a) 3 clases se definen: Loteria,ComisionJuegoEspectaculos y Apostador.

b) Sirve para ejecutar la función main solo dentro del propio modulo.

c) Si se retira, lo más probable es que se ejecute cierta parte del código que en realidad no queramos en otro modulo y por tanto estaríamos perdiendo el control de la app.

d) Se están creando 2 objetos de la clase Apostador.

e) apostador1 y apostador2

f) Se refiere a la instancia de clase Apostador (objeto y sus atributos) que llama al método,

si fuera apostador1.play se refiere a la instancia apostador1

g) Se crea 1 objeto de clase Loteria con atributos valor y apostador.

h) imprimiría 300 y luego "Necesitas poner mas dinero en tu wallet". puesto que tiene 300 de saldo y quiere apostar 400, cosa que no es lógica.

i)imprimiría 400 y luego se ejecutaría el método playGame.

j) El atributo apostador hace referencia a un objeto de tipo Apostador.

k) El atributo value hace referencia a tipo primitivo.

l) - @classmethod

- def changeProbability(cls,nprobability):

- cls.probability = nprobability

m) Si lo llamo desde la clase:

-Loteria.changeProbability(cambioDeProbabilidad)

n) Si es correcto ya que hacen referencia a lo mismo. Al usar cls estamos haciendo referencia directamente a la clase que llama al método en este caso Loteria, y al mismo tiempo usar el nombre de la clase (en este caso Loteria), estaríamos haciendo referencia a la clase, pero ya directamente.

o) 5 métodos (__init__,payMoney,recieveMoney,playGame,changeProbability)

p) No necesariamente, aunque pueda que pase no significa que si uno gana el otro también lo hará, ya que cada apostador tiene 50% de chances de ganar y como el resultado es aleatorio (puede que salga el 0 o el 1) entonces no.

q) A pesar de que cambiemos el atributo de clase probability a una constante, el método changeProbability seguirá funcionando ya que en python no hay

constantes, pero ya no tendría utilidad puesto que queremos que probability no cambie y por tanto no sería correcto.

r) Tanto el método gain() como el método comission() retornan float.

s) La variable self en este caso hace referencia a un objeto de tipo Loteria, y no se puede omitir puesto que en los métodos siempre debe ir al inicio ya que este es el que permite el acceso a sus atributos y/o métodos por tanto quitarlo no nos permitiría lo anterior y por tanto generaría errores en el código.

t) value pasa por valor y self por referencia: self hace referencia al objeto de tipo Apostador ya sea apostador1 o apostador2 en este caso. value se pasa por valor al método Loteria y crea una copia y por tanto cualquier cambio/modificación que se haga en este no afectará al valor original en Apostador.