Preguntas de análisis – Ejercicio 2 del taller de Python

Luis Esteban Rincon Jaimes

C.C. 1090384822

a) ¿Cuántas clases se están definiendo en este ejercicio?

Se están definiendo 3 clases: Apostador, Loteria y ComisionJuegoEspectaculos

b) ¿Para qué sirve la línea de código if __name__ = "__main__":?

Actúa como el método "main" de java, indicando que desde ese bloque de código se ejecutará el programa. Si se ejecuta desde otro archivo en donde no esté dicha línea de código, lo que esté dentro de esta no se ejecutará.

c) ¿Qué sucede si retiro la línea de la pregunta anterior en nuestro código?, ¿Este sigue corriendo o hay error? Explique en ambos casos.

Sigue corriendo, debido a que, en Python no es necesario tener un "método main" para que se ejecute el programa, ni una clase que contenga dicho bloque de ejecución, a diferencia de Java.

d) ¿Cuántos objetos de la clase Apostador se están creando?

2 objetos.

e) ¿Cuáles objetos de la clase Apostador se están creando?

apostador1 y apostador2.

f) ¿A quién está haciendo referencia la variable self de la línea 15 de la clase Apostador cuando se ejecuta el programa principal*?*

Hace referencia a la instancia que invoque el método play(), en este caso, al apostador1 y luego al apostador2

g) ¿Cuántos objetos de la clase Loteria se están creando?

1 objeto. El objeto loteria.

h) En la línea 4 del main.py cambiar el apostador1.deposit(500) por apostador1.deposit(300). ¿Qué imprimiría el código por parte del apostador1?

Imprimirá:

300

Necesitas poner mas dinero en tu wallet

300

i) En la línea 10 del main.py cambiar el apostador2.deposit(500) por apostador2.deposit(400). ¿Qué imprimiría el código por parte del apostador2?

Hay dos opciones:

Si en la variable aleatoria "a" del método playGame de la clase Loteria se guarda un número menor a 0.5, el código, por parte del apostador2, imprimirá lo siguiente: 400 Has ganado 720 720	Si en la variable aleatoria "a" del método playGame de la clase Loteria se guarda un número mayor o igual a 0.5, el código, por parte del apostador2, imprimirá lo siguiente: 400 Has perdido lo que apostaste 0

j) ¿Cuáles atributos de la clase Lotería están haciendo referencia a objetos?

El atributo "apostador".

k) ¿Cuáles atributos de la clase Lotería están haciendo referencia a tipos primitivos?

Los atributos "value" y "probability"

l) ¿Complete las siguientes líneas para que en la clase Loteria, se implemente el método de clase changeProbability?

```
-  _______________
-  def changeProbability(__, nprobability):
-          _______.probability = nprobability
```

@classmethod

def changeProbability(cls, nprobability):

cls.probability = nprobability

m) ¿Cómo sería la línea de código para llamar el método changeProbability?

Loteria.changeProbability(*argumento*)

n) ¿Es correcto si en el método changeProbability que se creó, cambiar lo siguiente? Explique:

Línea Original

```
- cls.probability = nprobability
```

Línea Nueva

```
- Loteria.probability = nprobability
```

No es correcto, debido a que, al usar la línea nueva se aplicará la modificación del atributo solamente a la clase Loteria, por lo tanto, se perdería la flexibilidad del método de clase changeProbability, en caso de querer ser llamado desde una subclase de Loteria, obstaculizando la reutilización de código en estas.

o) ¿Cuántos métodos tiene la clase Loteria después de agregarle el nuevo método?

Con el nuevo método tendría 5.

p) ¿Si el apostador1 gana el apostador2 también? Explique por qué pasa en caso de ser sí o no

No, porque eso depende de la variable aleatoria del método playGame(), pero dicho método se ejecuta de manera independiente en cada objeto.

q) ¿Qué sucede si decido cambiar el atributo de clase probability a una constante? ¿Se considera correcto el uso del método changeProbability teniendo en cuanta este nuevo cambio?

No se considera correcto, pues si se indica que dicho atributo es constante, esto tiene el objetivo de que su valor no cambie; por ende, se perdería el sentido de indicar lo anterior.

r) ¿Cuál es el tipo de retorno de los métodos gain() y commission() de la clase ComisionJuegoEspectaculos?

Retornan un número de tipo int, siendo el mismo en ambos métodos, ya que el método static gain() recibe los datos necesarios de la clase desde el método commission(), hace los cálculos de la comisión ganada, para posteriormente retornar el resultado al método commission() para retornar dicho valor.

En el caso presentado, donde el atributo de instancia “value” = 400, tanto el método gain() como el método commission() retornarán un valor de 320

s) ¿A quién está haciendo referencia la variable self de la línea 18 de la clase Loteria cuando se ejecuta el programa principal*?* ¿Podría omitirse el uso de la variable self en este caso?

Está haciendo referencia a la instancia actual de la clase Loteria que está ejecutando el método playGame(), es decir, primero a apostador1 y luego a apostador2 (orden en el que se va ejecutando el código), pasando dichas instancias por referencia a la clase ComisionJuegoEspectaculos.

No se puede omitir en ese caso el uso del self, ya que la clase ComisionJuegoEspectaculos necesita acceso a los atributos de las instancias de Loteria para ejecutar sus métodos.

t) ¿En la línea 15 de la clase Apostador vemos como la clase recibe dos parámetros (value, self) especificar cuál de estos pasa por valor y cuál por referencia y por qué?

El parámetro “value” pasa por valor, pues al ser un entero es de tipo inmutable; por otro lado, el parámetro self pasa por referencia, ya que es de tipo mutable al ser un objeto.