Capacitación Python
Laboratorio 2

Nelson R. Salinas

Abril 9, 2018

Dado que programar es muy aburrido hoy jugaremos triqui contra el computador. De paso practicaremos la declaración de funciones.

1. El problema

Supondremos que el computador no tiene ninguna aplicación para jugar triqui, así que debemos enseñarle cómo se debe jugar: cual es el tablero y las reglas de juego. ¿Como podemos segmentar el problema? Una técnica puede ser pensar en cada operación como una función:

1. Crear tablero vacío.
2. Mientras existan celdas vacías en el tablero o no exista un ganador:
    a) Mostrar el tablero.
    b) Preguntar jugada al jugador.
        1) Si es el turno del usuario: preguntar celda.
        2) Si es el turno del computador: seleccionar celda aleatoriamente.

A continuación se presentan soluciones a cada uno de los pasos mencionados arriba, acompañados por un texto explicativo. Digite cada una de las funciones proporcionadas en la consola de su predilección. Si utiliza Jupyter trate de digitar cada función en una celda.

2. El tablero

El tablero de triqui es una cuadrícula de 3×3. ¿Como podemos crear un tablero similar en la consola de Python? La solución propuesta es una lista bimensional (con dos niveles), uno para las filas y otro para las columnas. La forma de acceder a esta clase de listas multidimensionales es `mi_lista[índice filas][índice columnas]`.

Código 1: Declaración del tablero.

```
1  def tablero():
2      """Inicializa un tablero nuevo."""
3      mitab = []
4      for fila in range(3):
5          mitab.append([' ', ' ', ' '])
6      return mitab
7
8
9  def ver_tab(tabl):
10     """Imprime el tablero en pantalla."""
11     for fila in tabl:
12         print '|', fila[0], '|', fila[1], '|', fila[2], '|'
```

En el Código 1 está presentada la implementación de dos funciones: una para crear el tablero y otra para visualizarlo. Las celdas de la cuadrícula son inicializadas con un caracter de espacio y los caracteres posibles de asignación son 'X' (para el usuario) o 'O' (para el computador). La visualización simplemente imprime en pantalla cada una de las celdas separada por una pleca ('|').

3. Verificar espacios vacíos

Esta verificación simplemente se realiza comprobando que cada una de las celdas de tablero este vacía, es decir, corresponde a un caracter de texto de espacio, que es el valor asignado al inicializar el tablero. La función que implemente esta rutina retorna un valor booleano: `False` si aún hay espacio y `True` en el caso contrario.

Código 2: Verificación de caldas vacías.

```
1  def lleno(tab):
2      """Verifica si el tablero aun conserva espacios vacios."""
3      out = True
4      for row in tab:
5          for cell in row:
6              if cell == ' ':
7                  out = False
8      return out
```

La función `lleno` itera a través de todas las celdas antes de retornar la respuesta. ¿Existe una manera más eficiente de realizar esta verificación?

4. Verificación de ganador

Para revisar si existe un ganador en el tablero simplemente se confirman todos los casos posibles: primero si alguno de los jugadores completó alguna fila, luego si completó alguna columna, y por último se verifican las diagonales.

Código 3: Verificación del ganador de la partida.

```
def ganador(tab):
    """Determina si existe un ganador en el tablero."""
    gana = 'nadie'

    for row in tab:
        if row == ['X', 'X', 'X']:
            gana = 'usuario'
            break
        elif row == ['O', 'O', 'O']:
            gana = 'compu'
            break

    if gana == 'nadie':
        for ix in [0, 1, 2]:
            if [tab[0][ix], tab[1][ix], tab[2][ix]] == ['X', 'X', 'X']:
                gana = 'usuario'
                break
            elif [tab[0][ix], tab[1][ix], tab[2][ix]] == ['O', 'O', 'O']:
                gana = 'compu'
                break

    if gana == 'nadie':
        if [tab[0][0], tab[1][1], tab[2][2]] == ['X', 'X', 'X']:
            gana = 'usuario'
        elif [tab[0][0], tab[1][1], tab[2][2]] == ['O', 'O', 'O']:
            gana = 'compu'
        elif [tab[0][2], tab[1][1], tab[2][0]] == ['X', 'X', 'X']:
            gana = 'usuario'
        elif [tab[0][2], tab[1][1], tab[2][0]] == ['O', 'O', 'O']:
            gana= 'compu'

    return gana
```

