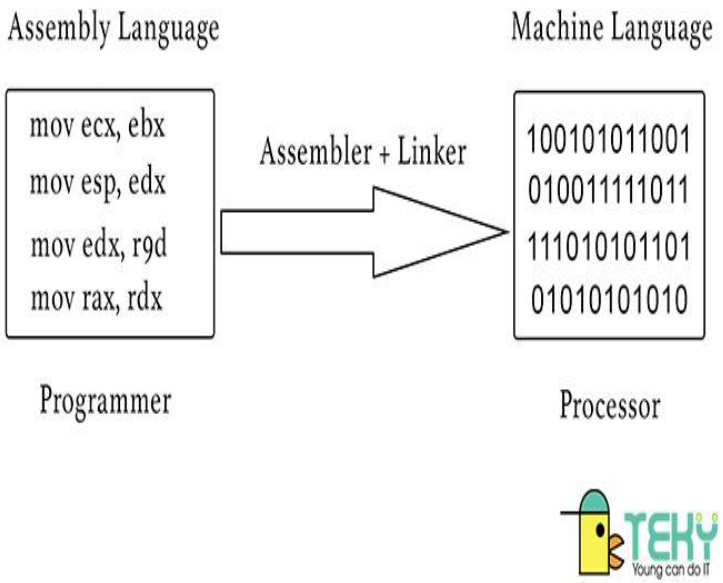
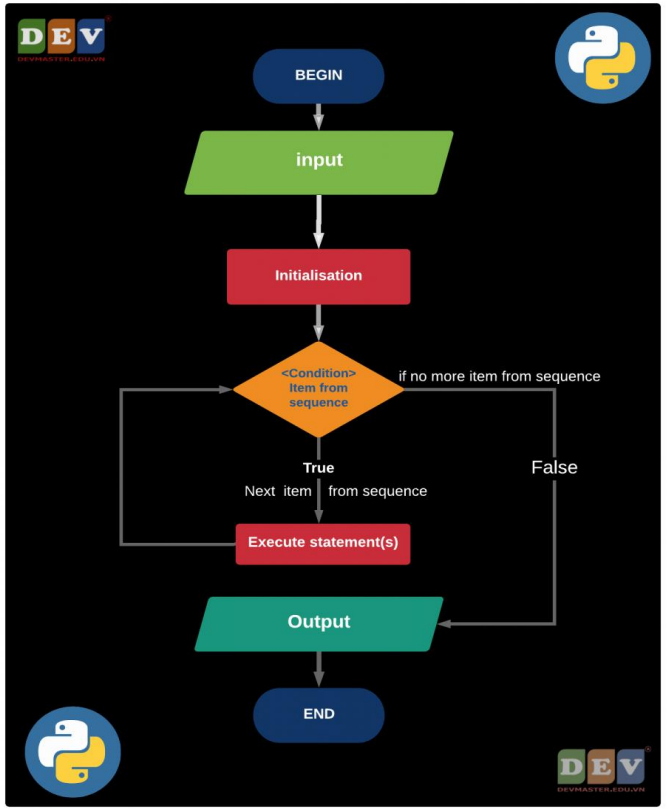
Lập trình hướng đối tượng

