Preguntas de análisis

Andrea Merino Mesa
T.I. 1034991011

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

Se definen **tres** clases, Lotería, ComisionJuegoEspectaculos y Apostador.

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

Se puede utilizar para **dividir el código**, indicando hasta qué parte van **las clases** y a partir de cuál empieza la **creación y operación con los objeto**s. También sirve para asegurarse que el código se ejecute solo si se inicia desde el mismo módulo donde está la condición (main.py). De lo contrario, evita que se ejecute, haciendo que se tenga un mayor control sobre el flujo de ejecución.

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

Si **sólo quitamos la línea de código** y no hacemos nada más, **genera error** gracias a que Python define los bloques de código del programa según la **identación** y al borrar esa línea, queda la parte siguiente del código identada sin necesidad (no hay condicionales, ciclos, funciones...).

Si por el contrario **eliminamos la línea de código y corregimos la identación**, debería seguir corriendo.

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

Desde el archivo main se crean **dos** objetos de **clase Apostador.**

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

apostador1 = id 1, name "Juan", phone_number 302, email "j@gmail.com"
apostador2 = id 2, name " Ricardo ", phone_number 548, email "r@gmail.com"

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

Se refiere **al objeto de tipo Apostador que llama al método**, esto para que desde la clase Lotería se pueda acceder a los atributos de instancia del objeto de clase Apostador.

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

Se crean **dos objetos de clase Loteria**, el primero cuando apostador1 llama al método play() y como tiene suficiente dinero en su billetera puede crear una instancia de Lotería, lo mismo pasa con el segundo, pero quien lo llama es apostador2.

- En la línea **4** del **main.py** cambiar el apostador1.deposit(500) por apostador1.deposit(300)

En ese caso **solo se crearía un objeto de clase Lotería** (desde apostador2), pues **apostador1 no tendría suficiente dinero** en la billetera para apostar el valor deseado (300<400) y por tanto no se crea la instancia.

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

En circunstancias normales:
En primer lugar, imprimiría el saldo de cuenta después de hacer la transferencia (500), luego cuando llama al método play(), depende del resultado de este se imprimiría un mensaje diferente ("Has ganado "+ str(total) o "Has perdido lo que apostaste") y por último imprimiría el saldo actual, es decir, lo que queda en su cuenta después de apostar (si gana sería 820.0 y si pierde, 100).

Según la condición de la pregunta anterior:
En primer lugar, **imprimiría el saldo de cuenta después de hacer la transferencia** (300), luego cuando llama al método play() imprimiría **"Necesitas poner más dinero en tu wallet"** pues quiere apostar más dinero del que tiene y, por último, **imprimiría el saldo actual de su cuenta que sería el mismo** ya que no pudo hacer la apuesta (300).

- En la línea 10 del main.py cambiar el apostador2.deposit(500) por apostador2.deposit(400)

Si hacemos este cambio, de igual manera el código por parte de apostador1 seguiría imprimiendo lo mismo pues el valor que se cambia pertenece es a apostador2.

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

En circunstancias normales:
Al igual que apostador1, imprimiría el saldo de cuenta después de hacer la transferencia (500), el resultado del método play() ("Has ganado "+ str(total) o "Has perdido lo que apostaste") y el saldo actual, es decir, lo que queda en su cuenta después de apostar (si gana sería 820.0 y si pierde, 100).

Según la condición de la pregunta anterior:
Al igual que apostador1, **imprimiría el saldo de cuenta después de hacer la transferencia** (500), **el resultado del método play()** ("Has ganado "+ str(total) o "Has perdido lo que apostaste") y el saldo actual, es decir, lo que queda en su cuenta después de apostar (**si gana 720.0 y si pierde 0**).

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

El atributo de la clase lotería que está haciendo referencia a un objeto es **apostador** que viene de la **clase Apostador.**

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

Los atributos de la clase Lotería que hacen referencia a tipos primitivos son **Probability** (que tiene un **float** de valor 0.5) y **value** (**que recibe números**, por lo que vemos en el ejemplo de tipo entero).

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

- @classmethod
- def changeProbability(cls, nprobability):
- cls.probability = nprobability

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

newprobability=Loteria.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

Si, es correcto hacer ese cambio porque **probability es un atributo de clase y por tanto debería poder accederse por medio de la misma clase de la que hace parte (Loteria).** Además, como se modifica desde la clase y no desde un objeto, evita que a partir de esta se cree un atributo de instancia.

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

Después de agregarle el nuevo método a Lotería, esta queda con **cinco métodos**, __init__() (método constructor), payMoney(), receiveMoney(), playGame() y el agregado recientemente, changeProbability().

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

No, porque la **probabilidad es la misma para ambos** y el que **ganen o pierdan lo define un número al azar** que se da con la librería randint y que podría dar un resultado igual o diferente para ambos.

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?

Como en Python no existen las constantes, es decir, se indican con mayúsculas para que el desarrollador lo identifique, pero no afecta el funcionamiento del programa, si cambiamos el atributo a una constante Python lo seguirá interpretando de la misma manera. Sin embargo, **no se considera correcto** el uso del método changeProbability() en este caso pues, **aunque para Python sea lo mismo, para la lógica del código no**, pues **una constante supone un valor que no puede ser cambiado.**

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

El método **gain() retorna un valor de tipo float** que representa la **ganancia** dada por la cantidad de dinero invertida menos el porcentaje que se queda la comisión de juegos para sí misma. De igual manera el método **commission() retorna este mismo valor** puesto que esta **llama al método gain(), guarda el valor que este devuelve en la variable commission y retorna dicha variable.**

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 al objeto** de clase Lotería que **crea el objeto Apostador cuando llama al método play()**. **No** se puede omitir el uso de la variable self **porque la clase ComisionJuegoEspectaculo** que es a la que se le pasa el parámetro cuando se llama en esa línea, **no podría definir el atributo lotería pues no se le pasó dicho parámetro** y por tanto no permitiría ejecutar el método comission().

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** es el que **pasa por valor** pues es de **tipo entero** y es el número que se le pasa al método cuando este es llamado. El parámetro **self** por otro lado **pasa por referencia** ya que hace **referencia al objeto que llama al método play()**, que en este caso es de **tipo Apostador**.