python coursework scripts:
http://www.soc.napier.ac.uk/~cs342/CSN08115/cw_webpage/index.html
A copy is at:
http://socrdlvideo.napier.ac.uk/~csn11118//CSN08115/index.html

The bad file hash set we want you to check against is:

'9d377b10ce778c4938b3c7e2c63a229a':'contraband_file1.jpg'
'6bbaa34b19edd6c6fa06cccf29b33125':'contraband_file2.jpg'
'e4e7c3451a35944ca8697f9f2ac037f1':'contraband_file3.jpg'
'1d6d9c72e3476d336e657b50a77aee05':'contraband_file4.gif'

If you discover bad files, attempt to view their real contents... you may need to use the forensic file type signature script for this.
Try to keep it simple and build your code (and regex's) bit by bit... and don't worry if you can't get it all working at first.

Marking

	Functionality/Scripts Required	Functionality Implemented	Code Understanding	Error Handling/ Documentation
1	Single web page download and parsing for links and outputting a summary of the total number of links, and a formatted list of the matched links [6]			
2	Build in parsing and printing of unique email addresses/phone numbers/MD5 hashed passwords [6]			
3	Crack hash passwords found [6]			
4	Exception Handling where appropriate [6]			
5	Build in parsing and printing of any links to image files (gif,jpg,bmp),and docx word files. [6]			
6	Downloading of the files into a specified directory **c:\temp\coursework**, reporting on any which don't download. [6]			
7	Perform forensic analysis on any files downloaded, matching against a bad files list. Create a dictionary of file hashes (a list will be provided) and compare the hashes of files downloaded, and report on any bad files. [6]			
8	Reuse of code in separate modules and functions, being called from one **getwebinfo.py** script. [6]			