**Prompt: *“Viết ví dụ Python minh họa nguyên lý kế thừa (Inheritance) trong OOP với lớp Animal và Dog. Giải thích từng dòng code.”***

**Đoạn code :**

1 class Animal:

2 def \_\_init\_\_(self, name):

3 self.name = name

4

5 def speak(self):

6 return f"{self.name} makes a sound."

7

8 def info(self):

9 return f"Animal(name={self.name})"

10

11 class Dog(Animal):

12 def \_\_init\_\_(self, name, breed):

13 super().\_\_init\_\_(name) # gọi constructor lớp cha

14 self.breed = breed

15

16 def speak(self): # override phương thức speak

17 return f"{self.name} says: Woof!"

18

19 def info(self): # mở rộng/ghi đè info, kết hợp với super()

20 parent\_info = super().info()

21 return parent\_info + f", breed={self.breed}"

22

23 if \_\_name\_\_ == "\_\_main\_\_":

24 my\_dog = Dog("Buddy", "Beagle")

25 print(my\_dog.info()) # In thông tin (sử dụng method ghi đè)

26 print(my\_dog.speak()) # In âm thanh (phương thức đã bị ghi đè)

27 print(isinstance(my\_dog, Animal)) # Kiểm tra là instance của Animal?

28 print(issubclass(Dog, Animal)) # Kiểm tra Dog có phải subclass của Animal?

**Giải thích từng dòng (theo số dòng trong đoạn code) :**

1. class Animal:
   * Khai báo một lớp tên Animal — đây sẽ là **lớp cha (base class)** chứa thuộc tính và hành vi chung cho các loại động vật.
2. def \_\_init\_\_(self, name):
   * Định nghĩa **constructor** cho lớp Animal. Khi tạo instance, hàm này được gọi để khởi tạo trạng thái ban đầu.
3. self.name = name
   * Gán tham số name vào thuộc tính instance self.name — mỗi đối tượng Animal sẽ lưu tên của nó.
4. (dòng trống)
   * Chỉ để tách logic, không có tác dụng về mặt thực thi.
5. def speak(self):
   * Khai báo một phương thức speak trong lớp Animal — phương thức này mô tả "động vật phát ra âm thanh".
6. return f"{self.name} makes a sound."
   * Mặc định trả về một chuỗi chung; lớp con thường **ghi đè (override)** phương thức này để đưa behavior cụ thể hơn (ví dụ: chó sủa).
7. (dòng trống)
   * Tách phần code cho dễ đọc.
8. def info(self):
   * Định nghĩa phương thức info để trả về chuỗi mô tả thông tin cơ bản của đối tượng.
9. return f"Animal(name={self.name})"
   * Trả về chuỗi chứa tên — lớp con có thể tái sử dụng hoặc ghi đè phương thức này.
10. (dòng trống)
    * Tách biệt giữa hai lớp.
11. class Dog(Animal):
    * Khai báo lớp Dog **kế thừa** từ Animal. Cú pháp (Animal) nghĩa là Dog thừa hưởng thuộc tính và phương thức từ Animal.
12. def \_\_init\_\_(self, name, breed):
    * Constructor của lớp Dog, nhận thêm tham số breed (giống chó).
13. super().\_\_init\_\_(name) # gọi constructor lớp cha
    * Gọi constructor của lớp cha (Animal.\_\_init\_\_) để xử lý phần khởi tạo chung (gán name). Dùng super() để truy cập hành vi lớp cha.
14. self.breed = breed
    * Gán tham số breed vào thuộc tính self.breed — đây là phần mở rộng riêng cho Dog.
15. (dòng trống)
    * Tách logic.
16. def speak(self): # override phương thức speak
    * Lớp Dog định nghĩa lại (override) phương thức speak của lớp cha để cung cấp hành vi cụ thể cho chó.
17. return f"{self.name} says: Woof!"
    * Khi gọi speak() trên đối tượng Dog, phương thức này sẽ chạy (thay vì phiên bản trong Animal) và trả về chuỗi biểu diễn âm thanh đặc trưng.
18. (dòng trống)
19. def info(self): # mở rộng/ghi đè info, kết hợp với super()
    * Lớp Dog cũng ghi đè phương thức info để thêm thông tin breed vào phần mô tả.
20. parent\_info = super().info()
    * Gọi info() của lớp cha để lấy phần mô tả cơ bản; đây là ví dụ dùng super() để **tái sử dụng** logic của lớp cha.
21. return parent\_info + f", breed={self.breed}"
    * Trả về chuỗi kết hợp thông tin lớp cha và thông tin riêng của Dog.
22. (dòng trống)
23. if \_\_name\_\_ == "\_\_main\_\_":
    * Kiểm tra xem file này có đang được chạy trực tiếp (không phải import) — nếu đúng thì thực thi khối bên trong. Thường dùng để cho ví dụ hoặc test nhanh.
24. my\_dog = Dog("Buddy", "Beagle")
    * Tạo một instance Dog tên "Buddy", giống "Beagle". Khi tạo, Python gọi Dog.\_\_init\_\_, bên trong nó gọi Animal.\_\_init\_\_ qua super().
25. print(my\_dog.info()) # In thông tin (sử dụng method ghi đè)
    * Gọi info() trên my\_dog. Vì Dog ghi đè info, phiên bản trong Dog chạy, kết quả sẽ chứa cả tên lẫn breed (ví dụ: Animal(name=Buddy), breed=Beagle).
26. print(my\_dog.speak()) # In âm thanh (phương thức đã bị ghi đè)
    * Gọi speak() trên my\_dog. Do Dog ghi đè, sẽ in "Buddy says: Woof!".
27. print(isinstance(my\_dog, Animal)) # Kiểm tra là instance của Animal?
    * Kiểm tra my\_dog có phải là thể hiện (instance) của lớp Animal không. Kết quả True vì Dog kế thừa từ Animal.
28. print(issubclass(Dog, Animal)) # Kiểm tra Dog có phải subclass của Animal?
    * Kiểm tra mối quan hệ lớp: Dog có phải là subclass (lớp con) của Animal không — trả về True.

**Ghi nhận (những điểm quan trọng liên quan đến kế thừa được minh họa)**

* Dog(Animal) cho thấy **quan hệ kế thừa**: Dog nhận các thuộc tính/phương thức từ Animal.
* super().\_\_init\_\_(...) minh họa **tái sử dụng constructor** của lớp cha.
* Ghi đè phương thức (def speak trong Dog) thể hiện **polymorphism** — cùng tên phương thức nhưng hành vi khác nhau tùy lớp.
* isinstance và issubclass dùng để kiểm tra mối quan hệ giữa objects và classes.