Preguntas de análisis

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

3 clases

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

Para comprobar si el archivo se está ejecutando como un programa principal y no como un módulo importado en otro script. El código dentro de este bloque solo se ejecutará si el archivo es el programa principal.

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

No sigue corriendo y en su lugar muestra un error debido a identación incorrecta, si arreglo el tema de identación el código corre con normalidad sin verificar si el archivo es el programa principal, ya que sin importar si la condición eliminada se verifique o no el archivo se mantiene siendo el programa principal y quién permite la ejecución

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

2

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 al objeto apostador1 o apostador2 cuando se ejecuta el programa, dependiendo de cuál objeto llame al método.

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)

R/Se crean objetos de la clase Loteria cada vez que se llama al método play() en la clase Apostador. En este caso, se crean dos objetos Loteria, uno para cada jugador

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)

R/ 300

Necesitas poner mas dinero en tu wallet

300

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

R/ 400

Has perdido lo que apostaste

0

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

self.apostador, este atributo hace referencia a un objeto de la clase Apostador. Es un objeto porque se le asigna una instancia de la clase Apostador al crear el objeto Loteria

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

self.value, este atributo es de tipo primitivo int o float, dependiendo del valor que se pase al instanciar la clase Loteria, y self.probability, el cual es de tipo float y tiene un valor predeterminado de 0.5.

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

- _____

- def changeProbability(__, nprobability):

- _ .probability = nprobability

Para implementar el método de clase changeProbability se necesita usar el decorador @classmethod y hacer referencia al atributo de clase probability usando cls en lugar de self. La implementación sería:

@classmethod

def changeProbability(cls, nprobability):

cls.probability = nprobability

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

Loteria.changeProbability(0.7)

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

R/Ambas líneas son correctas. En un método de clase, se puede usar cls.probability o Loteria.probability, ya que ambos hacen referencia al atributo de clase probability. Sin embargo, es más recomendado usar cls dentro de los métodos de clase porque cls se usa para referirse a la clase desde el interior de un método de clase, mientras que usar Loteria.probability es explícito, pero menos flexible

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

5

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

No, el apostador 1 y el apostador 2 no ganan simultáneamente porque el juego se resuelve individualmente para cada apostador. Cuando un apostador juega, su juego se desarrolla de forma independiente, por lo que, si el apostador 1 gana, eso no afecta al apostador 2, ya que sus apuestas son desarrolladas como juegos separados. Ambos pueden ganar o perder, pero el uno no depende del otro

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?

Si probability se convierte en una constante, lo más adecuado sería no permitir su modificación mediante el método changeProbability. Las constantes en Python se definen con letras mayúsculas y se espera que no cambien durante la ejecución del programa. Por lo tanto, sería más apropiado eliminar el método changeProbability o asegurar que la probabilidad sea siempre la misma.

Si se mantiene el método changeProbability y se trata de modificar una constante, no sería correcto, ya que las constantes, por definición, no deben cambiar

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

El método gain() tiene como tipo de retorno float, ya que el cálculo de gain implica una operación matemática que devuelve un valor numérico, mientras que el método commission() tiene como tipo de retorno float, ya que también devuelve un valor numérico relacionado con la comisión calculada por gain().

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?

Hace referencia a la instancia actual de la clase Loteria. Esta instancia es el objeto de tipo Loteria que se está creando y ejecutando en el programa principal, así que, no se podría omitir el uso de self, ya que se refiere a la instancia actual de la clase Loteria y se necesita acceder a sus atributos

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?

Self siempre hace referencia a la instancia actual de la clase Apostador, y siempre se pasa por referencia.

Value es un valor que se pasa al método, o sea, el monto que el apostador quiere apostar, así que este valor se pasa por valor.

Esto significa que self hace referencia al objeto actual de la clase Apostador, mientras que value es simplemente un dato numérico que se pasa como argumento y no se modifica fuera del método