Taller #5 python

1. Se debe redefinir el método ruido() en la clase Pajaro, pues este hereda el método de la clase Animal en cual simplemente pasa al ser llamado, se podría añadiar a la clase pájao las siguientes líneas:
   ```
   def ruido():
       return “cantar y silbar”
   ```
2. No pasaría nada puesto que no se especifica si sería una subclase de otra ya existente, en de caso que lo sea, por ejemplo de Animal, heredaría el atributo de clase _totalCreados, y el método de clase que obtiene dicho atributo, también heredaría los atributos de instancia propios de la clase Animal y el inicializador de esta, además se heredarían los métodos get y set de los respectivos atributos, finalmente se heredarían los métodos caminar, correr y ruido, sin embargo ninguno de estos haría algo, pues al ser llamados solo pasarían.
3. Al redefinirse en la clase Animal, los atributos nombre y edad reemplazan a los atributos heredados de la clase SerVivo, pero no serían el mismo atributo, por lo que los atributos de la clase SerVivo ya no son accesibles por la clase Animal, la forma de que estos atributos continúen siendo los misms sería borrar las líneas 7 y 8 de la clase animal y reemplazándolas por un llamado al inicializador de la clase superior, pasándole los argumentos nombre y edad, que fueron pasados al inicializar la clase Animal, de la siguiente manera:
   ```
   def __init__(self, nombre, edad, raza):
       super().__init__(nombre, edad)
       self._raza = raza
       Animal.totalCreados += 1
   ```
4. Se tendría que cambiar self por super().
5. Si se hereda este atributo, pero al ser redefinido en las sub-clases, se oculta el valor del atributo de la clase padre.
6. No, pues este métedo todavía puede ser llamado desdelas sub-clases con el método super.
7. El inicializador __init__, aduenarAnimal, y getTotalCreados, el resto de métodos de get y set se heredan desde la clase SerVivo
8. Se le podrían pasar objetos de clase Animal, Peero, Pajaro o Gato, de cualquier otro modo se generaría un error, pues en el método aduenarAnimal se llama al método ruido de la instancia que posee el argumento pasado, por lo que solo funcionaría bien con objetos de un clase donde este método esté definido, y por esta razón no se podría pasar un objeto de clase SerVivo si dicho objeto no posee la clase Animal o alguna de sus sub-clases.
9. Se podría pensar que llamando al método getTotalCreados de la clase SerVivo, sin embargo este método llama al atributo de clase _totalCreados, el cual no se actualiza cuando se crea un objeto de la clase Animal o alguna de sus sub-clases, lo que podría hacerse es crear un nuevo método de clase, el cual, sume el valor de los atributos _totalCreados de las clases Animal y SerVivo respectivamente, pues los objetos de clase SerVivo y Persona se cuentan en el atributo de SerVivo, mientras los de la clase Animal, Perro, Gato y Pajaro se cuentan en el de la clase Animal, este podría hacerse en la clase Animal de la siguiente manera:

```
    @classmethod
    def getTotalVivos(cls):
        return cls._totalCreados + super().getTotalCreados()
```

10. Se sobreescribe, pues en python no existe como tal una sobrecarga de métodos.