Diseño de Aplicaciones para Internet (2014-2015)
Guión de Prácticas 1:
Python

S. Alonso, J.M. Guirao
zerjioi@ugr.es, jmguirao@ugr.es

Resumen

Durante las prácticas de la asignatura *Diseño de Aplicaciones para Internet* (DAI) vamos a hacer uso intensivo de diversos lenguajes de programación. Entre otros, utilizaremos `Python`, un lenguaje de alto nivel sencillo, potente, libre, fácil de aprender, interpretado y multiplataforma (entre otras características). En esta primera práctica trataremos de resolver algunos problemas genéricos de programación usando dicho lenguaje para familiarizarnos con sus tipos de datos, estructuras de control, etc.

Para aquellos que no tengan conocimientos sobre este lenguaje, existen numerosos manuales en Internet que permiten acercarse a la programación Python de manera sencilla y amena como por ejemplo: *"A Byte of Python"*: `http://swaroopch.com/notes/python/` o el manual de Google sobre Python: `https://developers.google.com/edu/python/`

1. Problemas "Sencillos"

1. Programe un mini-juego de "adivinar" un número (entre 1 y 100) que el ordenador establezca al azar. El usuario puede ir introduciendo números y el ordenador le responderá con mensajes del estilo *"El número buscado el mayor / menor"*. El programa debe finalizar cuando el usuario adivine el número (con su correspondiente mensaje de felicitación) o bien cuando el usuario haya realizado 10 intentos incorrectos de adivinación.

2. Programe un par de funciones de ordenación de matrices (UNIDIMENSIONALES) de números distintas (burbuja, selección, inserción, mezcla, montículos...) (`http://es.wikipedia.org/wiki/Algoritmo_de_ordenamiento`). Realice un programa que genere aleatoriamente matrices de números aleatorios y use dicho métodos para comparar el tiempo que tardan en ejecutarse.

3. La *Criba de Eratóstenes* (`http://es.wikipedia.org/wiki/Criba_de_Erat\%C3\%B3stenes`) es un sencillo algoritmo que permite encontrar todos los números primos menores de un número natural dado. Prográmelo.

4. Cree un programa que lea de un fichero de texto un número entero `n` y escriba en otro fichero de texto el `n`-ésimo número de la sucesión de Fibonacci (`http://es.wikipedia.org/wiki/Sucesi\%C3\%B3n_de_Fibonacci`).

5. Cree un programa que:

- Genere aleatoriamente una cadena de [y].
- Compruebe mediante una función si dicha secuencia está *balanceada*, es decir, que se componga de parejas de corchetes de apertura y cierre correctamente anidados. Por ejemplo:
  - [] → Correcto
  - [[][]] → Correcto
  - [][] → Correcto
  -][→ Incorrecto
  - [][[→ Incorrecto
  - []][[] → Incorrecto

2. Problemas "Complejos"

- Implemente el *Juego de la Vida* (`http://es.wikipedia.org/wiki/Juego_de_la_vida`). La salida de cada iteración debe guardarse en ficheros de texto con nombre consecutivo. Opcionalmente, se puede usar la biblioteca `graphics.py` para mostrar la evolución del juego como imágenes en vez de ficheros de texto. **Nota:** Si ejecutas los ejercicios en una máquina virtual sin escritorio deberás hacer un X11 fordwarding (parámetro `-X` en el comando `ssh`) para poder visualizar ventanas con `graphics.py`. En Ubuntu Server es posible que necesite instalar el paquete `python-tk`.

- Realice un programa que cree un fichero de imagen que contenga una representación del *Conjunto de Mandelbrot* (`http://es.wikipedia.org/wiki/Conjunto_de_Mandelbrot`) entre unas coordenadas (x_1, y_1) y (x_2, y_2) que se le preguntarán al inicio del programa al usuario.

- Utilizando expresiones regulares (`http://docs.python.org/3.4/library/re.html`) realice funciones para:
  - Identifique cualquier palabra seguida de un espacio y una única letra mayúscula (por ejemplo: `Apellido N`).
  - Identifique correos electrónicos válidos (empiece por una expresión genérica y vaya refinándola todo lo posible).
  - Identifique números de tarjeta de crédito cuyos dígitos estén separados por `-` o espacios en blanco cada paquete de cuatro dígitos: `1234-5678-9012-3456` ó `1234 5678 9012 3456`.