5. Captura de la jugadas

En este punto se requieren dos funciones: una que pregunta al usuario y otra que "pregunta" al computador. En el caso del jugador usuario, se puede utilizar la función `raw_input` de Python, que solicita la entrada de datos interactivamente al usuario y lo guarda en una variable como datos tipo texto (str). El usuario puede ingresar tantos caracteres como desee, sólo tiene que presionar la tecla Enter para terminar. Al usuario se le preguntan coordenadas en el tablero, por lo cual los datos digitados por el usuario deben ser traducidos a tipo `int`, de lo contrario Python no podría acceder a la celda especificada. Por otro lado, se debe realizar una corrección substrayendo 1 al dígito proporcionado por el usuario, ya que los índices en Python comienzan en 0, no en 1.

Código 4: Jugada de usuario.

```
def escoja():
    """Solicita al usuario seleccionar una celda para marcar 'X'."""

    print "Seleccione fila (1-3):"
    f = raw_input()
    f = int(f) - 1

    print "Seleccione columna (1-3):"
    c = raw_input()
    c = int(c) - 1

    return (f,c)
```

La función `escoja` no verifica que los datos ingresados por el usuario sean correctos. ¿Cómo podría realizarse dicha comprobación?

La función `compu` selecciona automáticamente la jugada realizada por el computador. Por simplicidad la selección se realiza completamente al azar, para lo cual se necesita una herramienta que pueda generar números aleatorios. Una de esas herramientas es la función `random` del módulo `random`, que genera un número aleatorio —uniformemente distribuído— en el rango [0, 1).

Código 5: Jugada del computador.

```
from random import random

def compu():
    """Selecciona una celda al azar para marcar 'O'."""
    coors = [None, None]

    for ic in range(2):
        r = random()
        if r < 0.333333:
            coors[ic] = 0
        elif r < 0.666666:
            coors[ic] = 1
        else:
            coors[ic] = 2

    return tuple(coors)
```

6. Uniendo todas las piezas

En este punto todas las pequeñas tareas especificadas al inicio ya están implementadas por medio de funciones, ahora es necesario unirlas coherentemente en una rutina para desplegar el juego.

El primer turno se le adjudica al computador: dado que escoge las jugadas aleatoriamente se encuentra en desventaja. La primera iteración (Código 6, línea 7) confirma las condiciones necesarias para la continuación del juego: aún existen celdas vacías y todavía no existe un ganador. Si alguna de estas dos condiciones no se cumple se imprime el resultado (líneas 31–40).

En cada turno se utilizan las funciones `escoja` y `compu` apropiadamente. Sin embargo estas funciones pueden retornar celdas que ya han sido jugadas anteriormente, por lo cual se debe realizar una iteración hasta que la celda seleccionada esté disponible (línea 10).

Código 6: Función completa del juego.

```
1  def triqui():
2     """Juego interactivo de triqui."""
3
4     tab = tablero()
5     turno = 'compu'
6
7     while not lleno(tab) and ganador(tab) == 'nadie':
8        coors = (5, 5)
9
10       while coors == (5, 5) or tab[coors[0]][coors[1]] != ' ':
11
12          if turno == 'usuario':
13             coors = escoja()
14
15          else:
16             coors = compu()
17
18       print "Jugada", turno
19
20       if turno == 'usuario':
21          tab[coors[0]][coors[1]] = 'X'
22          turno = 'compu'
23
24       else:
25          tab[coors[0]][coors[1]] = 'O'
26          turno = 'usuario'
27
28       ver_tab(tab)
29       print "\n"
30
31    print "Resultado final:"
32    ver_tab(tab)
33
34    estgan = ganador(tab)
35
36    if estgan == 'usuario':
37       print "Felicidades, ganaste!"
38
39    elif estgan == 'compu':
40       print "El azar es mejor jugador de triqui que tu. Tu existencia
```

```
            en este planeta deberia ser reconsiderada seriamente."
41
42      else:
43          print "Empate!"
```