Introduction to Computer Science

Class

鄒年棣 (Nien-Ti Tsou)

`class`

- Classes are used to create objects. In fact, objects are said to be instances (實例) of classes.
- Example: We all know what a bike is, we know the class. But then I have my own bike, which is an instance of the class bike. And my bike is an object with its own characteristics and methods. You have your own bike. Same class, but different instance. Every bike ever created in the world is an instance of the bike class.

Object-oriented programming (OOP)

- OOP is all about code reuse.
- We define a class, we create instances, and those instances use methods that are defined only in the class. They will behave differently according to how the instances have been set up by the initializer.

Initializer: `__init__`

- Python magic to set up the objects with the values we pass when we create it.
- In other languages, this would be called a constructor.
- When an object is created, the `__init__` method is automatically run for us.
- Every method that has leading and trailing double underscore, in Python, is called magic method. Magic methods are used by Python for a multitude of different purposes, hence, never name a custom method using two leading and trailing underscores.
- `self` is always the first attribute of an instance method.

```
class Bike:
        def __init__(self, color, frame_material):
            self.color = color
            self.frame_material = frame_material
```

Example for `class`

```
class Bike:
    def __init__(self, color, frame_material):
        self.color = color
        self.frame_material = frame_material
    def WhoAmI(self):
        print(self.__class__.__name__)

My_bike = Bike('Red', 'Carbon fiber')
Your_bike = Bike('Blue', 'Steel')
print(My_bike.color)
print(Your_bike.frame_material)
My_bike.WhoAmI()
```

```
Red
Steel
Bike
```

Adding functions to `class`

```
class Account:
    def __init__(self, number, name):
        self.number = number
        self.name = name
        self.balance = 0

    def deposit(self, amount):
        if amount <= 0:
            print('It must be positive')
        else:
            self.balance += amount

    def withdraw(self, amount):
        if amount <= self.balance:
            self.balance -= amount
        else:
            print('Balance is not enough')

acct1 = Account(12345678, 'Justin')
acct1.deposit(100)
acct1.withdraw(30)
print(acct1.balance)

acct2 = Account(87654321, 'Brian')
acct2.withdraw(30)
print(acct2.balance)
```

The methods only belong to this class.

```
70
Balance is not enough
0
```

Inheritance and composition

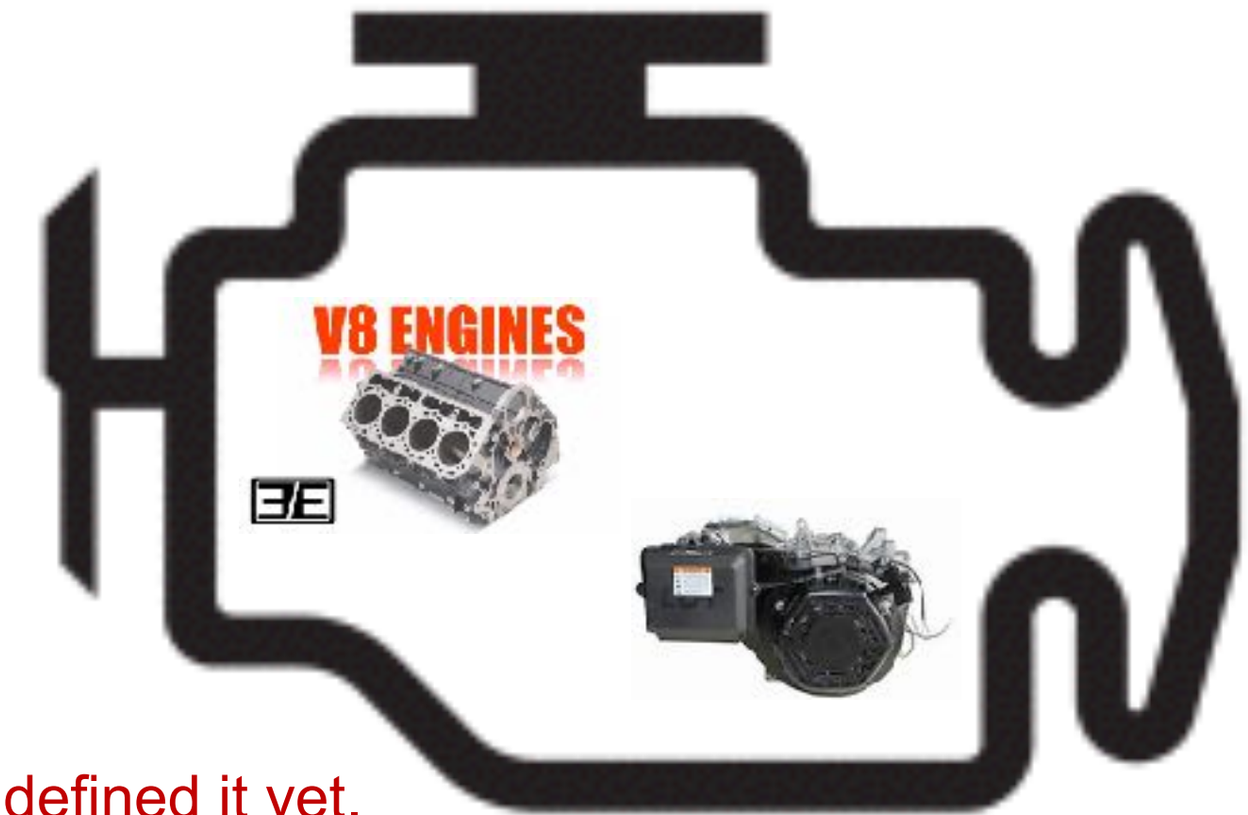
- Inheritance means that two objects are related by means of an Is-A type of relationship.
- Composition means that two objects are related by means of a Has-A type of relationship.

```
class Engine():
    def start(self):
        pass
    def stop(self):
        pass

class ElectricEngine(Engine):
    pass

class V8Engine(Engine):
    pass
```

