Respuestas Taller Python

Fabián Andrés Hurtado Arango

a) En este ejercicio se están definiendo 3 clases:

* Apostador
* ComisionJuegosEspectaculos
* Lotería

b) La línea de código if __name__ == "__main__": sirve para ejecutar el código si y solo si el archivo en el cual se encuentra se está ejecutando como el programa principal.

c) Si retiro la línea de código if __name__ == "__main__":, el resultado será el mismo siempre y cuando el archivo se ejecute directamente desde main.py. Si este es importado a otro archivo puede que genere errores debido a la ejecución de líneas de código no deseadas.

d) Se están creando un total de 2 objetos de la clase Apostador.

e) Los objetos de la clase Apostador que se crearon fueron:

* apostador1
* apostador2

f) En la variable self de la línea 15 de la clase Apostador, al ejecutar el programa principal, se está haciendo referencia a la instancia de la clase Apostador mediante la cual se invocó el método play().

g) Se están creando 2 objetos de la clase Lotería, ambos se crean al invocar al método play() desde una instancia de la clase Apostador.

h) Al modificar la línea 4 de main.py, de la siguiente manera:

apostador1.deposit(300)

lo que sucede es que la instancia apostador1 contará con 300 en wallet, y al invocar el método apostador1.play(400), se imprimirá “Necesitas poner mas dinero en tu wallet” debido a que apostdor1 solo “cuenta” con 300 en wallet y está intentando “jugar” por un valor de 400. Por lo anterior, al imprimir apostador1.wallet, el valor mostrado será 300.

i) Al modificar la línea 10 de main.py, de la siguiente manera:

apostador2.deposit(400)

no suceden cambios con respecto a la lógica del programa. El resultado si puede variar, pero eso se debe a la “aleatoriedad” del programa a la hora de apostar.

j) Los atributos de la clase Lotería que están haciendo referencia a objetos son:

* self.apostador

k) Los atributos de la clase Lotería que están haciendo referencia a tipos primitivos son:

* self.value

l) Para implementar el método de clase changeProbability, se deben escribir las siguientes líneas:

```
class Loteria:
  probability = 0.5

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

m) Para llamar al método changeProbability de la clase Lotería, se puede usar la siguiente línea: Loteria.changeProbability(nueva_probabilidad), donde nueva_probabilidad representa al valor que le queramos dar a la probabilidad.

n) Cambiar la línea cls.probability = nprobability de la clase Lotería por Loteria.probability = nprobability, puede ser correcto dado a que el método changeProbability se refiere específicamente a la clase Lotería.

o) La clase Lotería después de agregar el método changeProbability cuenta con 5 métodos en total:

* __init__
* payMoney
* recieveMoney
* playGame
* changeProbability

p) Debido a la independencia de los eventos, si apostador1 gana, apostador2, no ganaría necesariamente. Estos resultados dependen del “azar”.

q) Si convierto el atributo probability de la clase Lotería a una constante, esto implicaría que este ya no podría ser cambiado, y por lo tanto, el uso del método changeProbability no seria acertado, dado que este modifica el atributo probability, y al este ser una constante, ocurriría un error.

r) Los métodos gain() y commission() devuelven ambos un valor de tipo float.

s) La variable self en la línea 18 de la clase Lotería hace referencia a la instancia sobre la cual se desea aplicar el método commission(). Esta variable no se puede omitir ya que esta es la que permite al método commission() saber sobre cual instancia de la clase Lotería está actuando.

t) En la línea 15 de la clase Apostador, la instancia de la clase Lotería que se está creando recibe los parámetros value y self.
El parámetro value se pasa por valor al ser un tipo de dato inmutable (int).
El parámetro self se pasa por referencia dado a que es un objeto mutable (instancia de la clase Apostador).