**# Базовый класс для группы питомцев**

**class Pets:**

**def \_\_init\_\_(self, animals):**

**self.animals = animals**

**def walk(self):**

**for animal in self.animals:**

**print(animal.walk())**

**# Базовый класс Кошка**

**class Cat:**

**is\_lazy = True**

**def \_\_init\_\_(self, name, age):**

**self.name = name**

**self.age = age**

**def walk(self):**

**return f"{self.name} is just walking around."**

**# Подклассы кошек**

**class Bengal(Cat):**

**def sing(self, sounds):**

**return f"{sounds}"**

**class Chartreux(Cat):**

**def sing(self, sounds):**

**return f"{sounds}"**

**# ✅ Step 1: Создаём новый подкласс Siamese**

**class Siamese(Cat):**

**def sing(self, sounds="Meooow"):**

**return f"{self.name} sings: {sounds}"**

**# ✅ Step 2: Создаём список котов**

**bengal\_obj = Bengal("Luna", 3)**

**chartreux\_obj = Chartreux("Milo", 5)**

**siamese\_obj = Siamese("Simba", 2)**

**all\_cats = [bengal\_obj, chartreux\_obj, siamese\_obj]**

**# ✅ Step 3: Создаём экземпляр Pets**

**sara\_pets = Pets(all\_cats)**

**# ✅ Step 4: Прогуливаем всех котов**

**sara\_pets.walk()**

**# Step 1: Import the Dog class**

**# from dog\_module import Dog # если класс Dog в отдельном файле dog\_module.py**

**# Для примера мы повторно определим Dog здесь**

**import random**

**class Dog:**

**def \_\_init\_\_(self, name, age, weight):**

**self.name = name**

**self.age = age**

**self.weight = weight**

**def bark(self):**

**return f"{self.name} is barking"**

**def run\_speed(self):**

**return (self.weight / self.age) \* 10**

**def fight(self, other\_dog):**

**self\_power = self.run\_speed() \* self.weight**

**other\_power = other\_dog.run\_speed() \* other\_dog.weight**

**if self\_power > other\_power:**

**return f"{self.name} won the fight against {other\_dog.name}"**

**elif self\_power < other\_power:**

**return f"{other\_dog.name} won the fight against {self.name}"**

**else:**

**return "It's a draw!"**

**# Step 2: Create the PetDog Class**

**class PetDog(Dog):**

**def \_\_init\_\_(self, name, age, weight, trained=False):**

**super().\_\_init\_\_(name, age, weight)**

**self.trained = trained**

**def train(self):**

**print(self.bark())**

**self.trained = True**

**def play(self, \*args):**

**names = [dog.name for dog in args]**

**names.append(self.name)**

**print(f"{', '.join(names)} all play together")**

**def do\_a\_trick(self):**

**if self.trained:**

**tricks = ["does a barrel roll", "stands on his back legs", "shakes your hand", "plays dead"]**

**trick = random.choice(tricks)**

**print(f"{self.name} {trick}")**

**else:**

**print(f"{self.name} is not trained yet.")**

**# Step 3: Test PetDog Methods**

**dog1 = PetDog("Rex", 5, 20)**

**dog2 = PetDog("Buddy", 3, 15)**

**dog3 = PetDog("Max", 7, 30)**

**# Train dog1 and dog2**

**dog1.train()**

**dog2.train()**

**# Dogs play together**

**dog1.play(dog2, dog3)**

**# Do a trick**

**dog1.do\_a\_trick()**

**dog2.do\_a\_trick()**

**dog3.do\_a\_trick() # Not trained yet**

**========================================================**

## ****Step 1: Создание класса Dog****

class Dog:

def \_\_init\_\_(self, name, age, weight):

self.name = name

self.age = age

self.weight = weight

* Это **конструктор \_\_init\_\_**: он автоматически вызывается, когда создаётся новый объект.
* У собаки есть три атрибута:
  + name — имя
  + age — возраст
  + weight — вес

### ****Метод bark()****

def bark(self):

return f"{self.name} is barking"

* Когда этот метод вызывается, он возвращает строку вида:  
  "Rex is barking".
* То есть собака «лает».

### ****Метод run\_speed()****

def run\_speed(self):

return (self.weight / self.age) \* 10

* Формула для скорости бега:
* run\_speed = (вес / возраст) \* 10
* Чем моложе и тяжелее собака — тем быстрее она бежит.

### ****Метод fight(other\_dog)****

def fight(self, other\_dog):

self\_power = self.run\_speed() \* self.weight

other\_power = other\_dog.run\_speed() \* other\_dog.weight

* Для каждой собаки считается **сила**:
* сила = run\_speed \* weight
* Затем сравниваются силы двух собак.

if self\_power > other\_power:

return f"{self.name} won the fight against {other\_dog.name}"

elif self\_power < other\_power:

return f"{other\_dog.name} won the fight against {self.name}"

else:

return "It's a draw!"

* Побеждает та собака, у которой сила выше.
* Если силы равны — ничья.

## ****Step 2: Создание объектов****

dog1 = Dog("Rex", 5, 20)

dog2 = Dog("Buddy", 3, 15)

dog3 = Dog("Max", 7, 30)

* Создали 3 разных собаки:
  + **Rex**: возраст 5, вес 20
  + **Buddy**: возраст 3, вес 15
  + **Max**: возраст 7, вес 30

## ****Step 3: Тестирование методов****

print(dog1.bark())

print(f"{dog1.name}'s run speed: {dog1.run\_speed():.2f}")

print(dog1.fight(dog2))

* dog1.bark() → "Rex is barking"
* dog1.run\_speed() → скорость Рекса, например 40.0
* dog1.fight(dog2) → кто победил в бою между Рексом и Бадди

То же самое делается для других собак.

## 📌 Итого:

* Класс Dog описывает модель собаки с атрибутами и поведением.
* Каждый объект (dog1, dog2, dog3) — это конкретная собака.
* Методы позволяют им «лаять», «бегать» и «драться».