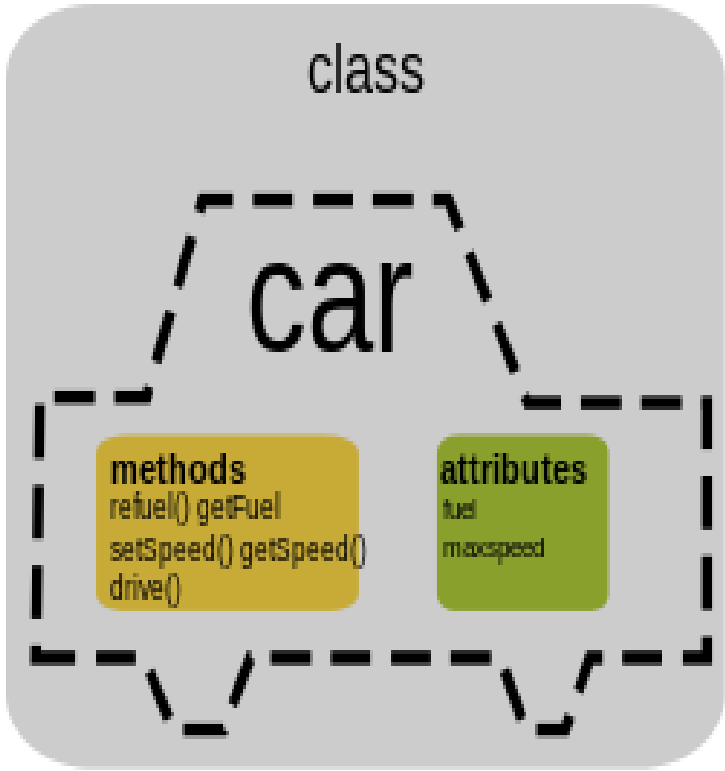
Trình bày: Nguyễn Tuấn Anh

Nội dung

- Các phương pháp lập trình
- Một số khái niệm cơ bản
- Các đặc điểm quan trọng của OOP
- Các ưu điểm của OOP
- Bài tập vận dụng







Các phương pháp lập trình

Lập trình không có cấu trúc

- Các ngôn ngữ như Assembly, Basic
- Sử dụng các biến toàn cục
- Khó hiểu, khó bảo trì, hầu như không thể sử dụng lại
- Chất lượng kém, Chi phí cao
- Không thể phát triển các ứng dụng lớn

Lập trình có cấu trúc

- Tổ chức thành các chương trình con
- Mỗi chương trình con đảm nhận xử lý một công việc nhỏ hay một nhóm công việc trong toàn bộ hệ thống.
- Mỗi chương trình con này lại có thể chia nhỏ thành các chương trình con nhỏ hơn.

- Chương trình = Cấu trúc dữ liệu + Giải thuật

Ưu điểm

- Chương trình được module hóa
- dễ hiểu, dễ bảo trì hơn
- Dễ dàng tạo ra các thư viện phần mềm

Nhược điểm

- Dữ liệu và mã xử lý là tách rời
- Người lập trình phải biết cấu trúc dữ liệu
- Khi thay đổi cấu trúc dữ liệu, thuật toán phải thay đổi theo
- Khó đảm bảo tính đúng đắn của dữ liệu
- Không tự động khởi tạo hay giải phóng dữ liệu động
- Không mô tả được đầy đủ hệ thống trong thực tế

