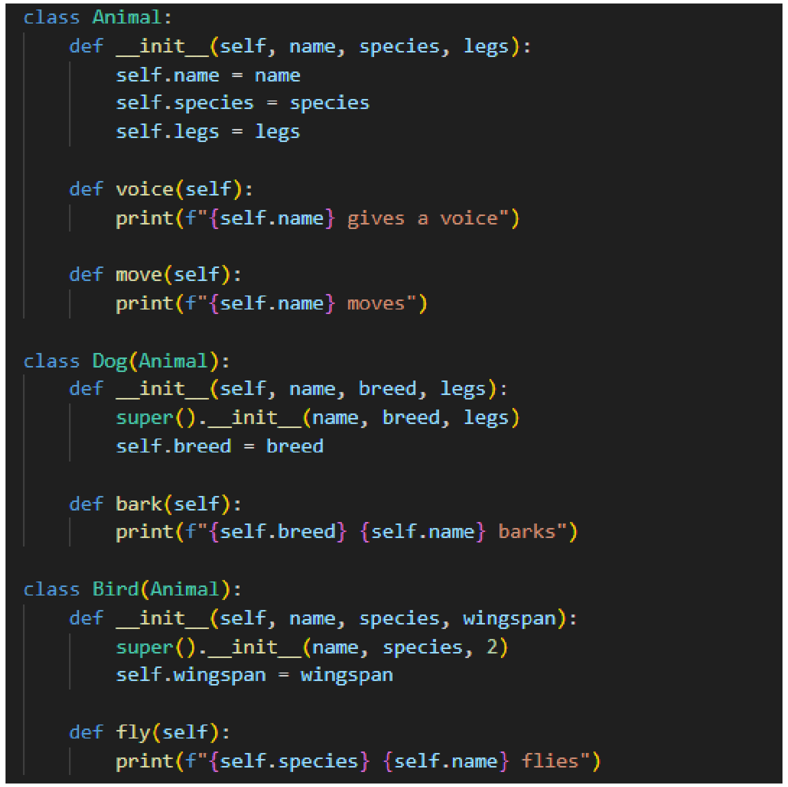
Revision

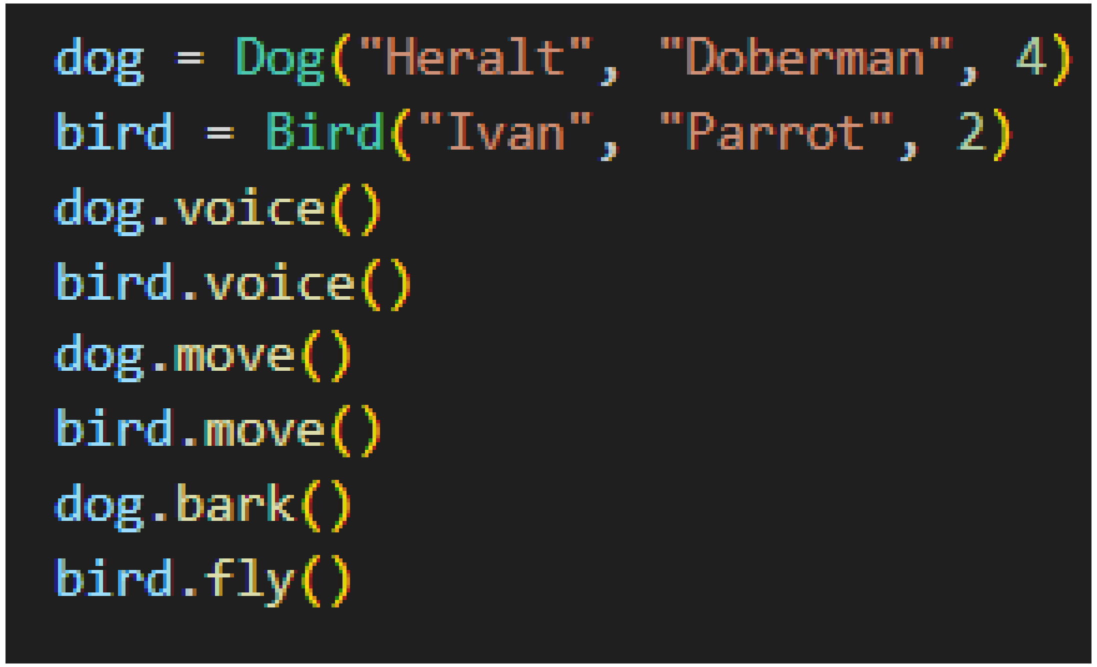
- 1. Create class **“Animal”**
- Properties: **name, species, legs**
- **Methods:**
- **voice() – outputs “___ gives a voice”**
- **move() – outputs “___ moves”**
- **Subclasses: Dog and Bird**
- **Dog consists method bark()**
- **Bird consists method fly()**
- **Add properties and methods as you wish.**

Answer

Answer of code and using



```python
class Animal:
    def __init__(self, name, species, legs):
        self.name = name
        self.species = species
        self.legs = legs

    def voice(self):
        print(f"{self.name} gives a voice")

    def move(self):
        print(f"{self.name} moves")

class Dog(Animal):
    def __init__(self, name, breed, legs):
        super().__init__(name, breed, legs)
        self.breed = breed

    def bark(self):
        print(f"{self.breed} {self.name} barks")

class Bird(Animal):
    def __init__(self, name, species, wingspan):
        super().__init__(name, species, 2)
        self.wingspan = wingspan

    def fly(self):
        print(f"{self.species} {self.name} flies")
```



```python
dog = Dog("Heralt", "Doberman", 4)
bird = Bird("Ivan", "Parrot", 2)
dog.voice()
bird.voice()
dog.move()
bird.move()
dog.bark()
bird.fly()
```

Polymorphism

11.4.1.4 create a class hierarchy;

11.4.2.1 explain the concept of polymorphism with examples;

11.4.2.2 explain the concept of inheritance with examples;

11.4.3.2 solve applied problems of various subject areas.

What is polymorphism?

- The literal meaning of polymorphism is the condition of occurrence in different forms.
- Polymorphism is a very important concept in programming. It refers to the use of a single type entity (method, operator or object) to represent different types in different scenarios.

- https://youtu.be/C2QfkDcQ5MU?si=VAXuV13Ix2lEfVyd&t=573

Python Polymorphism

- Read information about Polymorphism from next link.

Task

- **Shape Area Calculator:**
- **Base Class: ‘Shape’**

  Properties: name , color

  Method: calculate_area()

Subclasses : Circle, Rectangle, Triangle

Additional properties:

  Circle: radius

  Rectangle: width , length

  Triangle: bases, height

Override ‘calculate_area()’ method in each subclass

Task

- Base Class “Animal”
  - Properties: name, age
  - Method: ‘make_sound()’
  - Subclass: Dog, Cat, Cow
  - Additional properties: Dog – breed, Cat – color, Cow – milk_production
  - Override make_sound() method in each subclass

Task

- Base Class: Transport
- Method: travel()
- Subclasses: Car, Bicycle, Plane
- Additional properties: Car- fuel_type, Bicycle – number_of_gears, Plane – flight_range
- Override travel() method in each subclass to describe how that mode of transportation moves.

Task

- Class: Employee
- Properties: name, id, position
- Method: calculate_salary()
- Subclasses: HourlyEmployee, SalariedEmployee, CommissionedEmployee
- Additional Properties: HourlyEmployee: hourly_rate, hours_worked SalariedEmployee: salary CommissionedEmployee: base_salary, commission_rate, sales_volume
- Override calculate_salary() method in each subclass to compute the salary specific to that type of employee.