Solución taller 2 de Python

By: Andrés Felipe muñoz Ortiz

T.I: 1011395924

Solución:

a. Tres clases: Apostador, ComisionJuegoEspectaculos y Loteria
b. esta línea asegura que la creación de instancias de Apostador y las pruebas de juego solo se ejecuten cuando main.py se ejecuta directamente.
c. Sí se elimina la línea o if __ name = “ _main_ ”: entonces el código se ejecutará siempre que el archivo sea ejecutado directamente como modulo

d y e. En el archivo se están creando dos objetos de la clase apostador: apostador1 y apostador2

f. Cuando se ejecuta el programa principal, self dentro de Apostador hace referencia al objeto específico de la clase Apostador que está llamando al método, ya sea apostador1 o apostador2

g. Se crea un objeto por cada llamado al método play, ósea uno por cada apostador, dos en total

h. imprimirá: 300 Necesitas poner mas dinero en tu wallet 300

i. Podría imprimir dos posibles resultados dependiendo de la pseudoaletoriedad del método random, si gana imprimirá: 400

Has ganado 400

400 y si pierde impriría:

400

Has perdido lo que apostaste

0

j. self. Apostador hace referencia a un objeto de tipo Apostador y self.Loteria a un objeto de tipo Loteria en la clase ComisionJuegoEspectaculos

k. self.value hace referencia a un tipo primitivo numérico, int o float y self.probability referencia a un tipo primitivo float

l. @classmethod

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

m. Loteria.changeProbability(nuevo valor de la probabilidad)

n. Ambas líneas son correctas y lograrán el mismo resultado, modificando el atributo de clase probability. Usar cls.probability es más flexible y sigue las convenciones de los métodos de clase, ya que hace referencia a la clase a través del parámetro cls. Por otro lado, Loteria.probability también funciona, pero es más explícito y menos flexible si el nombre de la clase cambia.

o. Tendrá 5: __init__, payMoney, reciveMoney, playGame y changeProbability

p. No necesariamente, esto puede pasar dependiendo la aletoriedad del juego, pero cada apostador tienen un juego independiente

q. Si se diera el cambio no se podría usar changeProbability, ya que este método está destinado a modificar el valor de un atributo que no debería cambiar por lo que podría arrojar error

r. Ambos tienen un tipo de retorno float

s. self hace referencia a la instancia actual de la clase loteria, y no podría omitirse ya que es necesario pasar la instancia de Loteria a ComisionJuegoEspectaculos para que pueda acceder a los atributos de Loteria

t. value pasa por valor porque es un tipo primitivo que no altera su valor fuera de la funciopn, y self por referencia porque es una instancia de Apostador y cualquier cambio afectaría al original