CS2505 – Python Lab 04
30.03.2021

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**.

This week we offer students an additional means of maximising their Continuous Assessments marks, with an additional *Coding Exercise*, from which you can receive a maximum of 1 additional mark. This extra coding exercise is optional and will not affect your ability to receive the maximum 5 marks for the initial part of this lab. However, if a student receives only 4 marks from the main lab work, they can use the additional exercise to obtain one mark to achieve the maximum of 5 marks for the lab.
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 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
Where to get Python 3.7+?

Lab 4:

In this week's lab, we are going to look at UDP Sockets.

UDP Sockets:

There is no code to download this week. You will create the client and server python files. Look to previous Labs for details on running your client and server.

In all the labs so far, we have been using TCP to control our connections between the Client and Server and vice versa. In this lab you will use UDP.
Write a client/server program in python using datagram packets (UDP). The server listens for a message from a client. When it receives a message it prints the message, changes it to uppercase, logs it with date and time, and sends it back to the client. The client prints the received message. The following links will help in determining the correct methods to use:
https://docs.python.org/3.5/library/socket.html.

Steps for client:

- Write two different files called `UDPClient.py` and `UDPServer.py`.
- The client takes in the Domain Name of the server it is sending a message to.
- The client does DNS Lookup and uses the returned IP Address as the server's address.
- The client creates a Datagram Socket.
- Sends a message to the server. *Remember*: UDP sends a message but does not create a connection.
- The client receives the response.
- The client prints the answer on the screen.

Steps for server:

- The server creates a Datagram Socket.
- The server binds this socket to the host and port which is listening on port 6789
- The server stays in an infinite loop and listens for messages from clients.
- The server receives data and the network address and port number for each client request.
- The server prints the received message and saves the message to a log file with a timestamp.
- The server takes the sentence and changes it to uppercase.
- The server extracts the address and the port of the client.
- The server sends the timestamp and the uppercase sentence to the client.

To Do:

1. Write the two python files, calling them **`UDPClient.py`** and **`UDPServer.py`**. Include exception control in your code and remember to close all sockets as required.
2. Explain the differences between TCP and UDP both in Python Sockets and in general networking. Write and submit your answers in **`answers_lab4.txt`** file.
3. Submit the UDPClient.py and UDPServer.py files as `UDPClient_lab4.py` and `UDPServer_lab4.py` files. Don't forget the `answer_lab4.txt` file.

Additional Coding Assignment:

The additional assignment this week is to add a reliable data transfer mechanism to UDP, which is an inherently unreliable transport protocol. The specific mechanism should be based on the stop-and-wait mechanism discussed in the lecture. Add the finite state machine functionality to your UDPClient and UDPServer as discussed in the lecture in order to add reliability to UDP transfer. Submit the solution files as `UDPclient_additional_lab4.py` and `UDPserver_additional_lab4.py`.