Actividad 03 - Diez mil (simplificado)

Pensamiento Computacional

Primer Cuatrimestre 2024

El objetivo de esta actividad es realizar un programa en `Python` que *simule* varios individuos jugando con dados.

Vamos a escribir un programa en `Python` que *juegue* a los dados. Con suerte, en algún momento, llegaremos programar el `Diez mil S` (S de simplificado), un entretenimiento basado en el famoso *Diez mil* pero con reglas simplificadas. Es un juego muy popular llamado así porque el objetivo original es llegar a los 10.000 puntos. Forma parte de nuestro folklore, divulgado de boca en boca a lo largo de generaciones, con muchas variantes. Acá tendremos algunas especialmente pensadas para esta instancia inicial del taller.
A lo largo de esta guía utilizaremos dados equilibrados (no hay trampas), de forma tal que las caras caen con igual probabilidad.

Otras herramientas útiles de Python

Para que estén disponibles más funciones de `Python`, tenemos que utilizar el comando `import`. En particular, en esta actividad vamos a usar funciones implementadas en `random`, para lo cual necesitamos ejecutar `import random`.

Diez mil Simplificado

Vamos a introducir las reglas: en cada **jugada** se lanzan 5 dados y se calcula el puntaje obtenido de la siguiente manera: por cada uno obtenido, se suman 100 puntos, mientras que 3 unos suman 1000 puntos, 4 unos suman 1100, y 5 unos suman 10000 puntos. Además, por cada cinco obtenido se suman 50, mientras que 3 cincos suman 500, 4 cincos suman 550, y 5 cincos suman 600 puntos. Los dados dos, tres, cuatro y seis no suman puntos. Se juega por rondas. En cada ronda cada participante realiza una **jugada**, como se acaba de describir. Al finalizar la ronda se suman los puntos obtenidos por cada jugador a los que ya tenía. Cada participante comienza con 0 puntos.
El juego termina cuando al cabo de una ronda algún jugador alcanza (o supera) los 10.000 puntos, o saca 5 unos. En este caso ganan todos aquellos que superen dicho puntaje o sacan 5 unos. Vamos a utilizar `k` para denotar la cantidad de jugadores.
Una típica duda de domingo por la tarde es decidir si jugar o no al 10.000. Surge, en muchos casos, la siguiente pregunta: En promedio, ¿cuántas rondas tiene una partida con 10 jugadores?
Para dar respuesta a esta duda existencial de las familiar argentinas, comencemos implementando las siguientes funciones:

1. `tirar_cubilete()`: devuelve una lista con los valores de los 5 dados tirados. **Cabe notar que esta función no recibe parámetros**.

   Para probar en consola: Correr la función `tirar_cubilete()` varias veces para asegurarnos que devuelva lo que esperamos.

2. `cuantos_hay(elemento, lista)`: toma un *elemento* y una *lista* y devuelve la cantidad de veces que *elemento* aparece en *lista*.

3. `puntos_por_unos(lista_dados)`: toma una lista de valores obtenidos y devuelve el puntaje obtenido por la cantidad de unos en *lista dados* según las reglas actuales.

4. `puntos_por_cincos(lista_dados)`: toma una lista de valores obtenidos y devuelve el puntaje obtenido por la cantidad de cincos en *lista dados* según las reglas actuales.

5. `total_puntos(lista_dados)`: toma una lista de valores obtenidos y devuelve el puntaje obtenido según las reglas actuales.

6. `jugar_ronda(cant_jugadores)`: simula una ronda de jugadas y devuelve una lista con los puntos obtenidos por cada uno de los *cant jugadores* jugadores.

7. `acumular_puntos(puntajes_acumulados, puntajes_ronda)`: recibe como parámetro una lista con los *puntajes acumulados* de cada jugador y los resultados de la última ronda (*puntajes ronda*). La función devuelve una lista de puntajes actualizados.

   Para probar en consola: Creá dos variables llamadas `lista_prueba1` y `lista_prueba2` usando la función `jugar_ronda()`. Luego pasaselas como parámetros a `acumular_puntos(lista_prueba1, lista_prueba2)` para asegurarte de que funciona bien. (Siempre en la consola)

8. `hay_10mil(puntajes_acumulados)`: toma como parámetro una lista con *puntajes acumulados* por los jugadores, y devuelve el valor lógico `True` si algún puntaje es mayor o igual a 10000 o `False` si no.

   Para probar en consola: Probá tu función dandole como argumento estas listas: [100, 10000, 50, 0], [50, 0, 50, 100]

9. `partida_completa(cant_jugadores)`: Simula rondas del juego con *cant jugadores* jugadores hasta que algún jugador llega a sumar 10000 puntos. Devuelve la cantidad de rondas jugadas.

   Para pensar Esta función solo te devuelve la cantidad de rondas que fueron necesarias. ¿Cómo te podes asegurar que el juego evolucionó correctamente?

10. Vamos ahora a utilizar las funciones implementadas para responder las siguientes preguntas:

    a) En promedio, ¿cuántas rondas tiene una partida con `cant_jugadores`=10 jugadores?

    b) ¿Qué chances hay de terminar una partida con `cant_jugadores`=10 jugadores (que algún jugador alcance los diez mil puntos) si solo tenemos tiempo para jugar a lo sumo 18 rondas?

    Para responderlas, vamos a simular $Nrep = 10000$ partidas completas con `cant_jugadores`=10 jugadores y guardamos en cada paso el resultado del numero de rondas jugadas en una lista llamada `cant_rondas`. Para aproximar la probabilidad pedida se puede contar la cantidad de partidas que terminan en a lo sumo 18 rondas, y dividir por la cantidad de partidas jugadas totales (en este caso, $Nrep = 10000$).

    Anticipar qué resultados deberían obtener con `cant_jugadores`=20 jugadores. Repetir la simulación, pero con `cant_jugadores`=20. ¿Los resultados obtenidos coinciden con los anticipados?

Nota: en caso de que existan funciones de `Python` que resuelvan alguno de los ítems pedidos total o parcialmente, **no es posible** utilizarlas (salvo que esté explícitamente mencionado).