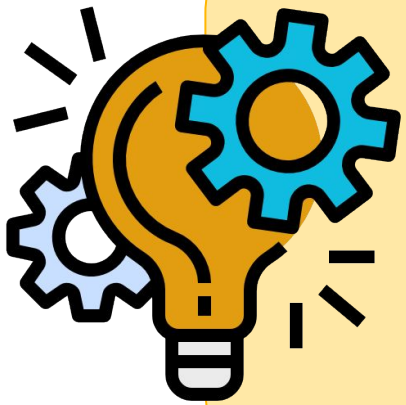
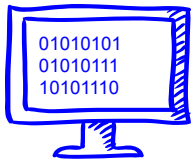
STEAM
FOR VIETNAM



FLASH CARD

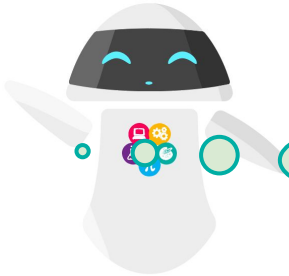
LESSON 6



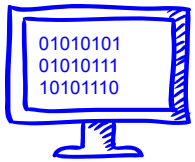
LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

OBJECT-ORIENTED PROGRAMMING

- Dựa vào đề bài, chúng ta hãy đánh dấu **nhân vật, đồ vật** cùng với *đặc điểm* và *các hành động* của chúng để nắm chắc yêu cầu của game nhé!
- Cách chia nhỏ chương trình thành các **vật thể (object)** có các **đặc điểm** và **hành động** được gọi là **lập trình hướng đối tượng (Object-Oriented Programming)**



Dùng phương pháp
Tách Lớn Ra Nhỏ



LỚP
CLASS

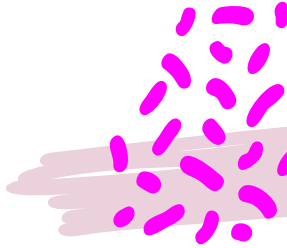
- **Lớp** được ví như một bản mẫu để tạo nên các **đối tượng** có cùng các **thuộc tính** và **phương thức**
- Hãy cùng Trẩu ghi nhớ các từ vựng Tiếng Anh của bài học hôm nay nhé:

lớp → **class**

đối tượng → **object**

thuộc tính → **attribute**

phương thức → **method**



ĐỐI TƯỢNG
OBJECT

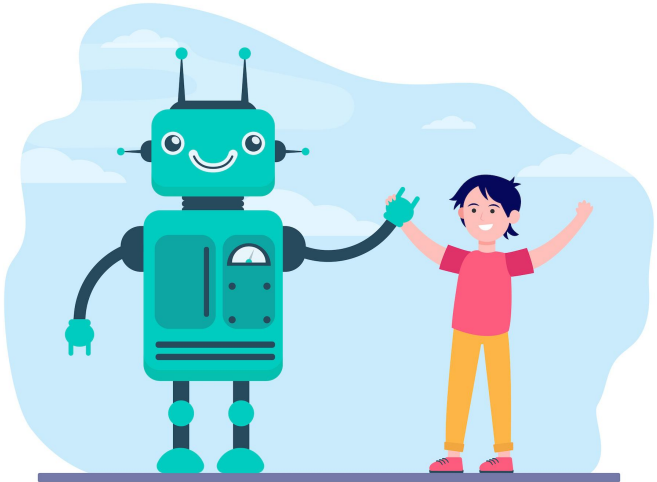
ĐỐI TƯỢNG
(Object)

Thuộc tính
(attribute)

Mô tả các **đặc tính.**
Ví dụ: William
có Tên, Tuổi, Trường,
và Sở thích

Phương thức
(method)

Mô tả **hành động** mà đối tượng có thể làm.
William có thể ăn, học bài, lập trình



TẠO ĐỐI TƯỢNG

- Để tạo đối tượng (object) cho một lớp (class), ta cần có:

  tên biến, tên lớp, và nội dung thuộc tính

```
TRE = Student("Tre", "STEAM for Vietnam", "0913000", "lap trinh")
```

Thuộc tính

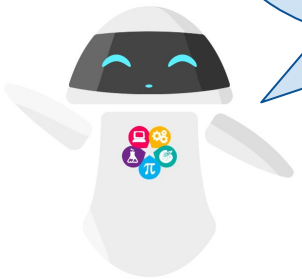
- Tên
- Trường
- Điện thoại
- Sở thích

THUỘC TÍNH
ATTRIBUTE

- Sử dụng hàm **print()** để lấy giá trị thuộc tính. Chỉ có thể lấy được một thuộc tính trong một hàm print()

```
trau = Student("Trau", 12)

print(trau.name)
print(trau.age)
```

