Laboratorio de Informática I – 2547100
Departamento de Ingeniería Electrónica y de Telecomunicaciones
Facultad de Ingeniería
Universidad de Antioquia

Práctica 6: Programación modular usando funciones

1. Objetivos

- Comprender los conceptos básicos sobre la codificación de funciones en Python.
- Desarrollar programas que involucren programación modular mediante el uso de funciones y módulos.
- Aprender a manipular archivos para los datos de entrada y salida de un programa.
- Aprender a procesar arreglos de dos dimensiones utilizando ciclos anidados.

2. Marco teórico

Las funciones, presentes en la gran mayoría de lenguajes de programación, son la manera como un lenguaje le permite al programador crear sub-programas. Con las funciones, el programador puede encapsular un segmento de código que en la mayoría de los casos va a ser utilizado múltiples veces, o simplemente, le permite organizar un programa de manera modular. Al crear una función se debe dejar claro, por medio de un *docstring*, la especificación que describe las condiciones de los argumentos (datos de entrada) y las garantías del trabajo que hace la función. Las funciones deben ser probadas, una por una, a medida que se van integrando al programa principal.

3. Tareas a realizar

Esta práctica consiste en la implementación de un programa para el juego clásico de **Sudoku**. El programa debe ser desarrollado de manera modular mediante el uso de funciones.

3.1. Resumen del juego

Sudoku es un juego numérico de habilidad mental. El juego consiste en rellenar una cuadrícula de 9 × 9 celdas (81 en total) dividida en sub-cuadrículas de 3 × 3 (también llamados "bloques"). Las celdas deben llenarse con números del 1 al 9, partiendo de algunos números previamente dispuestos en algunas de las celdas que sirven como punto de partida. Al llenar las casillas debemos tener en cuenta que en una misma fila, columna o bloque no puede haber números repetidos. Un sudoku está bien planteado si la solución es única, algo que el matemático Gary McGuire ha demostrado que no es posible si no hay un mínimo de 17 dígitos de pista al inicio del juego. El juego finaliza cuando el usuario llena completamente las 81 casillas conforme a las reglas ya mencionadas.

En esta práctica, usted debe desarrollar un programa en Python que le permita al usuario jugar Sudoku, es decir, que le muestre el tablero de juego a medida que avanza el juego, y que verifique las reglas de éste para decidir si cada número que ingresa el usuario es válido o no. Otros detalles del funcionamiento del programa se explican a continuación.

3.2. Funcionamiento del programa

A continuación se dan los requerimientos que deberá cumplir el juego a implementar:

- Cuando el programa inicia por primera vez deberá imprimir en pantalla un anuncio de bienvenida con el nombre del juego de manera semi-gráfica (ver Figura 1). Note que este mensaje deberá ser cargado desde un archivo.

Figura 1. Bienvenida del Juego de forma semi-gráfica

- Antes de iniciar el juego, el programa deberá seleccionar aleatoriamente entre una serie de juegos iniciales almacenados en archivos para luego imprimir el tablero en su estado inicial. Las celdas conteniendo pistas deberán imprimirse en pantalla de una manera distintiva ya sea cambiando el color del texto o del fondo (ver Figura 2).

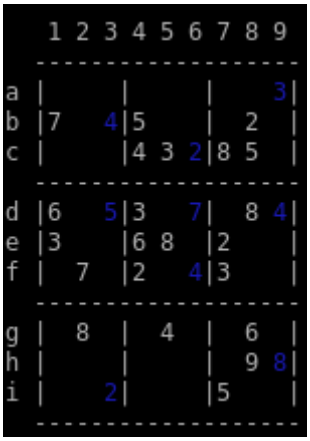

Figura 2. Ejemplo tablero inicial de Juego Sudoku

- Las filas y las columnas del tablero deben estar rotuladas con letras (a, b, c, d, e, f, g, h, i) y números (1, 2, 3, 4, 5, 6, 7, 8, 9) respectivamente. Así el usuario puede referirse a cada celda por las coordenadas de fila y columna: a5, e8, f9, b3, etc (ver Figura 2).
- Al iniciar el juego y luego de imprimir el tablero inicial, el programa entra en un estado de espera para permitir que el usuario por medio de comandos textuales ingrese los dígitos faltantes en el tablero. A cada comando del usuario, el programa responde actualizando el tablero para mostrar el efecto del comando.
- Los comandos disponibles para la interacción con el usuario son los siguientes:
  a. ***Insertar o cambiar valor de celda.*** Permite insertar un número en una celda vacía o cambiar el número que ya estaba en la celda. La sintaxis del comando consiste en indicar la coordenada de la celda seguida del valor que se le quiere asignar. Por ejemplo, para asignar a la casilla de la fila g y columna 8 el valor 5 entonces en consola se debe escribir: ***"g8 5"***. Tenga en cuenta que el programa no puede permitir modificar las celdas que contienen números del juego inicial.
  b. ***Limpiar valor de celda***. Este comando permite borrar el contenido de una celda. Pensado para que una vez detectado un error el usuario pueda limpiar la celda de valores que puedan distraerlo. La sintaxis del comando consiste de la palabra clave ***"clear"*** seguida por la coordenada de la celda. Por ejemplo para limpiar la celda de la fila b y columna 7,

en consola se debe escribir: ***"clear b7***". Recuerde que el programa no puede permitir borrar las celdas que contienen números del juego inicial.