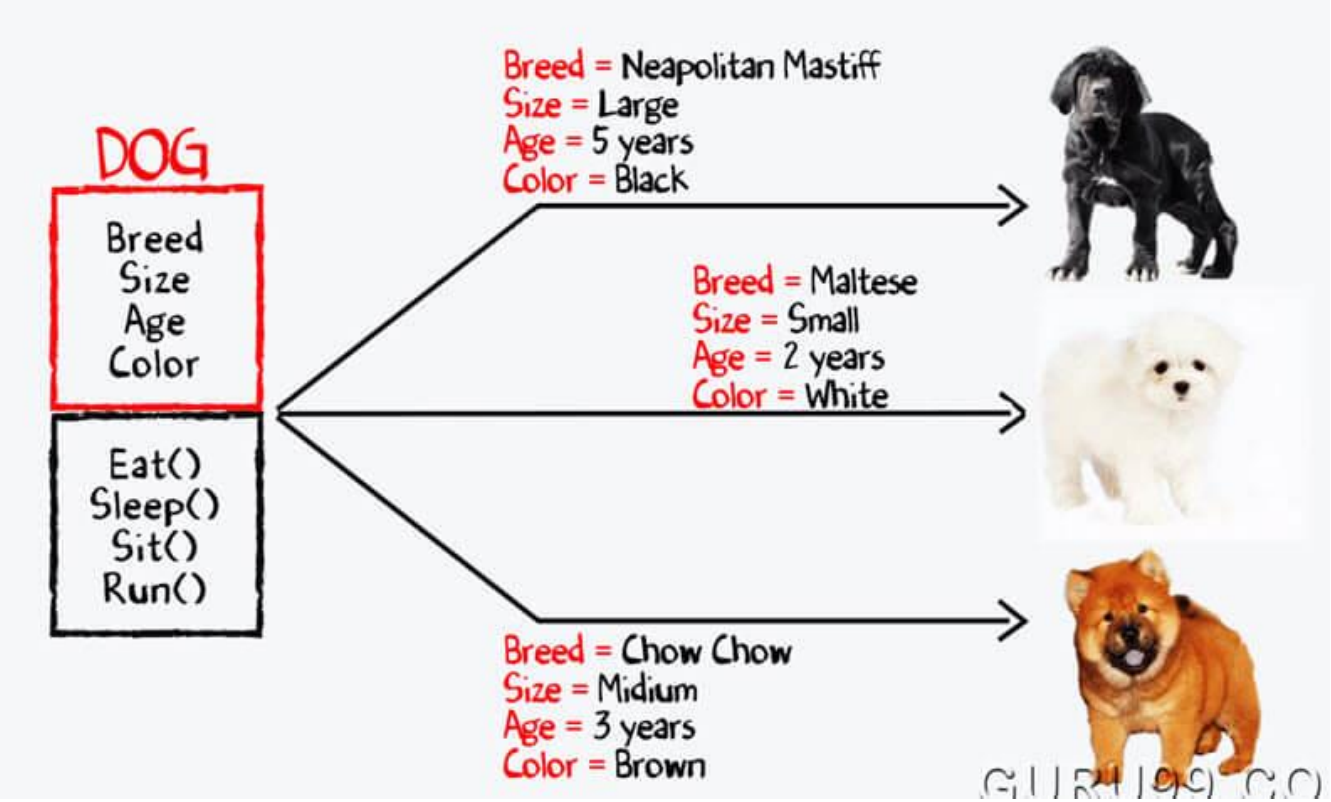
Lập trình hướng đối tượng

- Là phương pháp lập trình lấy đối tượng làm nền tảng để xây dựng thuật giải, xây dựng chương trình.
- Đối tượng = Dữ liệu + Phương thức

Lập trình hướng đối tượng

dựa trên kiến trúc:

- Lớp (class)
- Đối tượng (object)



Một số khái niệm cơ bản

- Đối tượng(object):
  - Trong thế giới thực, đối tượng được hiểu như là một thực thể: người, vật hoặc một bảng dữ liệu
  - Đối tượng giúp biểu diễn tốt hơn thế giới thực trên máy tính
  - Mỗi đối tượng bao gồm 2 thành phần thuộc tính và hành động.
- Lớp (class):
  - Các đối tượng có các đặc tính tương tự nhau được gom chung thành lớp đối tượng

