Programacion Orientada a Objetos

Santiago Abelardo Salcedo Rodriguez

Taller 5 parte 2 python

Preguntas de Analisis.

1.Si deseo modificar el mensaje del método ruido al crear un objeto Pajaro sin alterar la clase Animal ¿Qué debo agregarle al código? (Por ejemplo, al llamar el método ruido imprima, cantar y silbar).

R= Dentro de la clase Pajaro "sobreescribes" aunque en Python no existe como tal la sobreescritura el método ruido agregarías algo tal que asi:

```
def ruido(self):
    print("cantar y silbar")
```

2. Si se crea una nueva clase Pez, y no se definen nuevos métodos, constructor y atributos. ¿Qué constructor tendrá esta clase, qué argumentos recibe? ¿Qué otros métodos y atributos tendrán estos mismos?

R= Todo lo que hereda de la clase object, El constructor por defecto aunque primero antes el inicializador, no recibiría ningún método, y tendría otros métodos como __del__() , __str__(), __init__(), y algunos otros métodos mas que podemos observar cuando utilizamos print(dir(pez))

3. ¿Qué ocurre con el atributo nombre y edad de la clase SerVivo, al momento de definirse en la Clase Animal? ¿Cómo cambiaría el código del constructor para que estos atributos sean el mismo?

R= Cuando la clase Animal hereda de SerVivo, si la clase SerVivo ya tiene los atributos nombre y edad definidos, la clase Animal en su forma actual vuelve a definirlos en la propia clase Animal, creando nuevos atributos independientes y ocultando los otros de los de SerVivo.

Cambiaria en la clase animal el inicializador asi:

```
def __init__(self, nombre, edad, raza):
    super().__init__(nombre, edad)
    self._raza = raza
    Animal._totalCreados += 1
```

4. En la clase Animal se sobrescribieron los métodos setNombre y getNombre, ¿Como modificaría estos métodos para que su funcionamiento no oculte algún valor de la clase padre? ¿Podría plantearse esta solución usando super()?

R= para esto precisamente lo mejor es utilizar los métodos ya creados de la clase padre mediante el super(), bastaría con cambiar la línea dentro de los set y get asi: super().setNombre(nombre), igual con el get, asi no ocultamos ningún valor de la clase padre

5. El atributo totalCreados de la clase SerVivo ¿es heredado por las demás clases? ¿En este caso ocurre ocultación de los atributos al definirlo de nuevo en las clases hijas?

R= No, los atributos de clases son propios de la clase, no son heredados, no ocurre ocultación en este caso, simplemente la clase hija lo redeclara nuevamente

6. ¿Los métodos getTotalCreados sobrescriben al metodo de su padre?

R= Si, quedan sobreescritos, no como en java

7. ¿Qué métodos hereda una clase que hereda de la clase Persona?

R= __init(): (en caso que no lo haya declarado en su clase), aduenarAnimal, getTotalCreados, setNombre,getNombre, setEdad, getEdad, getTotalCreados, __str__(), __del__().

8. ¿Qué tipo de objetos podrían pasársele al método aduenarAnimal de la clase Persona?¿Se le puede pasar un objeto serVivo?

R= técnicamente podría entrar cualquier tipo de objeto pues Python no discrimina los tipos, pero que entren y siga el curso normal del código serian objetos de tipo perro y gato,si se le pasa un objeto un objeto de serVivo al momento de ejecutar la línea x.ruido) habría un error por que este no lo tiene definido

9. ¿Cómo podría obtener la cantidad de seres vivos que se creen (objetos creados)?

R= no serviría con utilizar el método getTotalCrteados de la clase seres vivos y persona, pues persona llama al constructor de serVivo, mas no las de sus subclases animal, perro y gato, tocaria también acceder al _totalCreados de Animal.

10. Si se define el siguiente método en la clase Perro:

```
def getRaza(self, tipo):
    return self._raza + " " + tipo
```

¿Qué ocurre con el método getRaza de la clase Animal? ¿Este método se sobreescribe o se sobrecarga

R= Queda ocultado, En Python no existe la sobrecarga de métodos pues para Python la firma del método es únicamente su Nombre, entonces no tiene sentido preguntarse por sobrecarga, aparte que Python es un lenguaje únicamente interpretado y lee el ultimo método que se haya definido en su ejecución (con algunas exepciones)