Actividad 01 - Diez mil (simplificado)

Exactas Programa

Verano 2022

El objetivo de esta actividad es realizar un programa en `Python` que *simule* varios individuos jugando con dados. **Deben subir la resolución a:** PENDIENTE .

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.

Descargar el archivo `diez_mil.py` y completar cada una de las funciones pedidas para que realicen las tareas que se describen a continuación.

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`.

Calentando Motores - Juego 1 - Reglas

Este primer juego (j1) consiste en tirar n vces un dado y gana el que NUNCA saca un seis. Es decir, cada jugador tira un dado n veces; podemos pensar, a modo de ejemplo, que $n = 8$ pero hagamos el programa pensando que n va a ser un parámetro (recordemos que, para este curso, llamamos parámetro a los valores que recibe una función). Al terminar con los lanzamientos se examinan los valores que obtuvo cada jugador y ganan quienes no sacaron el número 6.

1. `tirar_dado()`: esta función nos devuelve un valor al azar entre 1 y 6. No toma ningún parámetro. **Pista**: Se puede utilizar la función de `Python` `random.randint(d, h)` del módulo `random` para simular el dado. Explorar qué valores deberían ser "d" y "h".

2. `tirar_varias_veces(n)`: recibe como parámetro la cantidad n de veces a tirar el dado. Esta función debe devolver una lista con los n valores obtenidos.

3. `es_tirada_ganadora(lista_dados)`: recibe como parámetro una lista de valores obtenidos *lista_dados* y devuelve `True` si no hay ningún 6 en los valores pasados o `False` caso contrario.

4. `jugar_varios_j1(k, n)`: recibe como parámetro la cantidad k de jugadores y el número n de veces que cada uno tira el dado. Esta función debe devolver una lista que indique si cada jugador ganó o perdió.

Utilizando las funciones definidas, podemos aproximar la probabilidad de ganar, haciendo jugar a muchos jugadores, y calculando qué proporción de ellos ganan. Simulando 50.000 partidas (es decir,

$k = 50.000$ jugadores) aproximar la probabilidad de ganar tirando el dado $n = 8$ veces (esto se puede hacer contando la cantidad de veces que se gana, dividido la cantidad de veces que se juega). ¿Qué pasa si la cantidad de veces que se tira el dado sube a $n = 10$?

Poné Segunda - Juego 2 - Reglas

En este segundo juego (j2) sofisticamos un *poquito* las reglas. Ahora no prefijamos la cantidad de veces que lanzamos el dado. Por el contrario, cada jugador va tirando un dado hasta que le sale el número 6 por primera vez. En ese momento, para de tirar y se suman todos los valores obtenidos (incluyendo el 6). Si el resultado de la suma es 20 gana, si no pierde.

Para este segundo juego implementaremos las siguientes funciones:

1. `tirar_hasta_seis()`: devuelve una lista con los valores observados en el dado tirado hasta que se obtiene un 6, incluyendo el 6 por último valor de la lista.

2. `cuanto_suman(lista_dados)`: toma una lista de valores obtenidos `lista_dados` y devuelve la suma de los elementos.

3. `jugar_varios_j2(k)`: recibe como parámetro la cantidad k de jugadores. Esta función debe devolver una lista que indique si cada jugador gana (sus dados suman 20) o pierde (no suman 20).

Con estas nuevas reglas, aproximar simulando 50.000 partidas la probabilidad de ganar.

Diez mil Simplificado (optativo)

Este es un tercer juego (j3) *varios poquitos* más complejo; y bastante más cercana al verdadero 10mil! Una buena excusa para seguir pensando. Ahora, en cada **jugada** se lanzan 5 dados y se calcula el puntaje obtenido de la siguiente manera: por cada 1 obtenido, se suman 100 puntos, y por cada 5 obtenido, se suman 50. 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. En este caso ganan todos aquellos que superen dicho puntaje. Vamos a asumir que la cantidad de jugadores está fija en $k = 4$.

Para esta tercera versión realizaremos las siguientes funciones:

1. `tirar_dados()`: devuelve una lista con los valores de los 5 dados tirados.

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

3. `jugar_ronda()`: simula una ronda de jugadas y devuelve los puntos obtenidos por cada jugador.

4. `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. La función devuelve una lista de puntajes actualizados.

5. `hay_10mil(puntajes)`: toma como parámetro una lista con puntajes de los jugadores, y devuelve True si algún puntaje es mayor o igual a 10mil.

6. `jugar_varios_j3()`: inicializa los puntajes en 0 para los 4 jugadores. Simula rondas del juego hasta que algún jugador llega a sumar 10mil puntos (atención! podría ocurrir que dos ganen en la misma ronda). Devuelve el número de jugador que llegó primero a 10mil (un entero entre 0 y 3).

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).