Ví dụ 1

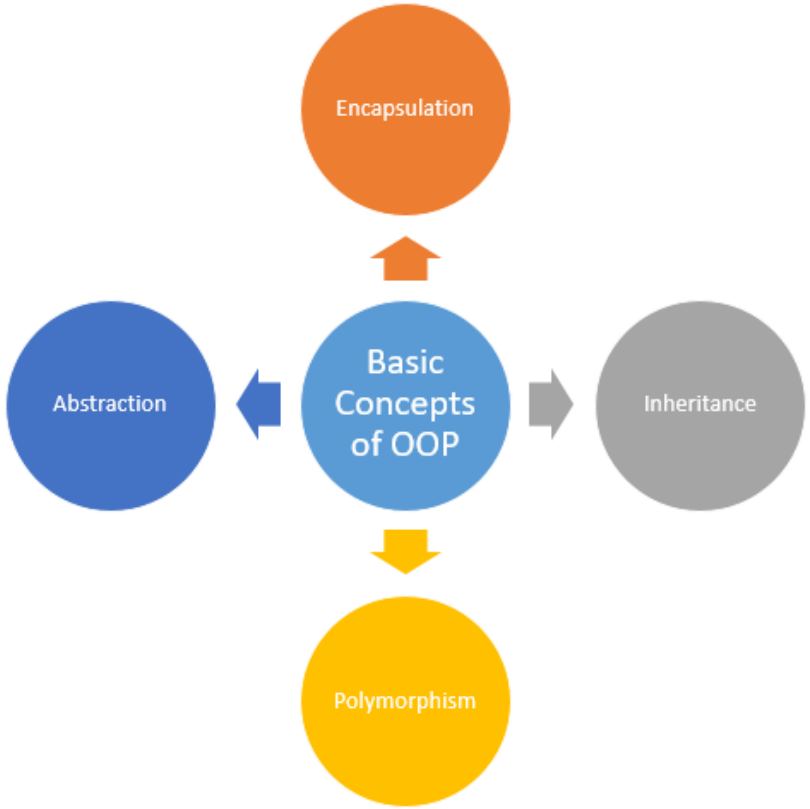
```
class Car:
     # thuộc tính lớp
     loaixe = "Ô tô"

     # thuộc tính đối tượng
     def __init__(self, tenxe, mausac, nguyenlieu):
        self.tenxe = tenxe
        self.mausac = mausac
        self.nguyenlieu = nguyenlieu

# instantiate the Car class
toyota = Car("Toyota", "Đỏ", "Điện")
lamborghini = Car("Lamborghini", "Vàng", "Deisel")

# access the class attributes
print("Toyota là {}.".format(toyota.__class__.loaixe))
print("Lamborghini cũng là
{}.".format(lamborghini.__class__.loaixe))

# access the instance attributes
print("Xe {} có màu {}. {} là nguyên liệu vận hành.".format(
toyota.tenxe, toyota.mausac, toyota.nguyenlieu))


print("Xe {} có màu {}. {} là nguyên liệu vận hành.".format(
lamborghini.tenxe, lamborghini.mausac,lamborghini.nguyenlieu))
```

Ví dụ 2: Phương thức

```
class Car:
     # thuộc tính đối tượng
     def __init__(self, tenxe, mausac, nguyenlieu):
        self.tenxe = tenxe
        self.mausac = mausac
        self.nguyenlieu = nguyenlieu

     # phương thức
     def dungxe(self, mucdich):
        return "{} đang dừng xe để{}".format(self.tenxe,mucdich)

     def chayxe(self):
        return "{} đang chạy trên đường".format(self.tenxe)

     def nomay(self):
       return "{} đang nổ máy".format(self.tenxe)

# instantiate the Car class
toyota = Car("Toyota", "Đỏ", "Điện")
lamborghini = Car("Lamborghini", "Vàng", "Deisel")

# call our instance methods
print(toyota.dungxe("nạp điện"))
print(lamborghini.chayxe())
```

