SOLUCIÓN TALLER # _ (PYTHON/JAVA)

ASIGNATURA

Programación Orientada a Objetos

PROFESOR

Jaime Alberto Guzmán Luna

ESTUDIANTE

SAMUEL TORO AGUDELO

NÚMERO DE IDENTIFICACIÓN

1020107389

Universidad Nacional de Colombia

Sede Medellín

2024

RESPUESTA PREGUNTAS

a) En este ejercicio se están definiendo tres clases: Apostador, ComisionJuegoEspectaculos y Loteria.

b) La línea de código if __name__ == "__main__": se utiliza para verificar si el script se está ejecutando directamente o se está importando como un módulo. Si el script se está ejecutando directamente, __name__ se establece en "__main__" y el código dentro de este bloque se ejecuta.

c) Si retiras la línea if __name__ == "__main__":, el código seguirá funcionando si lo estás ejecutando directamente. Sin embargo, si importas este script como un módulo en otro script, todo el código que estaba dentro de ese bloque se ejecutará inmediatamente.

d) Se están creando dos objetos de la clase Apostador: apostador1 y apostador2.

e) Los objetos de la clase Apostador que se están creando son apostador1 y apostador2.

f) La variable self en la línea 15 de la clase Apostador hace referencia al objeto Apostador que está llamando al método play. Cuando se ejecuta el programa principal, self hará referencia a apostador1 cuando apostador1 llame al método play, y a apostador2 cuando apostador2 llame al método play.

g) Se están creando tantos objetos de la clase Loteria como veces se llame al método play de un objeto Apostador. En el programa principal, se llaman dos veces al método play, por lo que se crean dos objetos de la clase Loteria.

h) Si se cambia apostador1.deposit(500) por apostador1.deposit(300), el código imprimirá 300 después de la línea print(apostador1.wallet). Luego, si apostador1 gana el juego, imprimirá "Has ganado " seguido del total ganado, y luego imprimirá el nuevo saldo de apostador1.wallet. Si apostador1 pierde, imprimirá "Has perdido lo que apostaste" y luego imprimirá el nuevo saldo de apostador1.wallet.

i) Si se cambia apostador2.deposit(500) por apostador2.deposit(400), el código imprimirá 400 después de la línea print(apostador2.wallet). Luego, si apostador2 gana el juego, imprimirá "Has ganado " seguido del total ganado, y luego imprimirá el nuevo saldo de apostador2.wallet. Si apostador2 pierde, imprimirá "Has perdido lo que apostaste" y luego imprimirá el nuevo saldo de apostador2.wallet.

j) El atributo apostador de la clase Loteria está haciendo referencia a un objeto de la clase Apostador.

k) Los atributos value y probability de la clase Loteria están haciendo referencia a tipos primitivos (float y int).

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

m) Para llamar al método changeProbability, se hace de la siguiente manera: Loteria.changeProbability(nueva_probabilidad), donde nueva_probabilidad es la nueva probabilidad que quieres establecer.

n) Sí, es correcto cambiar cls.probability = nprobability por Loteria.probability = nprobability en el método changeProbability. Ambas líneas hacen lo mismo: cambian el valor del atributo de clase probability de la clase Loteria.

o) Después de agregar el nuevo método changeProbability, la clase Loteria tiene seis métodos: __init__, payMoney, recieveMoney, playGame, changeProbability y el método especial __init__.

p) No necesariamente. Cada vez que un apostador juega, se genera un número aleatorio que determina si gana o pierde. Por lo tanto, es posible que apostador1 gane y apostador2 pierda, o viceversa, o que ambos ganen o ambos pierdan.

q) Si se cambia el atributo de clase probability a una constante, ya no se podra cambiar su valor con el método changeProbability. En este caso, el uso del método changeProbability ya no sería correcto, ya que su propósito es cambiar el valor de probability.

r) Los métodos gain() y commission() de la clase ComisionJuegoEspectaculos retornan un float, que representa la ganancia después de deducir la comisión.

s) La variable self en la línea 49 de la clase Loteria hace referencia al objeto Loteria que está llamando al método playGame. No se puede omitir el uso de la variable self en este caso, ya que se utiliza para acceder a los atributos y métodos del objeto Loteria.

t) En la línea 15 de la clase Apostador, self pasa por referencia y value pasa por valor. self es una referencia al objeto Apostador que está llamando al método, y value es un valor primitivo que se pasa al método. En Python, los objetos mutables como las listas,

los diccionarios y las instancias de clases se pasan por referencia, mientras que los tipos primitivos como los enteros, los flotantes y las cadenas se pasan por valor.