OOP
USING PYTHON

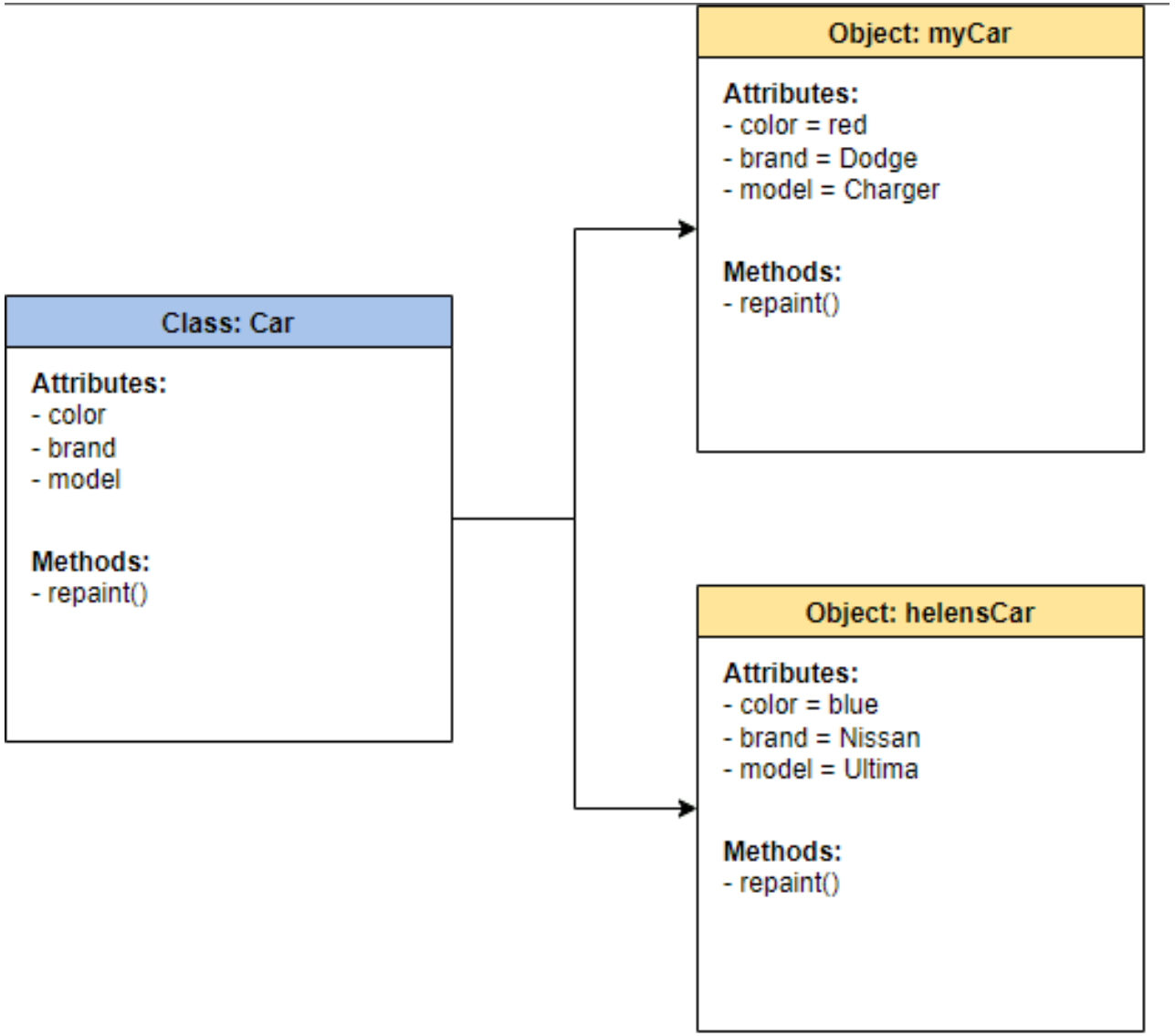
Outline

- OOP definition
- Why OOP
- OOP Keywords
- Relations

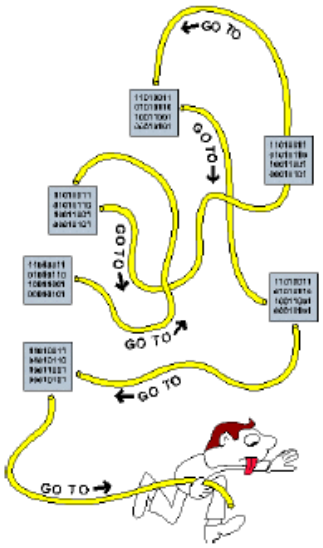
Object-Oriented Programming (OOP)

- Is a programming paradigm in computer science that relies on the concept of classes and objects.
- Think as data and actions.
- It is used to structure a software program into simple, reusable pieces of code blueprints (usually called classes)
- There are many object-oriented programming languages, including JavaScript, C++, Java, and Python.
- Python support OOP and procedural paradigm.