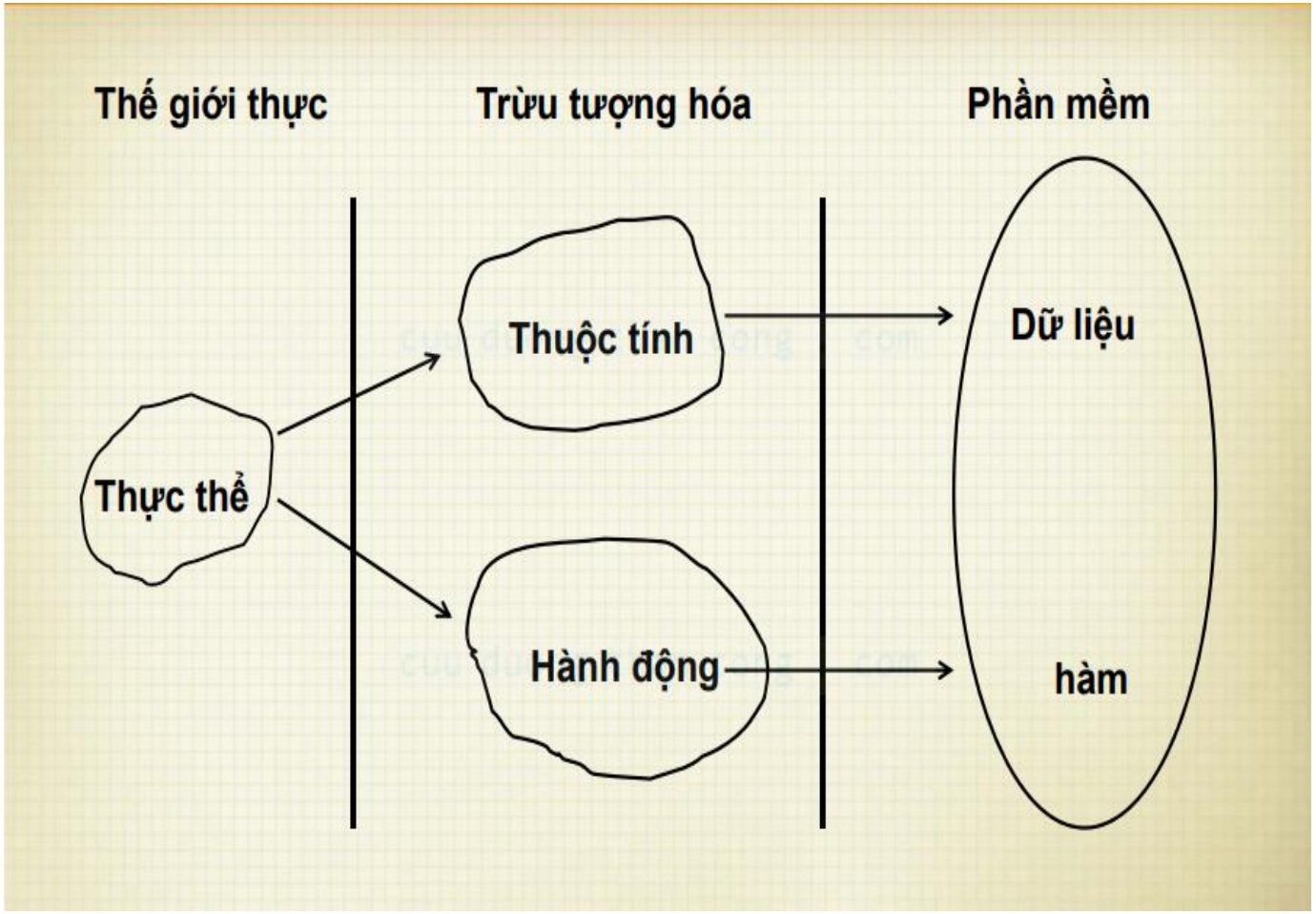
Các đặc điểm quan trọng của OOP

- Tính trừu tượng (Abstraction)
- Tính đóng gói (Encapsulation)
- Tính kế thừa (Inheritance)
- Tính đa hình (Polymorphism)



Trừu tượng hóa

- Khái quát hóa về một tập các đối tượng có chung các đặc điểm
- Bỏ qua những chi tiết không cần thiết

