Desarrollo taller de Python:

1.Debo agregarle al código lo siguiente:

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

2.Si se crea una nueva clase Pez sin nuevos métodos, constructor ni atributos, esta clase heredará todo de la clase Animal, incluyendo el constructor, los métodos y los atributos. El constructor será el de la clase Animal porque no se define uno nuevo en Pez. Es decir, Pez no tendrá atributos ni métodos adicionales.

El constructor de pez será:

```
def __init__(self, nombre, edad, raza):
    super().__init__(nombre, edad, raza)
```

Además, los atributos heredados serán nombre, edad y raza, y los métodos disponibles serán los de Animal (setNombre, getNombre, setEdad, getEdad, caminar, correr, ruido).

3 Cuando se define la clase Animal (que hereda de SerVivo), los atributos nombre y edad ya están definidos en la clase SerVivo, por lo que al crear un objeto de tipo Animal, estos atributos se heredan automáticamente. El constructor de Animal simplemente invoca el constructor de la clase padre SerVivo con super()

4 Para evitar que los métodos setNombre y getNombre de la clase Animal oculten los valores de la clase SerVivo, se puede utilizar super() para hacer referencia a los métodos de la clase padre:

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

def getNombre(self):
    return super().getNombre()
```

5. El atributo totalCreados es heredado por todas las clases hijas (como Animal, Gato, Pajaro, etc.). Sin embargo, se oculta en cada una de las clases hijas porque se redefine. Cada clase hija tiene su propia versión de totalCreados, por lo que no compartirá el valor con la clase padre.

Si se desea que todas las clases hijas compartan este contador, se debe asegura que solo se defina en la clase padre (SerVivo) y no en las clases hijas. Entonces, las clases hijas usarían el contador de la clase SerVivo.

6. Sí, el método getTotalCreados sobrescribe al método de su clase padre. Cada clase hija tiene su propia implementación de este método, que incrementa el contador de esa clase particular.

Si no se desea sobrescribirlo, se podría utilizar super() en lugar de redefinirlo en cada clase hija.

7 La clase que herede de Persona heredará los métodos y atributos de Persona, como:

El constructor __init__(self, nombre, edad)

El atributo _animalAcargo

El método aduenarAnimal(self, x)

El método getTotalCreados (aunque este puede ser sobrescrito)

No heredará los métodos de las clases Animal o SerVivo a menos que se haga explícitamente una llamada a super() desde la clase hija.

8. El método aduenarAnimal de Persona espera recibir un objeto de tipo Animal (o sus clases derivadas como Gato, Pajaro, Perro). No se puede pasar un objeto de tipo SerVivo directamente, porque SerVivo es una clase general, mientras que el método está diseñado para interactuar con instancias de Animal.

9. Para obtener la cantidad de objetos creados de cualquier clase que herede de SerVivo, se puede acceder al atributo totalCreados de SerVivo:

print(SerVivo.getTotalCreados())

10. Si se define el siguiente método en Perro:

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

Esto sobrecarga el método getRaza de la clase Animal. La razón es que la firma del método cambia (se le agrega un nuevo parámetro tipo), lo cual es una sobrecarga. La sobrecarga ocurre cuando un método de la misma clase tiene el mismo nombre pero diferentes parámetros. Sin embargo, no es una sobrescritura porque los parámetros de getRaza en Animal y Perro son diferentes.