Trabajo Práctico
Introducción a la Programación de Protocolos de la Pila TCP/IP con Python

Bibliografía (ampliada)

El tutorial de Python . http://python.org.ar/pyar/Tutorial?action=AttachFile&do=view&target=TutorialPython.pdf
Python Standard Library. Fredrik Lundh. O'Reilly & Associates. http://effbot.org/zone/librarybook-ora.htm
Documentación Módulos de Internet: http://docs.python.org/library/internet.html

Formato de entrega de los scripts

Todos los scripts tienen que ser guardados con el formato: **apellido**_tp-**NRO_DE_TP**-**NRO_DE_EJERCICIO**.py

Solo deben admitir parámetros pasados por línea de comandos. Bajo ningún concepto deben ser interactivos.

Objetivo: Conocer los fundamentos de la programación en lenguaje Python y experimentar con las facilidades para el manejo de protocolos de la pila TCP/IP en alto nivel (utilizando módulos del lenguaje).

1 Codifique un script que tome una cadena como parámetro, ejecute una búsqueda en Yahoo, ASK y en Google. Investigue qué opciones debe utilizar en cada caso. Comentar brevemente qué módulos utilizó y con qué dificultades se ha encontrado. Utilice además para la llamada el comando time para controlar el tiempo de ejecución de su proceso.

2 Escriba un programa que recupere y guarde en un directorio una página HTML (ambos recibidos como parámetro) y todos los objetos que contiene (imágenes, applets, etc.). Implemente – además – un caché de forma tal que si el objeto a recuperar existe y está actualizado no sea descargado nuevamente (use los headers HTTP correspondientes).

3 Siguiendo con el espíritu del ejercicio 3, escriba un programa que permita realizar un *crawling* utilizando urllib. El mismo debe recibir dos parámetros por linea de comandos, el primero será la url semilla desde la cual comenzará el *crawling* y el segundo la profundidad. Ejemplo:

```
$ python crawler.py www.unlu.edu.ar 2
```

Explique brevemente cuál fue su estrategia de profundidad y corte del algoritmo. Además el programa debe retornar un informe HTML con los resultados del *crawling* donde se muestre para cada url la cantidad de objetos (imágenes, div, etc.) contenidos. Finalizado el informe publicarlo en un servicio Web generado con el módulo SimpleHTTPServer.

4 Investigue el módulo BaseHTTPServer y escriba un programa que permita recuperar una página html básica escrita por ud. El servidor http debe generar un log con información relevante de cada petición y volcarla a un archivo a tal efecto.

5 Escriba un programa que permita sincronizar dos directorios remotos vía protocolo HTTP. Explique someramente cómo trabaja su aplicación, sus capacidades y sus limitaciones.

6 Utilizando el mismo módulo del ejercicio 4 y *urllib*, escriba un programa que funcione como servidor proxy reverso hacia diferentes dominios de sindicación de noticias (RSS) de diferentes diarios. Desde nuestro navegador deberíamos poder consultar por GET cualquiera de estos canales a través de nuestro *proxy*. Un ejemplo de llamada: http://nuestro.proxy.com:8000/?p=http://pagina12.com/rss. ¿Qué utilidades puede tener este proxy para la Web 2.0?

7 Pensemos que estamos gestionando un servidor Web que atiende a un gran número de usuarios, en estos casos es posible mejorar el rendimiento de nuestro servicio utilizando balanceo de carga. Investigue esta técnica y explique someramente en qué consiste y con qué algoritmos puede ser atacada. Luego, escriba un programa que trabaje como *Load Balancer* de al menos 3 nodos http internos. Nuestro balanceador debe soportar al menos un algoritmo de *Round Robin*.