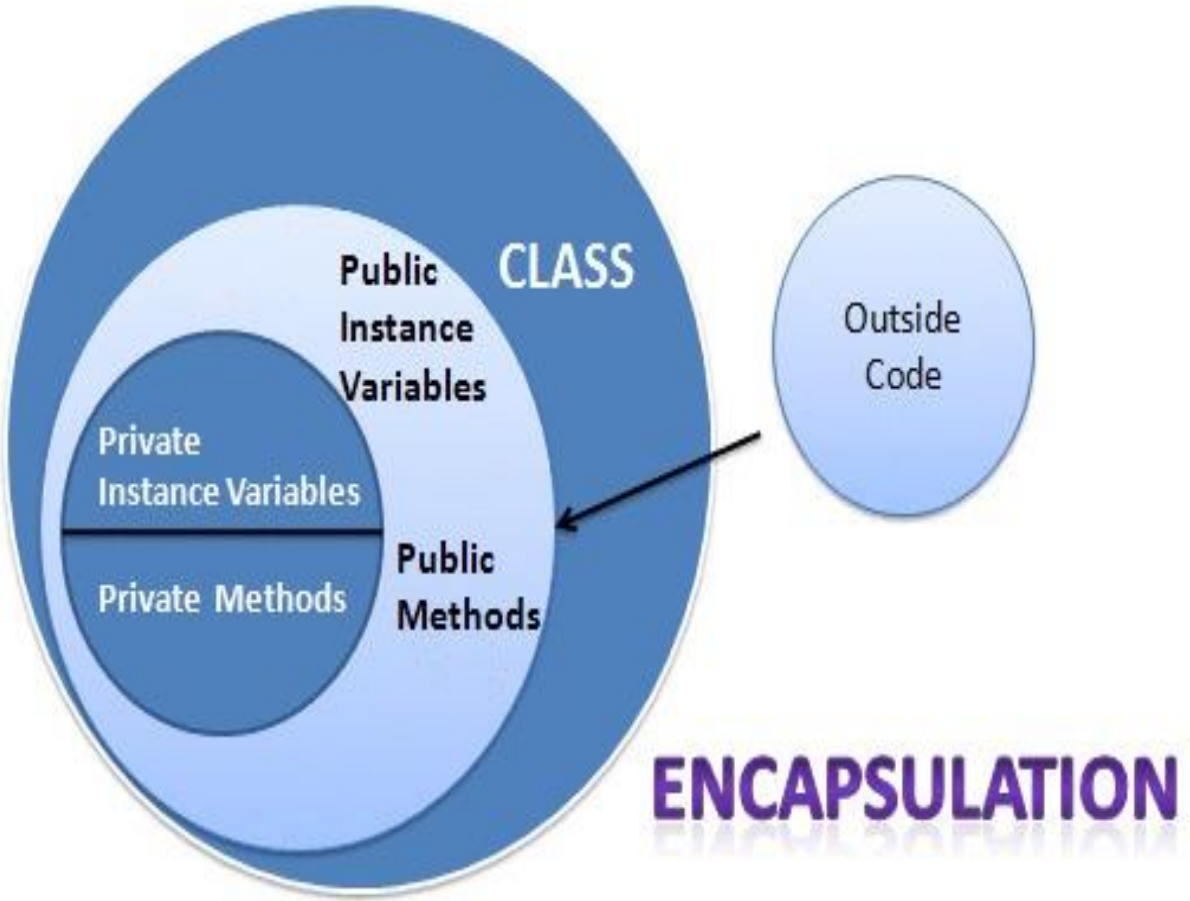


Đóng gói

- Đóng gói: Nhóm những gì có liên quan với nhau vào làm một, để sau này có thể dùng một cái tên để gọi đến
- Các hàm/ thủ tục đóng gói các câu lệnh
- Các đối tượng đóng gói dữ liệu của chúng và các thủ tục có liên quan

