PROGRAMACIÓN ORIENTADA A OBJETOS - UNALMED 2024 -2

Preguntas de análisis

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

b) ¿Para qué sirve la línea de código **`if____name___= "__main___":?`**

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

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

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

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

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)

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

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

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

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

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

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

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
```

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

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

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?

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

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?

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?

SOLUCION

a) En este ejercicio se definieron 3 clases
   a. Apostador
   b. ComisionJuegoEspectaculos
   c. Loteria

b) Esta línea de código es útil para controlar la ejecución del código que se encuentra dentro de la condicional, ya que esta solo se ejecutará si se ejecuta directamente el código y no lo hará si este es importado a otro script es decir que el código dentro de la condicional solo se podrá usar si es el script a que este pertenece el que se ejecuta y no si se ejecuta por otro medio como la importación

c)En principio si se ejecuta el programa al que pertenecía la condicional no debería haber cambios en el comportamiento del código, la cosa cambia si este es importado a otro código en el cual la línea de código fue puesta específicamente para evitar problemas con los procesos que se desarrollan dentro del código y no interfieran con el resultado esperado al importar el código.

d)En el código se están creando 2 objetos de la clase apostador

e) Se crean en la línea 3 del código main con el apuntador apostador1 y en la línea 9 del código main con el apuntador apostador2

f) La variable self hace referencia al objeto sobre el cual se ejecuta el método en este caso si lo analizamos en la línea 15 de la clase apostador en esta se está ejecutando el método play el cual crea un objeto de clase lotería usando en este caso en la línea 6 del código main al

objeto apuntado por apostador1 y en la línea 12 al objeto apuntado por apostador2

g) Los objetos de clase lotería se crean al realizarse el método de play este se realiza dos veces en el código main creando dos objetos de clase lotería

sí en la línea 4 cambiamos el valor de 500 a 300 entonces no se creará un objeto de la clase lotería y solo aparecerá el mensaje "Necesitas poner más dinero en tu wallet" al tener menos dinero en su wallet que el apostado

h) Por parte del objeto apuntado por apostador1 existen dos posibles salidas al ejecutarse su código, la primera es que el numero generado sea menor al 0.5 y este gane su apuesta aumentando el dinero de su wallet el otro posible resultado es que el resultado sea mayor a 0.5 y este pierda los 500 apostados

Si cambiamos esta línea de código a pesar de tener menos dinero en su wallet este sigue siendo igual o mayor al dinero apostado entonces el código se correrá con normalidad creando el objeto de tipo lotería y continuando el juego.

i)El resultado sería exactamente el mismo que con el objeto apuntado por apostador1.

j) El atributo apostador es el que hace referencia a un objeto en este caso al objeto de tipo apostador llamado con el método play

k) seria value que hace referencia un número y probability que hace referencia a la probabilidad de ganar de la persona siendo un numero de tipo float.

l) @classmethod

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

m) la línea de código para llamar al método changeProbability seria

```
loteria = Loteria(value, self)
loteria.changeProbability(Loteria, npropability)
```

n) la ejecución del código seguiría funcionando, pero no de una forma optima y simple.

o) Con este tendría 4 metodos aparte del que funciona para crear los objetos, estos serían, payMoney, recieveMoney, playGame, change Probability.

p) Si el apostador1 gana no necesariamente ganaría el apostador2 esto debido a que al ejecutarse la línea de código “apuntador.play(cantidad de dinero)” se genera un nuevo numero aleatorio en el método playGame, y de este numero aleatorio es del que depende si el apostador gana o pierde, si al apostador1 le sale un numero ganador puede que al apostador2 el cual tendrá un numero aleatorio diferente no gane, esto debido a que el número de cada juego se guarda en la variable a la cual al finalizar de realizarse el proceso se libera del valor anterior o elimina de cierta forma hasta que se vuelva a ejecutar el método y reciba un nuevo número aleatorio.

q) Si se cambia el atributo probability a una constante no seria coherente tener un método que cambia dicho valor y sería un error lógico.

r) El método gain de vuelve un numero tipo float resultante a la operación del porcentaje de comisión y el valor de la loteria mientras que el método comisión devuelve un valor número tipo float que le es entregado por el método gain.

s) Hace referencia a la variable tipo lotería que le es entregada al método playGame y en el caso de la línea 18 es necesario entregarle este self ya que no hay mas cosas que le indiquen sobre que objeto se creara el nuevo objeto de tipo ComisionJuegoEspectaculos.

t) value pasa por valor y self por referencia.