Preguntas de análisis

a) ¿Cuántas clases se están definiendo en este ejercicio?

Tiene tres clases (Apostador, ComisionJuegoEspectaculos, Loteria)

b) ¿Para qué sirve la línea de código if __ name = “ _main_ ”:?

Esa línea se asegura de que el código solo se ejecute si se está usando este archivo directamente, no si es importado en otro programa. Es como decir: "Solo corre este código si este archivo es el principal”.

c) ¿Qué sucede si retiro la línea de la pregunta anterior en nuestro código?, ¿Este sigue corriendo o hay error? Explique en ambos casos

Eliminando la línea 2	Sin eliminar
500 Has ganado 720.0 820.0 500 Has ganado 720.0 820.0	500 Has ganado 720.0 820.0 500 Has ganado 720.0 820.0

- **Con if __name__ == "__main__":**, el código solo se ejecuta cuando lo ejecutas directamente, no cuando lo importas.
- **Sin if __name__ == "__main__":**, el código se ejecuta tanto cuando lo ejecutas directamente como cuando lo importas

Así que el código funcionará igual si lo ejecutas directamente, pero el comportamiento cambia si se utiliza el código en otro archivo.

d) ¿Cuántos objetos de la clase Apostador se están creando?
Se están creando dos objetos de la clase Apostador

e) ¿Cuáles objetos de la clase Apostador se están creando?
Los objetos son **apostador1** y **apostador2**.

f) ¿A quién está haciendo referencia la variable self de la línea 15 de la clase Apostador cuando se ejecuta el programa principal?
self dentro del método play de la clase Apostador se referirá a **apostador1**, que es el objeto de la clase Apostador creado en el programa principal.

g) ¿Cuántos objetos de la clase Loteria se están creando?
En el código, **un solo objeto de la clase Loteria** se crea cada vez que un apostador llama al método play y tiene suficiente dinero en su billetera para hacer la apuesta.

- En la línea 4 del main.py cambiar el apostador1.deposit(500) por apostador1.deposit(300)

Antes del cambio	Después
500 Has perdido lo que apostaste 100 500 Has ganado 720.0 820.0	300 Necesitas poner mas dinero en tu wallet 300 500 Has perdido lo que apostaste 100

h) ¿Qué imprimiría el código por parte del apostador1?
Necesitas poner mas dinero en tu wallet.

i) ¿Qué imprimiría el código por parte del apostador2?

Si el número aleatorio es menor que 0.5, el apostador gana, Si el número aleatorio es mayor o igual a 0.5, el apostador pierde.

Gana	Pierde
300 Necesitas poner mas dinero en tu wallet 300 400 Has ganado 720.0 720.0	300 Necesitas poner mas dinero en tu wallet 300 400 Has perdido lo que apostaste 0

j) ¿Cuáles atributos de la clase Lotería están haciendo referencia a objetos?

self.apostador: Este atributo hace referencia al objeto de la clase Apostador que está participando en el juego de lotería.

k) ¿Cuáles atributos de la clase Lotería están haciendo referencia a tipos primitivos?
self.value (número para la apuesta) y self.probability (número flotante para la probabilidad de ganar).

l) ¿Complete las siguientes líneas para que en la clase Loteria, se implemente el método de clase changeProbability?

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

m) ¿Cómo sería la línea de código para llamar el método changeProbability?
Loteria.changeProbability( ).

n) ¿Es correcto si en el método changeProbability que se creó, cambiar lo siguiente? Explique:

No es correcto.

Usar cls.probability = nprobability es lo adecuado porque hace referencia a la clase que llama el método, lo que lo hace más flexible, especialmente si se usa herencia.

En cambio, Loteria.probability = nprobability solo funciona para la clase Loteria y no sería útil en subclases.

o) ¿Cuántos métodos tiene la clase Loteria después de agregarle el nuevo método?
La clase Loteria tiene **4 métodos** después de agregar el nuevo método changeProbability. Estos son:

1. __init__(self, value, apostador)
2. payMoney(self, gain)
3. recieveMoney(self)
4. playGame(self)
5. changeProbability(cls, nprobability) (nuevo)

p) ¿Si el apostador1 gana el apostador2 también? Explique por qué pasa en caso de ser sí o no.
No, no necesariamente. El resultado de la lotería para cada apostador es independiente. Aunque ambos apostadores apuesten la misma cantidad, cada uno tiene su propia instancia de la clase Loteria, por lo que el resultado del juego se determina de forma aleatoria e individual para cada apostador. Cada uno tiene su propio valor de probability, y el número aleatorio (a) se genera por separado para cada uno.

q) ¿Qué sucede si decido cambiar el atributo de clase probability a una constante? ¿Se considera correcto el uso del método changeProbability teniendo en cuanta este nuevo cambio?
Si probability se convierte en una constante (es decir, un valor que no cambia), entonces el método changeProbability no tendría sentido. El propósito del método changeProbability es cambiar un valor que puede modificarse, pero si probability es constante, no se debe cambiar y el uso de changeProbability sería innecesario.

r) ¿Cuál es el tipo de retorno de los métodos gain() y commission() de la clase ComisionJuegoEspectaculos?
Ambos métodos retornan un **número decimal (float)** que representa el monto de la ganancia o comisión calculada.

s) ¿A quién está haciendo referencia la variable self de la línea 18 de la clase Loteria cuando se ejecuta el programa principal? ¿Podría omitirse el uso de la variable self en este caso?
La variable self hace referencia a la instancia actual de la clase Loteria en la que se está ejecutando el método. En la línea 18, self.probability accede al atributo probability de esa instancia de Loteria. No se puede omitir self en este caso porque es necesario para acceder a los atributos y métodos de la instancia. Sin self, el código no sabría a qué instancia de la clase Loteria se refiere.

t) ¿En la línea 15 de la clase apostador vemos como la clase recibe dos parámetros (value, self) especificar cuál de estos pasa por valor y cuál por referencia y por qué?

- **self**: Pasa **por referencia** porque se refiere a la instancia de la clase Apostador. Siempre que se pasa self, se está pasando la referencia al objeto actual.
- **value**: Pasa **por valor** porque es un valor que se recibe como argumento en el método. El valor de value (como un número entero) no se modifica directamente fuera del método; solo se usa dentro del método para realizar operaciones.