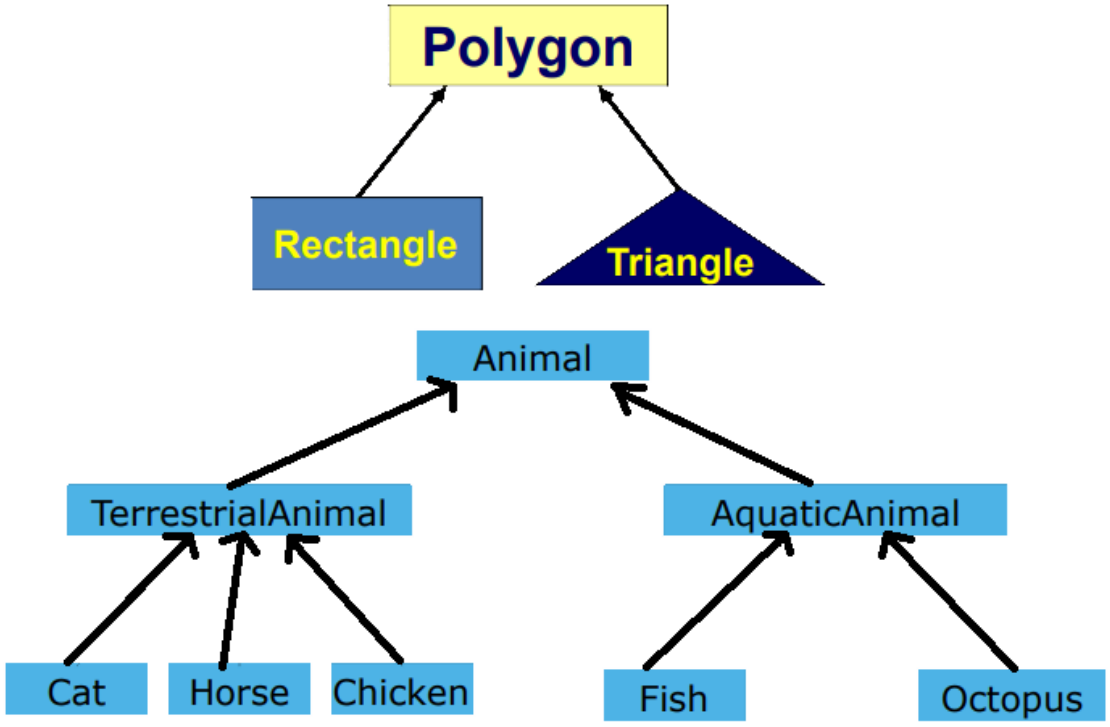
Che dấu thông tin

che một số thông tin và chi tiết cài đặt nội bộ để bên ngoài không nhìn thấy



Kế thừa

- Xây dựng một lớp mới dựa trên các định nghĩa của lớp đã có là lớp cha
- Nhằm tái sử dụng lại mã nguồn



Ví dụ 3: Kế thừa

```
class Xe:
    name = 'Đây là tên xe'

class XeDap(Xe):
    def showName(self):
        # sử dụng thuộc tính name của lớp cha
        print(self.name)


# Cách dùng
d = XeDap()
d.showName()
```

Đa hình

- Cho phép một tên thao tác hoặc thuộc tính có thể được định nghĩa tại nhiều lớp
- Có thể có nhiều cài đặt khác nhau tại mỗi lớp trong các lớp đó.

Ưu điểm của OOP

- Dễ mở rộng, nâng cấp
- Mô phỏng thế giới thực tốt hơn.
- Nguyên lý kế thừa: tránh lặp, tái sử dụng.
- Nguyên lý đóng gói - che dấu thông tin:
  - chương trình an toàn không bị thay đổi bởi những đoạn chương trình khác

Đặc tính chính của OOP

- Chương trình được chia thành các đối tượng
- Mô phỏng thế giới thực tốt hơn.
- Nguyên lý kế thừa: tránh lặp, tái sử dụng.
- Nguyên lý đóng gói - che dấu thông tin:
  - chương trình an toàn không bị thay đổi bởi những đoạn chương trình khác

Bài tập vận dụng

Viết 2 class: Polygon và Triangle trong đó:

- Polygon có các phương thức: nhập vào độ dài của các cạnh, in ra độ dài của các cạnh.
- Triangle kế thừa Polygon, có thêm phương thức tính chu vi và diện tích.