Taller Poo en Python

Santiago Barrientos Medina

Universidad Nacional de Colombia

Programación orientada a objetos

Jaime Alberto Guzmán

Grupo 2

Facultad de Minas

Ingeniería en sistemas en informática

a). En este ejercicio se están definiendo tres clases:

-Apostador

-comisionJuegosEspectaculos

-Loteria

b).sirve para que el código que hay dentro de esta línea de código solo se ejecute desde el propio modulo y cuando no sea asi, que no se ejecute nada, razón por la que cuando uno va a ejecutar el código la clase loteria o apostador el código no corre. Es decir , lo ponemos cuando queremos que el código solo se ejecute en ese modulo y no desde afuera

c).si quitamos esta línea de código, el programa seguirá funcionando pero ahora si se puede ejecutar desde otro modulo, perderíamos el control que nos da la línea de código y puede causar errores no deseados

d).se están creando dos objetos de la clase apostador

e).los objetos de la clase apostador que se están creando son:

Apostador1

Id:1 , Nombre:”Juan”,phone_number:302 y email: "j@gmail.com"

Apostador2

Id:2,Nombre:”Ricardo”,phone_number:548 y email: r@gmail.com

f).cuando se ejecuta el programa, esa variable self de la línea 15 esta haciendo referencia al objeto sobre el que se esta ejecutando el método. Esta variable es usada para hacer un paso por referencia , mas en concreto de un objeto de tipo apostador

g).se están creando dos objetos de la clase loteria

h).por parte del apostador 1 el código imprimirá :

-“300” al imprimir el atributo wallet del apostador 1

-“Necesitas poner mas dinero en tu Wallet, esto al invocar el método play

-“300” al imprimir el atributo wallet del apostador 1

i)por parte del apostador 2 el código imprimirá :

-“400” al imprimir el atributo Wallet del apostador 2

-“has perdido lo que apostaste” si cuando se invoca el método playgame el numero que sale en randint es 0(POSIBILIDAD 1)

-“has ganado 720.0” si cuando se invoca el método playgame el numero que sale en el randint es 1(POSIBILIDAD 2)

-“0” si se cumple la posibilidad 1

-720.0 si se cumple la posibilidad 2

j).el atributo de la clase lotería que esta haciendo referencia a objetos es el atributo apostador que esta haciendo referencia a un objeto de tipo Apostador

k).los atributos que están haciendo referencia a tipos primitivos son :

-probabilty que esta haciendo referencia a un tipo de dato float

-value que luego en la ejecución del programa hace referencia a un int

l).las líneas quedarían asi:

-@classmethod

-def changeProbability(cls,nprobability)

-cls.probability=nprobability

m).la línea seria asi:

-Loteria.changeProbability(0.21) el 0.21 lo pongo como ejemplo de lo que se le podría entregar

-también lo podríamos llamar desde un objeto. Supongamos que creamos el objeto loteria1, pues bueno, podríamos llamar al método de clase desde el

objeto asi: loteria1.changeProbability(0.54)el 0.54 lo pongo como ejemplo de lo que se le podría entregar

n).no es correcto cambiar en ese método de clase esa línea ya que los métodos de clase utilizan el prefijo cls para referirse a los atributos de la clase, esto lo hacen por convención y practicidad. Es cierto que el código funcionaria pero no seria lo mas correcto el cambio de cls por el nombre explicito de la clase

o).despues de agregarle el nuevo método, la clase loteria tiene 5 metodos

p).si el apostador1 gana no es totalmente cierto que el apostador2 va a ganar, esto porque hay probabilidades de por medio, por ejemplo, fijémonos en el método playgame. Podemos observar que en este método según lo que salga en a (el valor de a se define por una aleatoriedad de ser 1 o 0) el apostador o pierde o gana. Como cada apostador tiene su probabilidad, no es totalmente cierto que si gana uno ganan los dos ya que la probabilidad en el valor de a puede no ser la misma

q).es importante recordar que las constantes en Python no existen pero si la queremos ver como constante la pondríamos en mayúsculas, luego de esto, pues el método nuevo de change probability no estaría correcto ya que el método intentaría cambiar una constante y aunque lo puede hacer no estaría correcto esto ya que si la definimos como una constante es porque queremos que no sea cambiada asi que aunque esto no salte error, seria algo erroneo y una mala practica . es decir, no se consideraría correcto el método

r).el tipo de retorno de esos dos métodos es de tipo inmutable y es un float

s).la variable self de la línea 18 de la clase loteria esta haciendo referencia a la instancia actual de la clase loteria sobre la que se esta ejecutando el método , no

podría omitirse el uso del self en este caso porque entonces no le estaríamos entregando a la instancia de clase comisionjuegoespectaculos que estamos creando el atributo que ella esta esperando y que es de clase Loteria, el código reventaría

t).el value pasa por valor ya que es un dato de tipo int y como este tipo de dato es un inmutable pasa por valor y el self esta pasando por referencia ya que es una instancia de clase creada por el usuario y estas son de tipo de dato mutable y estos pasan por referencia