CS2505 – Python Lab 03
23.03.2021

Quick recap from last week's lab instructions:

1. This year, you are required to submit 2 assignments worth 10 marks each, which form the Continuous Assessment grade. Each of the four labs contribute to the assignments (Lab 1 and Lab 2 to the first assignment and Lab 3 and Lab 4 to the second assignment). In grading these assignments, we will take into account the **correctness** of your solution, the **approach** taken, and **comments**, which should be clear and concise. We will be checking carefully for plagiarism and penalties will be strictly applied.
2. If you don't understand a question, please ask us (lecturer and assistants), we are happy to help.
3. All Labs will consist of some Python programming and some questions. To maximise your Continuous Assessments marks, please answers all sections.
4. We do not accept solutions which are written in Python 2 (https://pythonclock.org/). **Make sure your solutions work in Python 3**.

Your solutions for this Lab, including the solutions for the additional exercise should you decide to attempt same, must be included in the 2nd assignment report (due at 5pm Cork local time on Friday, 16th April 2021), which you must be submitted on Canvas within the specified deadline. Please note that no late submission will be accepted by Canvas. If your solution files cannot run successfully, you will lose marks. So, make sure that there is no **syntax, compilation or run-time error**. You do not need to include your name or UCC ID in the name of the submitted files (Canvas recognises you by your account automatically). **Follow the file naming conventions** as mentioned in the description of the lab exercises. ☺
We recommend (but not obligate) that you follow the official style guide for Python:
https://www.python.org/dev/peps/pep-0008/
The official Python 3.7 documentation is located here: https://docs.python.org/3.7/index.html

---

Lab 3:

For this lab, we will NOT need to use the two UCC servers we used in the last lab, but you can run the exercises on your own computer again. In this lab, you will interact with a python web server that handles one HTTP request at a time. The web server will accept and parse the HTTP request, get the requested file from the server's file system, create a HTTP response message consisting of the requested file preceded by header lines, and then send the response directly to the client. You will notice that if the requested file is not present in the server, the server should send an HTTP "404 Not Found" message back to the client.

Create a project folder for this lab on your computer and download the following files from the Laboratory Exercise section of the module's Canvas site (they are listed under the lab 3 instructions): 'WebServer.py' and 'HelloWorld.html'.

Running the Server:

Run the server program using "`python3 WebServer.py`" or similar depending on what OS you are using. Determine the IP address of the computer that is running the server. Open a browser and provide the corresponding URL. For example: http://192.168.1.5:6789/HelloWorld.html

'HelloWorld.html' is the name of the file you placed in the server directory. Note also the use of the port number after the colon. The browser should now display the contents of HelloWorld.html. If you omit the port number, i.e. ":6789", the browser will assume port 80 and you will get the web page from the server only if your server is listening at port 80. Now try to get a file that is not present at the server, e.g. http://192.168.1.5:6789/GoodbyeWorld.html. You should get a "404 Not Found" message.

In this week's lab:

1. Instead of using a browser, write your own HTTP client to test your server. Your client will connect to the server using a TCP connection, send a HTTP request to the server, and display the server's response as an output. You can assume that the HTTP request sent is a GET method. (Hint: Check the message sent to the server by the browser to get an idea of the message syntax that your client should use. You may remember the way we used a telnet client to interact with a web server during the lectures – have a look at the lecture recording if you forgot.)

   The client should take command line arguments specifying the server IP address or host name, the port at which the server is listening, and the path at which the requested object is stored at the server. The following is an input command format to run the client.

   ```
   client.py server_host server_port filename
   ```

Questions:
As there are a lot of concepts to understanding in the coding section of this lab, all that is required when you put your answers to the 2nd assignment together is to explain in the question section: the problems you found and the steps you took to design this Lab.

When you submit your solutions as part of the 2nd assignment, submit the solution files as "`client_solution_lab3.py`" and "`WebServer_solution_lab3.py`". Submit the answers to questions as "`answers_lab3.txt`" file. Details on the second assignment to follow.

Additional Coding Assignment:

Currently, the web server handles only one HTTP request at a time. The additional coding exercise this week is to implement a multithreaded server that is capable of serving multiple requests simultaneously. Using threading, first create a main thread in which your modified server listens for clients at a fixed port. When it receives a TCP connection request from a client, it will set up the TCP connection through another port and services the client request in a separate thread. There will be a separate TCP connection in a separate thread for each request/response pair. Submit the solution file as “`WebServer_solution_additional_lab3.py`”.