Programación Orientada a Objetos. Taller 5 Python

David Fernando Cerón Quintero

1. Se podría crear un método llamado "ruido" en la clase Pajaro, y que reciba como parámetros ciertos argumentos y los imprima (En el ejemplo sería que reciba un string, y lo imprima).

2. Suponiendo que es una clase hija de Animal, hereda el constructor de esta clase (Es decir, que crear una clase Pez necesitaría nombre, edad y raza), y podría invocar todos los métodos que tenga la clase animal (Tanto los de la propia clase y los que hereda de la clase SerVivo).

3. Los atributos de SerVivo serían opacados por los atributos de la clase Animal, haciendo que estos últimos sean los únicos. Para cambiar esto, se tendría que las asignaciones de _nombre y _edad por el código super.__init__(nombre, edad). Aunque, Python trataría a estos "dos" atributos como el mismo, ya que lo redefine y lo trata como si fuese el único existente.

4. Se podría retornar el valor de super().getNombre() y ejecutar el método super().setNombre(nombre) en los métodos getNombre y setNombre de Animal respectivamente.

5. Sí, es heredado, y los oculta, ya que siguen existiendo (Ocupando un espacio de memoria), pero solo reconocen al totalCreados más próximo que existe de la clase que hereda.

6. Sí, porque son métodos de clase que utilizan la misma firma.

7. El __init__, aduenarAnimal y getTotalCreados, los cuales son heredados de la clase Persona. También setNombre, getNombre, setEdad y getEdad de la clase SerVivo (Que es la clase padre de Persona). También hereda todos los métodos mágicos asociados a Persona.

8. En esencia, debería pasársele un objeto de clase Animal o una clase que herede esta misma, esto siguiendo las buenas prácticas. Sin embargo, cualquier objeto que tenga el método ruido() incorporado va a hacer que este método se ejecute de manera adecuada. Por este motivo, objetos de la clase SerVivo no pueden ser puestos en los parámetros de aduenarAnimal.

9. Existirían 2 maneras, como la variable totalCreados lelva esta cuenta cada vez que se crea un objeto de clase SerVivo, podría cambiarse el código de la clase Animal para que así llame el método __init__ de su clase padre a través de super, y así seguir llevando la cuenta de todos los objetos que son de la clase SerVivo incluyendo los que heredan la misma. Otra forma es contar sumar el valor totalCreados de SerVivo y totalCreados de Animal para llevar una cuenta de todos los objetos (Tanto los de la clase original como los que la heredan).

10. Se redifiniría para cada objeto que sea de la clase Perro, y pasaría a ser solamente la descrita en el enunciado.