Exercise: Polymorphism

Problems for exercise and homework for the [Python OOP Course @SoftUni](). Submit your solutions in the SoftUni judge system at https://judge.softuni.bg/Contests/1943

1. Vehicle

Create an **abstract class called `Vehicle`** that should have abstract methods **`drive`** and **`refuel`**. Create **2 vehicles** that **inherit the `Vehicle`** class (a **`Car`** and a **`Truck`**) and simulates **driving** and **refueling** them. **`Car`** and **`Truck`** both have **`fuel_quantity`**, **`fuel_consumption`** in liters per **km** and can be driven a given **distance**: **`drive(distance)`** and refueled with a given amount of fuel: **`refuel(fuel)`**. It is summer, so both vehicles use air conditioners and their fuel consumption per **km** when **driving** is **increased by 0.9 liters** for the **car** and **with 1.6 liters** for the **truck**. Also, the **`Truck`** has a tiny hole in its tank and when it's refueled it keeps only **95% of the given fuel**. The car has no problems and adds all the given fuel to its tank. If a vehicle **cannot travel** the given distance, its fuel **does not change**.

Note: Submit all your classes and imports in the judge system

Examples

Test Code	Output

`car = Car(20, 5)`
`car.drive(3)`
`print(car.fuel_quantity)`
`car.refuel(10)`
`print(car.fuel_quantity)`
	`2.299999999999997`
	`12.299999999999997`

`truck = Truck(100, 15)`
`truck.drive(5)`
`print(truck.fuel_quantity)`
`truck.refuel(50)`
`print(truck.fuel_quantity)`
	`17.0`
	`64.5`

2. Wild Farm

Your task is to create a class **hierarchy** like the described below. The **`Animal`**, **`Bird`**, **`Mammal`** and **`Food`** classes should be abstract:

- **`Food`** - **`quantity`** (int) - **abstract class**
  - **`Vegetable`**
  - **`Fruit`**
  - **`Meat`**
  - **`Seed`**
- **`Animal`** - **`name`** (string), **`weight`** (float), **`food_eaten`** (attribute, 0 upon initialization) - **abstract class**
  - **`Bird`** - **`wing_size`** (float) - **abstract class**
    - **`Owl`**
    - **`Hen`**
  - **`Mammal`** - **`living_region`** (string) - **abstract class**
    - **`Mouse`**
    - **`Dog`**

        - **`Cat`**
        - **`Tiger`**

All **animals** should also have the ability to ask for food by producing a sound. **`make_sound()`** method that returns the sound:

- **`Owl`** - **`"Hoot Hoot"`**
- **`Hen`** - **`"Cluck"`**
- **`Mouse`** - **`"Squeak"`**
- **`Dog`** - **`"Woof!"`**
- **`Cat`** - **`"Meow"`**
- **`Tiger`** - **`"ROAR!!!"`**

Now use the classes that you have created to instantiate some animals and feed them. Add method **`feed(food)`** where the food will be instance of some of the food classes.

Animals will only eat a certain type of food, as follows:

- Hens eat **everything**
- Mice eat **vegetables** and **fruits**
- Cats eat **vegetables** and **meat**
- Tigers, Dogs and Owls eat only **meat**

If you try to give an animal a **different type** of food, it will not eat it and you should return:

- **`"{AnimalType} does not eat {FoodType}!"`**

The weight of an animal will increase with every piece of food it eats, as follows:

- Hen - **0.35**
- Owl - **0.25**
- Mouse - **0.10**
- Cat - **0.30**
- Dog - **0.40**
- Tiger - **1.00**

Override the **`__repr__()`** method to print the information about an animal in the formats:

- Birds - **`"{AnimalType} [{AnimalName}, {WingSize}, {AnimalWeight}, {FoodEaten}]"`**
- Mammals - **`"{AnimalType} [{AnimalName}, {AnimalWeight}, {AnimalLivingRegion}, {FoodEaten}]"`**

<u>Note: Submit all your classes and your imports in the judge system</u>

Examples

Test Code	Output

`owl = Owl("Pip", 10, 10)`
`print(owl)`
`meat = Meat(4)`
`print(owl.make_sound())`
`owl.feed(meat)`
`veg = Vegetable(1)`
`print(owl.feed(veg))`
	`Owl [Pip, 10, 10, 0]`
	`Hoot Hoot`
	`Owl does not eat Vegetable!`
	`Owl [Pip, 10, 11.0, 4]`

`print(owl)`	

`hen = Hen("Harry", 10, 10)`
`veg = Vegetable(3)`
`fruit = Fruit(5)`
`meat = Meat(1)`
`print(hen)`
`print(hen.make_sound())`
`hen.feed(veg)`
`hen.feed(fruit)`
`hen.feed(meat)`
`print(hen)`
	`Hen [Harry, 10, 10, 0]`
	`Cluck`
	`Hen [Harry, 10, 13.15, 9]`