Inheritance



We haven't defined it yet. Maybe we can leave this for someone pro to do it.

Inheritance and composition

```
class Car():
    engine_cls = Engine
    def __init__(self):
        self.engine = self.engine_cls()
    def start(self):
        print('Starting engine: {0}, for car: {1}... Wroom, wroom!'.format(
                self.engine.__class__.__name__,
                self.__class__.__name__))
        self.engine.start()
    def stop(self):
        self.engine.stop()
```

Composition

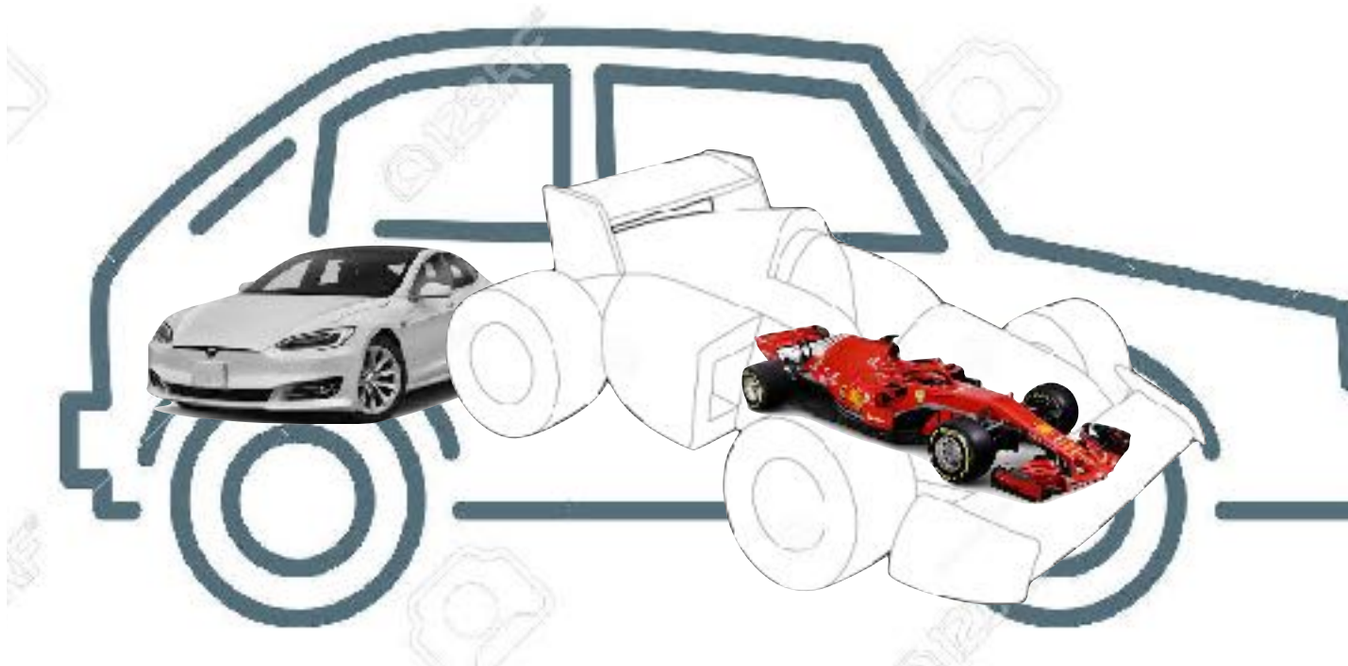
*Note that we don't know which engine this car has exactly. All we know is it is a engine.

Inheritance

```
class RaceCar(Car):
    engine_cls = V8Engine

class CityCar(Car):
    engine_cls = ElectricEngine

class F1Car(RaceCar):
    engine_cls = V8Engine
```



```
My_Car = Car()
Gordon_car = RaceCar()
Musk_car = CityCar()
Schumacher_Car = F1Car()
cars = [My_Car, Gordon_car, Musk_car, Schumacher_Car]
for car in cars:
    car.start()
```

The power of OOP is that all you need to do is call the general function, and everyone can do its job.

```
Starting engine: Engine, for car: Car... Wroom, wroom!
Starting engine: V8Engine, for car: RaceCar... Wroom, wroom!
Starting engine: ElectricEngine, for car: CityCar... Wroom, wroom!
Starting engine: V8Engine, for car: F1Car... Wroom, wroom!
```