Ejercicio 2 taller 5 Python

1. Se debe sobrescribir el método ruido () en la clase Pajaro.
   def ruido(self):
       print(“cantar y silbar”)
2. Esta nueva clase vacía (que hereda de Animal) tendrá como inicializador el de su clase padre (Animal), por lo tanto, si se crea un objeto de tipo Pez, se tendrían que pasar los parámetros que se le pasan al inicializador de la clase Animal, ya que esta nueva clase Pez lo estaría heredando (puesto el __init__ también es un método, entonces se puede heredar). Además, esta clase heredaría tanto métodos y atributos de la clase Animal (siempre que sean aptos para heredarse).
3. Los atributos nombre y edad de la clase SerVivo son redefinidos en la clase Animal cuando se crea un objeto de tipo Animal. Para modificar el constructor de modo que estos atributos se compartan correctamente, se debe redefinir el constructor de la clase Animal de la siguiente forma:
   ```
   def __init__(self, nombre, edad, raza):
     super().__init__(nombre, edad)
     self._raza = raza
     Animal._totalcreados += 1
   ```
4. Sí, de la siguiente manera:

   ```
   def setNombre(self, nombre):
     self._nombre = nombre
     super().setNombre(nombre)

   def getNombre(self):
     return super().getNombre()
   ```
   Al usar super(), se invocan los métodos setNombre y getNombre de la clase padre SerVivo, lo cual es necesario porque el atributo nombre es privado.

5. Este atributo es heredado, pero al redefinirse en las subclases (Persona, Animal, Gato, Pajaro, Perro), se oculta el atributo de la clase padre. Esto causa que cada subclase mantenga su propio contador independiente.
6. Sí, el método getTotalCreados() se define con el mismo nombre en cada subclase de SerVivo (Persona, Animal, Gato, Pajaro, Perro), sobrescribiendo el método de la clase padre. Esto hace que cada subclase retorne solo su propio contador.

7. Cualquier clase que herede de Persona obtendrá todos sus métodos, incluyendo __init__(), aduenarAnimal(), getTotalCreados(), y los métodos heredados de SerVivo: setNombre(), getNombre(), setEdad(), getEdad()
8. El método aduenarAnimal() puede recibir cualquier objeto que tenga el método ruido(), porque en la línea 13 intenta llamarlo explícitamente. Si el objeto no tiene este método, se generará un error. Los objetos que cuentan con este método y pueden pasarse son aquellos de la clase Animal y sus subclases (Perro, Gato, Pajaro). No se puede pasar un objeto SerVivo porque no tiene el método ruido(), lo que causaría un error en la línea 13.
9. Debe haber solo un atributo de clase llamado _totalCreados en la clase SerVivo, para que este atributo sea único y lleve el contador de todos los objetos creados. Se deben eliminar los atributos _totalCreados en las subclases de SerVivo y modificar los métodos getTotalCreados() en cada una de ellas así:

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

   Esto permitirá imprimir el total de seres vivos creados:

   ```
   print(SerVivo.getTotalCreados())
   ```

   O desde cualquier subclase de SerVivo:

   ```
   print(Animal.getTotalCreados())
   print(Persona.getTotalCreados())
   print(Gato.getTotalCreados())
   ```

10. El método getRaza() se sobrescribe. En Python, no existe la sobrecarga de métodos porque el lenguaje no diferencia el tipo de dato de los parámetros. Por lo tanto, si hay dos métodos con el mismo nombre en una clase, solo se mantendrá la última definición. En consecuencia, el nuevo método getRaza() en Perro sobrescribe el método heredado de Animal y requiere un parámetro al ser invocado.