Object-Oriented Programming (OOP)



Why OOP



Speghatti Level

+ Functions

Procedural Level

+ Modules

Modular Level

Object Oriented Level

Why OOP

- Abstraction
- Encapsulation
- Polymorphism
- Inheritance

OOP Keywords

OOP Keywords

height = 175 cm
weight = 76 kg

Properties

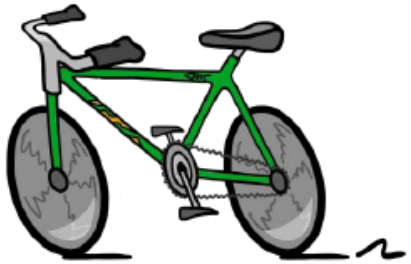
velocity = 200 m/s
brakes = 2

walk()
speak()

Methods

stop()
move()

ride(BikeObj)



Man Object

Bike Object

Class

- **A class is a template definition of an object's properties and methods.**

```
class Human:
    pass
```

Object

- **An Object is an instance on a Class.**

```
class Human:
      pass

man = Human()
```

Human Class

Man Object

Constructor

- is a method called at the moment an object is instantiated.

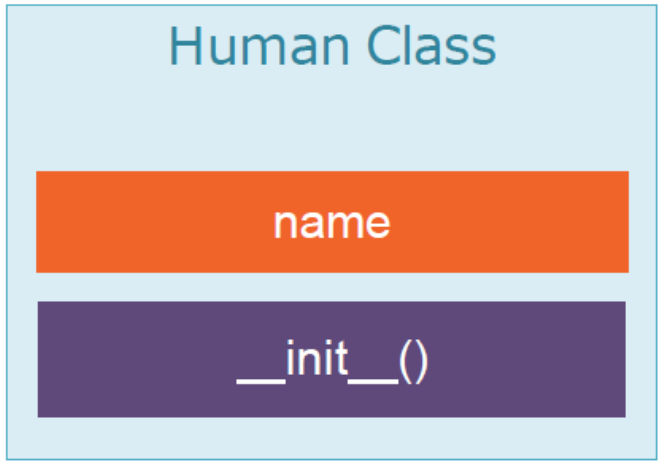
```
class Human:


      def__init_(self):
          print("Hi there")


man = Human()
```

```
Output:

Hi there
```

Instance Variable

- Is an object characteristic. such as name.

```
class Human:

    def __init_(self, name):
            self.name = name

man = Human("Ahmed")
```



