CS2505 – Python Lab 04
14.03.2019

Let us start with the **Important** stuff first:

1. The grading for these Labs 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 Yusuf or Ahmed, the lab demonstrators, who will be only 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. The Continuous Assessment labs are worth 20% of your final year mark for this module. Thus, each of the five labs is graded out of maximum of 4 marks. To offer students an additional means of maximising their Continuous Assessments marks, each week we will also offer an additional *Coding Assignment*, from which you can receive a maximum of 1 additional mark. This extra assignment is optional and will not affect your ability to receive the maximum 4 marks for the initial part of this lab. Note: irrespective of how many questions you attempt over the five weeks, the maximum number of marks that can be received from the Continuous Assessment aspect of this module is 20 marks.
5. 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 assignment should you decide to attempt same, must be submitted on Moodle within the specified deadline. Please note that no late submission will be accepted by Moodle. 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 (Moodle does that automatically). **Follow the file naming conventions** as mentioned in the description of assignment. ☺
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+?
In order to run Python 3.7 on machine in class room, type python3 (or python3.7 to be 100% sure) in Kubuntu or python in Windows. You can run your script by typing "python3.7 script.py" in Kubuntu or "python script.py" in Windows.
If you use your own machine:
- for Windows OS, go to python.org and download Python 3.7 or higher;
- Ubuntu usually has the latest Python installed by default
- for older versions of Ubuntu (for 14.04, for example) you get the latest Python version from https://launchpad.net/~fkrull/+archive/ubuntu/deadsnakes ppa:

```
sudo add-apt-repository ppa:fkrull/deadsnakes
sudo apt-get update
sudo apt-get install python3.7
```

- another very good solution is usage of conda package system(https://conda.io/docs/). Go to https://conda.io/docs/install/quick.html and read manual for your OS. With conda you can quickly install a package by typing “conda install numpy” and quickly update all installed packages by typing “conda upgrade –all” (upgrade is alias for update here).

Lab 4:

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

UDP Sockets:

Create a new project folder called “CS2505_lab4”, 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.txt** file.
3. Submit the UDPClient.py and UDPServer.py files.

Additional Coding Assignment:

The additional assignment this week is to rewrite your UDPClient and UDPServer in the Python 2.7 (python2) on your machine. Also include exception control in both UDPClient and UDPServer. Submit the solution files as UDPclient_additional.py and UDPserver_additional.py.
Easiest way to run a specific version of python is to mention it in the command e.g. “python2.7 ./UDPServer_additional.py”.