c. ***Limpiar tablero.*** Este comando permite borrar todos los cambios realizados por el usuario para volver al juego inicial. El comando se hace efectivo al escribir en consola las palabras ***"clear board"***. Tenga en cuenta que las celdas con pistas iniciales al iniciar el tablero no pueden ser limpiadas.
d. ***Comando nuevo tablero.*** Este comando aborta el juego actual y carga un tablero nuevo (desde un archivo) seleccionando uno de forma aleatoria. El comando se hace efectivo al escribir la palabra ***"new"***.
e. ***Comando salir del Juego.*** Al escribir ***"exit"*** en la consola, el programa termina.

Figura 3. Ejemplo de interacción del usuario con el juego.

En la Figura 3 se muestra un ejemplo de la interacción del usuario con el juego por medio de algunos comandos.

- A medida que el usuario ingresa comandos, el programa debe chequear la sintaxis de los comandos. Por ejemplo, que las coordenadas ingresadas por el usuario sean correctas o que el valor de la casilla esté en el rango [1,9]. El programa también debe verificar cada vez que se inserte o cambie el valor de una celda si se viola alguna de las reglas del juego e indicarle al usuario.
- Cuando el usuario complete las 81 casillas del Sudoku sin errores, el programa informará al usuario que el juego ha terminado.

3.3. Metodología para el desarrollo del programa

Es importante que en lugar de ir directamente al desarrollo de las funciones, primero se haga un diseño, utilizando diagramas de flujo abstractos, del funcionamiento global del juego, es decir, del algoritmo que deberá implementarse en el módulo main.py. Una vez se tenga claro cómo funcionará el juego, debe hacerse el diseño de cada función para luego implementarlas en Python.

Para el desarrollo del programa se suministran los siguientes archivos que deberán ser descargados de la página del curso:

- **sudoku.py**: archivo donde encontrará la descripción de las funciones que usted deberá implementar y usar en el programa principal del juego Sudoku.
- **main.py:** archivo donde se implementará el programa principal haciendo uso de las funciones implementadas en el archivo sudoku.py
- **sd1.txt, sd2.txt, sd3.txt, sd4.txt y sd5.txt**: archivos de texto plano que sirven de plantilla para la inicialización del tablero. Inspeccione los archivos para entender su formato y lograr que el programa los sepa procesar. Los dígitos en cero representan casillas vacías en el tablero del Sudoku.
- **sudoku.txt**: Archivo que contiene el nombre del juego de manera semi-gráfica.

Comentarios y tips:

- Recuerde y estudie el tema de manipulación de archivos. Para mayor información puede consultar en el siguiente enlace: http://docs.python.org.ar/tutorial/3/inputoutput.html#leyendo-y-escribiendo-archivos
- Tenga en cuenta que el archivo **sudoku.txt** le será entregado y no será necesario que lo modifique o cree nuevos archivos para desplegar la entrada semi-gráfica. Sin embargo, si desea experimentar, consulte el sitio web http://www.network-science.de/ascii/
- Dentro del programa puede encontrase con la necesidad de copiar una lista multidimensional, para esto es necesario utilizar la función ***deepcopy***, la cual recibe como argumento la lista que quiere copiar y retorna la copia de la lista. Para usar deepcopy agregue la siguiente línea al inicio de su código:

  ```
  from copy import deepcopy
  ```
- Para este ejercicio necesitará generar un número aleatorio entre 0 y la cantidad de plantillas disponibles. Para esto use la función randint(a,b) la cual retorna un entero aleatorio N tal que (a <= N <= b). Para poder hacer uso de la función anterior no olvidar importar la librería random dentro de la función.

Opcional

Las características opcionales se premiarán con una bonificación en la nota. Para obtener la bonificación adicional implemente al menos una de las siguientes características adicionales en el programa:

- Agregue los comandos ***"save <filename>"*** y ***"load <filename>"*** para guardar y cargar juegos no completados. El comando ***"save <filename>"*** guardará el estado actual del juego en un archivo de texto con nombre ***filename*** usando el formato que usted elija. Por su parte, el comando ***"load <filename>"*** cargará el tablero de juego en el estado en que fue guardado el juego en el archivo de nombre ***filename***.
- Agregue el comando ***"undo"*** para deshacer último comando de movimiento o comando ***"clear"*** realizado.
- Agregue temporización a la duración del juego y que se lleve el registro del menor tiempo requerido para solucionar cada una de las 5 plantillas. Tenga en cuenta que esta información debe persistir incluso después de cerrado el juego.

4. Evaluación

La evaluación se basará en los códigos enviados y un quiz sobre los temas de la práctica. Además, se tendrá en cuenta una bonificación para quien implemente alguna de las características opcionales.