# Respuestas Taller 2 POO Python

a) En el ejercicio se están definiendo en total 4 clases que son las siguientes: Apostador, ComisionJuegoEspectaculos, Loteria y la clase Main.

b) La línea de código “if __name__ == "__main__":” sirve para controlar o modular tu código, es decir, todo lo que escribas después de esa línea de código se va a ejecutar únicamente cuando el archivo sea ejecutado directamente y no desde otro módulo o paquete. Esto es importante ya que te permite crear funcionalidades que solo quieres que se ejecuten directamente en cierto archivo y así evitas que se ejecuten cuando importas tu archivo en otro módulo manteniendo así un orden y estructura en tu código.

c) Si elimino la línea “if __name__ == "__main__":” realmente ocurre un error pero es por la identacion, sin embargo, entiendo que a esto no se refiere la pregunta así que respondiendo de verdad a la pregunta, si eliminas esa línea de código en este caso en particular no ocurre ningún error y sigue funcionando igual porque esta línea se encuentra en la clase main, esto quiere decir que esta clase jamás va a ser solicitada o importada en alguna otra clase, al contrario, es esta clase main la que se encarga de importar todas las clases creadas y como estos otros archivos que contienen las otras clases en ningún momento hacen algo diferente a crear y definir clases y métodos, entonces jamás van a ejecutar algo que imprima o muestre resultados, por lo cual, la única que imprime cosas es la clase main que sigue funcionando igual con o sin esa línea de código. Si alguna otra clase tuviera alguna funcionalidad que imprime algo por pantalla entonces ahí si afectaría el quitar el “if __name__ == "__main__":”, pero repito nuevamente, en este caso en particular no afecta en nada.

d) Se están creando en total dos objetos de la clase Apostador.

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

f) En la línea 15 de la clase Apostador “loteria = Loteria(value, self)” el self se está pasando como parámetro para la instancia que se está creando de la clase Loteria y a su vez está haciendo referencia a la instancia de la clase Apostador que creó a la instancia de Loteria.

g) En todo el código se crea solamente un objeto de la clase Loteria y es justamente en la línea 15 de la clase Apostador: loteria = Loteria(value, self)

h) Si cambias en la línea 4 del main.py el “apostador1.deposit(500)” por “apostador2.deposit(300)” el código va a imprimir el siguiente mensaje: “Necesitas poner mas dinero en tu wallet” porque por como esta estructurado el método play de la clase Apostador que dice que el dinero que tu tengas debe de ser mayor o igual a la cantidad de plata que vas a apostar, en este caso en concreto más adelante dice “apostador1.play(400)” y se evidencia claramente que el dinero que tiene el apostador (300) es menor que el dinero que quiere apostar (400).

i) Si cambias en la línea 10 del main.py el “apostador2.deposit(500)” por “apostador2.deposit(400)” el código realmente se va a seguir ejecutando con normalidad por lo mismo que expliqué en la pregunta anterior con la única diferencia

que en este caso la cantidad de dinero que el apostador quiere apostar es la misma cantidad de dinero que tiene el apostador (400) por lo que no debe haber ningún error.

j) El único atributo que está haciendo referencia a un objeto en la clase Loteria es el atributo apostador que está haciendo referencia a una instancia de la clase Apostador.

k) Los atributos de la clase Loteria que están haciendo referencia a tipos primitivos son el atributo “value” (que esta haciendo referencia a un int) y el atributo probability (que está haciendo referencia a un float).

l) El método de clase con changeProbability con las líneas completadas quedaría de la siguiente manera:

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

m) La línea de código para llamar el método changeProbability debería ser la siguiente: Loteria.changeProbability(123) (nota: el 123 es solo un valor cualquiera que hace referencia a que hay que pasar por lo menos un parámetro), sin embargo, técnicamente también se puede llamar directamente desde una instancia de la clase Loteria pero es una práctica que no es recomendada, por lo cual la forma correcta es llamarlo usando la clase ya que se trata de un método de clase.

n) Si cambias cls.probability = nprobability por Loteria.probability = nprobability en el método changeProbability técnicamente debería de ser correcto y no habría ningún problema ya que cls es simplemente una referencia a la clase Loteria así que ambas formas dan el mismom resultado.

o) La clase Loteria ahora tiene en total 4 métodos después de añadirle el nuevo método y estos son: payMoney, recieveMoney, playGame y changeProbability.

p) No necesariamente gana el apostador2 si el apostador1 también gana y viceversa, en realidad es una probabilidad del 50% ya que en el método playGame de la clase Loteria se define un número aleatorio entre 0 y 1 (en la línea que dice: a = random.randint(0,1)) y luego dice que para ganar este numero aleatorio debe de ser menor que la probabilidad (que está definida en 0.5) por lo cual en teoría es una probabilidad del 50%.

q) Dentro del funcionamiento de Python todo funcionaría igual si decides cambiar el atributo probability a una constante ya que como tal Python no puede definir constantes como en Java, sin embargo, en la teoría sería incorrecto el uso del método changeProbability porque al tratarse de una constante debería de ser un valor que jamás se va a cambiar y esto es precisamente lo que hace el método, por lo cual seria incorrecto usar y crear este método.

r) El tipo de retorno de los métodos gain() y commission() de la clase ComisionJUegoEspectaculos debería de ser un float.

s) La variable self de la línea 18 de la clase Loteria “commi =ComisionJuegoEspectaculos(self)” hace referencia a la instancia u objeto de la clase Loteria que está creando a este nuevo objeto “commi” y no se puedo omitir usar la variable self en este caso ya que el objeto “commi” requiere de ese parámetro para ser creado y si se omite entonces saldrá un error en la ejecución.

t) En la línea 15 de la clase Apostador “loteria = Loteria(value, self)” la variable value está siendo pasada por valor y la variable self esta siendo pasada por referencia porque self está haciendo referencia al objeto de clase apostador que está creando esta nueva instancia “loteria” de la clase Loteria.