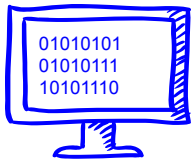


Làm cách nào để **lấy giá trị thuộc tính** của một đối tượng?

```
trau.say_hello()
trau.study()
```

tên của phương thức

chạy phương thức của đối tượng



HÀM __init__

- Hàm **__init__** là hàm chạy đầu tiên khi tạo một đối tượng

tên **lớp**

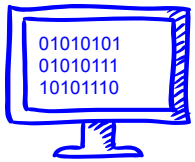
self là đối tượng đang sử dụng phương thức

tạo một lớp

```
class Student:
    def __init__(self, name, school):
        self.name = name
        self.school = school
```

tạo phương thức __init__

thuộc tính

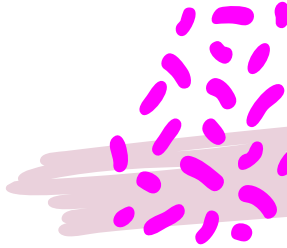


TÍNH KẾ THỪA
INHERITANCE

```
class Pet:
    def __init__(self, name, favorite_food):
        self.name = name
        self.favorite_food = favorite_food
```

lớp Cat kế thừa lớp Pet ⇒ lớp Cat sẽ có các **thuộc tính và phương thức** từ lớp Pet. Đồng thời, lớp Cat có thể tạo thêm thuộc tính và phương thức khác.

```
class Cat(Pet):
    def speak(self):
        print(self.name, "noi meo meo")
```



STEAM
FOR VIETNAM

COMMON MISTAKES

LESSON 6

Các Lỗi Thường Gặp

Lỗi:

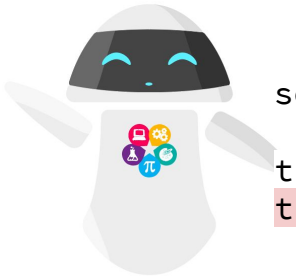
Chương trình không in ra gì

Khi chạy đoạn code:

```
class Student:
    def __init__(self, name,
school):
        self.name = name
        self.school = school

    def say_hello(self):
        print ("Hello, I'm " +
self.name + "!")

trau = Student ("Trau", 12)
trau.say_hello
```



Nguyên nhân:

Nếu thiếu dấu **()** thì phương thức sẽ không được gọi

Khắc phục:

```
trau.say_hello()
```

Các Lỗi Thường Gặp

Bị báo lỗi:

TypeError: say_hello() takes 0 positional arguments but 1 was given

Khi chạy đoạn code:

```
class Student:
    def __init__(self, name, school):
        self.name = name
        self.school = school

    def say_hello():
        print ("Hello, I'm " + self.name + "!")
```

Nguyên nhân:

Chúng ta quên **self** của phương thức say_hello

Khắc phục:

```
def say_hello(self):
    print ("Hello, I'm " + self.name + "!")
```

Các Lỗi Thường Gặp

Bị báo lỗi:

TypeError: can only concatenate str (not "int") to str

Khi chạy đoạn code:

```
class Player:
    def __init__(self):
        self.x = 60
        self.y = 60

    def report(self):
        print("Player's at position (x,y) = (" + self.x + "," + self.y + ")")
```

`self.x` và `self.y` đang là số nguyên

Nguyên nhân:

Chưa đổi kiểu dữ liệu số x và y từ số nguyên (int) sang kiểu dữ liệu (str)

Khắc phục:

```
def report(self):
    print("Player's at position (x,y) = ("
+ str(self.x) + "," + str(self.y) + ")"
```

chuyển đổi sang kiểu dữ liệu sử dụng **`str()`**

Các Lỗi Thường Gặp

Bị báo lỗi:

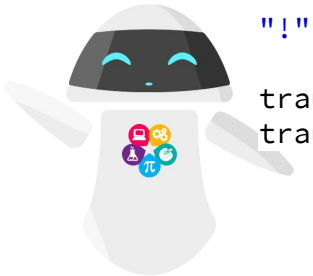
NameError: name 'name' is not defined

Khi chạy đoạn code:

```
class Student:
    def __init__(self, name, school):
        self.name = name
        self.school = school

    def say_hello(self):
        print ("Hello, I'm " + name +
"!")

trau = Student ("Trau", 12)
trau.say_hello()
```



gọi **tên biến** chưa chính xác

Nguyên nhân:

Tên biến chưa chính xác, mình tạo biến **self.name**

Khắc phục:

```
    def say_hello(self):
        print ("Hello, I'm " + self.name +
"!")
```

hãy chú ý và gọi tên biến đúng với tên mình đã tạo nhé!