Taller 2 Python, ejercicio de codificación.

a. En este ejercicio se están definiendo 4 clases: Apostador, ComisionJuegoEspectaculos, Loteria, main.

b. La línea de código: if __name__ == "__main__": sirve para verificar que el código dentro de ese bloque se ejecute sólo cuando el archivo es ejecutado directamente como el programa principal, y no cuando se importa desde otro módulo.

c. Si retiro esa línea de código, el bloque de pruebas se ejecutará siempre que el archivo sea ejecutado o importado, lo cual puede ser indeseado si solo buscas importar las clases. No genera error, el programa seguirá corriendo.

d. Se están creando 2 objetos de la clase Apostador.

e. Los objetos de la clase Apostador que se están creando son: apostador1, apostador2.

f. Cuando se ejecuta el programa principal, la variable self de la línea 15 de la clase Apostador está haciendo referencia a la instancia de la clase Apostador que está ejecutando el método play, al que está haciendo la apuesta.

g. Se crea un objeto de la clase Loteria cada vez que un Apostador ejecuta el método play (siempre y cuando tenga suficiente dinero en su wallet). Al ejecutar el código, serán 2: uno para apostador1 y otro para apostador2.

h. Con este cambio, el código imprimirá: 300 (el valor de apostador.wallet luego de depositar los 300), "Necesitas poner más dinero en tu wallet" (mensaje de error porque el wallet es insuficiente para cubrir la apuesta de 400).

i. Con ese cambio, el código de salida dependerá del valor aleatorio generado para a = random.randint(0, 1). Si gana, se imprimirá: 400, Has ganado 480.0, 480.0 y, si pierde, se imprimirá: 400, Has perdido lo que apostaste, 0.

j. En la clase Loteria, el único atributo que hace referencia a un objeto es self.apostador.

k. En la clase Loteria, los atributos que hacen referencia a tipos primitivos son: self.value (de tipo int o float) y self.probability (de tipo float).

l.
```
@classmethod
    def changeProbability (cls, nprobability):
        cls.probability = nprobability
```

m. La línea de código para llamar al método de clase changeProbability sería la siguiente: Loteria.changeProbability(nprobability)

n. Sí, ambas formas funcionan correctamente y serían válidas Ambos enfoques funcionan porque el atributo probability es de clase y ambas formas se pueden utilizar para modificarlo, sin embargo, usar cls es la forma más general y adaptable.

o. La clase Loteria tiene 5 métodos en total, luego de agregarle el nuevo método (changeProbability): __init__, payMoney, recieveMoney, playGame, changeProbability.

p. No, si apostador1 gana, no necesariamente apostador2 también ganaría. Cada apostador tiene su propia instancia de la clase Loteria, con su propia probabilidad y resultado del juego.

q. Si conviertes probability en una constante, no debería modificarse, por lo que el uso del método changeProbability ya no sería correcto, ya que su propósito sería modificar la probabilidad, lo cual va en contra del concepto de "constante". En este caso, es mejor no permitir la modificación del valor de probability una vez asignado.

r. Ambos métodos, gain() y commission(), de la clase ComisionJuegoEspectaculos retornan valores float.

s. La variable self hace referencia al objeto de tipo Loteria que se está utilizando en ese momento (la instancia en curso del juego). No se puede omitir el uso de self en este caso, porque el constructor de ComisionJuegoEspectaculos necesita saber de qué instancia de Loteria se trata para poder realizar los cálculos correctos.

t. El parámetro value pasa por valor porque es un tipo inmutable (un número), y cualquier cambio no afectará a la variable fuera de la función. Por otro lado, el parámetro self pasa por referencia porque es un objeto mutable, por lo que la clase Loteria puede acceder y modificar los atributos del objeto Apostador que se le pasa.