PROGRAMACIÓN ORIENTADA A OBJETOS - UNALMED 2024 -2

Preguntas De Análisis

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

Se están definiendo tres clases en el ejercicio.

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

Esta línea de código se utiliza para determinar si el script actual se está ejecutando como programa principal o si se está importando como un módulo en otro script.

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

Esta línea es útil para evitar que el código se ejecute accidentalmente al importarlo en otros scripts, es decir, con esta línea el código solo se ejecuta si se corre el archivo directamente, pero si se retira dicha línea, el código se ejecuta cada vez que se importe ese archivo en otro script, lo que puede causar comportamientos no deseados.

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

Se están creando dos objetos de la clase “Apostador”.

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

Se están creando los objetos “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 self de la línea 15 hace referencia al objeto de la clase “Apostador” que llame al método “play”. En este caso, el primer objeto que utiliza el método “play” es “apostador1”, por lo que self haría referencia a “apostador 1”. Luego el objeto “apostador2” llama al método “play”, haciendo que el self se refiera al objeto “apostador2”

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)

Se están creando dos objetos de la clase “Loteria”, uno para cada objeto de la clase “Apostador”.

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)

Al cambiar el deposito del “apostador1” por 300 en la línea 3, el código nos arrojara:

300

Necesitas poner mas dinero en tu wallet

300

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

Al cambiar el deposito del “apostador2” por 300 en la línea 10, el “apostador2” seguirá apostando su deposito, arrojándonos en caso de ganar:

400

Has ganado 720.0

720.0

Y en caso de perder:

400

Has perdido lo que apostaste

0

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

El atributo de la clase “Loteria” que están haciendo referencia a objetos es: “apostador”

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

Los atributos de la clase “Loteria” que están haciendo referencia a tipos primitivos son: “value” y “probability”.

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

@staticmethod

def changeProbability(*cls*, nprobability):

cls.probability = nprobability

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

Loteria.changeProbability(-La nueva probabilidad-):

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
```

Si es correcto, ya que el “cls.probability” esta haciendo referencia al atributo de clase “probability”, el cual también puede ser llamado por la misma clase, como se muestra en la nueva línea “Loteria.probability”, sin alterar el funcionamiento del codigo.

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

Con el nuevo método la clase “Loteria” tiene cinco métodos.

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

Si el “apostador”1 gana no necesariamente gana el “apostador2”, esto se debe a que al momento de realizar el método “play” que es el que determina quien gana, se crean dos objetos “Loteria” diferentes, uno para cada apostador, por lo que al tener diferentes espacios de memoria y realizar la selección del ganador de manera random, como lo indica el método “play”, los apostadores no se influencian entre ellos, por lo que sus resultados no se influyen entre sí.

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 el uso del método “changeProbability”, si se cambia el atributo de clase “probability” a una constante, ya que el propio termino de constante no indica que el valor de esta no debe variar, y al aplicar dicho método se está cambiando. Por lo que a pesar de que el código lo puede correr, es algo erróneo de realizar.

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

El método “gain” retorna un float y el método “commission” retorna un 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?

El self de la linea 18 de la clase “Loteria” hace referencia a los atributos del objeto de la clase “Loteria” que realiza el método “playGame”. Este self es indispensable para el código porque nos permite acceder a los atributos del objeto de la clase “Loteria”, los cuales son necesarios al momento de crear el objeto “commi” de la clase “ComisionJuegoEspectaculos”.

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, ya que es un tipo inmutable y el parámetro “self” pasa por referencia, ya que hace referencia a una instancia de la clase ”Apostador”.