TALLER 2-PYTHON

Kevin Andrés Rubiano Gómez cc 1035417435 Grupo1

Ejercicio de codificación

Preguntas de análisis:

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

Se están definiendo 3 clases.

b) ¿Para qué sirve la línea de código if __ name = “ _main_ ”:?

La línea if __name__ == "__main__": asegura que el código dentro de ese bloque solo se ejecute cuando el código sea ejecutado directamente, pero no cuando sea importado para otras parte de código.

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

En primer caso no debería afectar ya que simplemente es una validación que en este caso va a ser verdad, pero todo lo que está después de esa línea está identado dentro del if, por lo que en primera instancia saldrá error por identación, pero quitando este hecho de la identacion, el programa ejecutaría sin problema.

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

Se están creando dos objetos de tipo Apostador.

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

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?

La variable en esta línea de código hace referencia al objeto que está invocando al método play() en este instante en el código principal.

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

- En la línea 4 del main.py cambiar el apostador1.deposit(500) por

apostador1.deposit(300)

Al cambiar apostador1.deposit(500) por apostador1.deposit(300) solo se estará creando un objeto de tipo Loteria en vez de dos como era inicialmente, ya que cuando apostador1 ejecuta el método play(), su wallet será menor a el value por lo que el objeto de loteria no se creara.

h) ¿Qué imprimiría el código por parte del apostador1?

- En la línea 10 del main.py cambiar el apostador2.deposit(500)

por apostador2.deposit(400)

Este cambio no afecte directamente a apostador1, por lo que imprimirá el wallet de apostador1 que originalmente es 500, luego imprimirá si ganó en playGame() o no y por ultimo imprime el valor de wallet luego de pasar por playGame(), todo esto en el contexto de apostador1. En el caso de que el valor como parámetro que recibe deposit() para valor uno es 300, este ejecutara play() e imprimirá “Necesitas poner mas dinero en tu wallet”,

i) ¿Qué imprimiría el código por parte del apostador2?

Imprimirá el wallet de apostador2 que ahora es de 400, luego imprimirá si ganó en playGame() o no (ya que su wallet si le permite crear el objeto tipo Loteria y ejecutar playGame()) y por ultimo imprime el valor de wallet luego de pasar por playGame(), todo esto en el contexto de apostador2.

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

En la clase Lotería, el atributo self.apostador es el único atributo que hace referencia a un objeto, en este caso a un objeto de tipo Apostador.

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

En la clase Lotería serían probability y value.

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

-@classmethod

- def changeProbability(cls, nprobability):

- cls.probability = nprobability

1. @classmethod

2. cls

3.cls

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

Loteria.changeProbability(nprobability)

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
```

En este caso no alteraría nada en la ejecución del código ya que simplemente se está indicando explícitamente que la clase que actúa es Lotería, al igual que con cls, ya que sea crea este método dentro de la clase Loteria.

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

Luego de agregado el nuevo método, la clase Lotería ahora tiene 5 métodos.

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

No, ya que ambos son procesos independientes dentro del método play() y si gana uno, el otro no tiene nada que ver en esto, la probabilidad de que gane apostador1 o apostador2, van por separado.

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?

Al cambiar el atributo de clase probability a una constante, este se volvería inmutable, por lo que no tendría sentido el uso del método changeProbability ya que nunca se podría cambiar.

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

Ambos métodos retornar un valor de tipo float.

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?

self hace referencia a la instancia de la clase Loteria que está ejecutando el método playGame(). No, no se podría omitirse el uso de self en este caso, porque se necesita recibir una instancia de Loteria para funcionar correctamente, y self es la manera en que se accede a esa instancia.

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?

value se pasa por valor (es decir, como una copia del valor original, porque es un tipo inmutable y no se afecta este valor por fuera del método).

self se pasa por referencia (es decir, la referencia a la instancia de Apostador, porque es un objeto mutable y este si se afecta dentro del método).