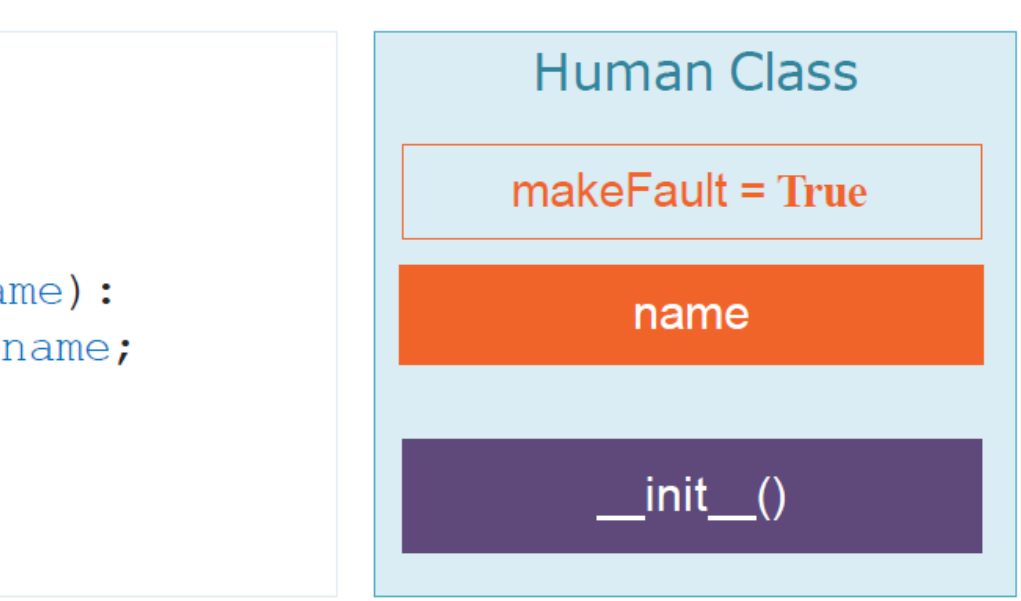
Name is Ahmed

Man Object

Class Variable

- is the variable that shared by all instances.

```
class Human:

        makeFault = True

        def__init_(self, name):
                self.name = name;

man = Human("Ahmed")
man2 = Human("Mohamed")
```



Name is Ahmed

He makes faults

Name is Mohamed

He makes faults

Class Variable

```
class Human:
        faults = 0
        def__init_(self, name):
                self.name = name;

man = Human("Ahmed")
man2 = Human("Mohamed")

man.faults = 1
print("Man :", man.faults)
print("Man 2:", man2.faults)
print("Human:", Human.faults)
Human.faults = 2
print("Man 2:", man2.faults)
print("Human:", Human.faults)
print("Man :", man.faults)
```

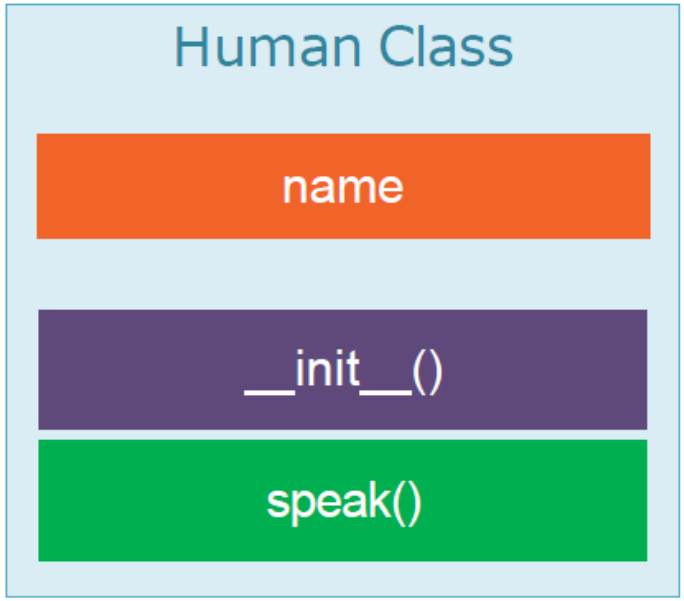
Output:

```
Man : 1

Man2 : 0

Human : 0

Man2 : 2

Human : 2

Man : 1
```

Instance Method

- **is an object capability(action), such as walk.**

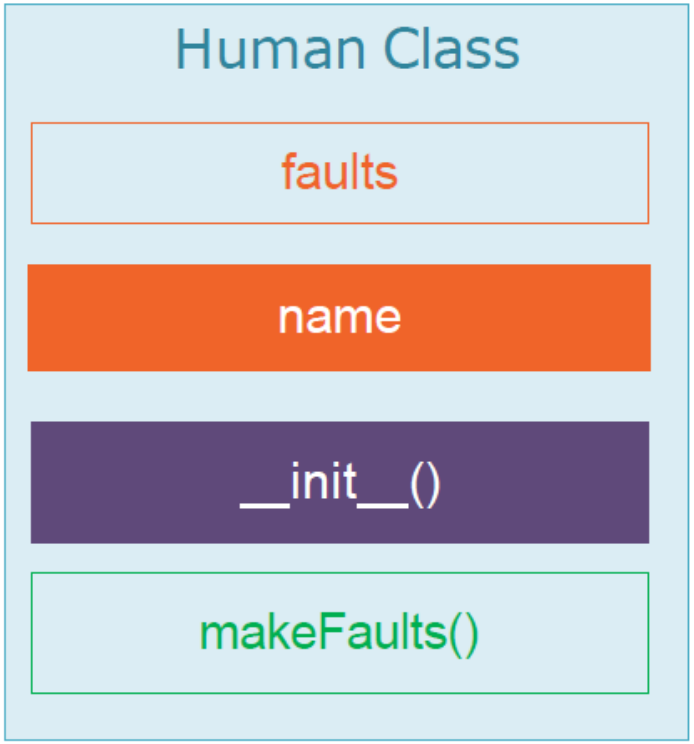
```
class Human:
    def_init_(self, name):
            self.name = name

    def speak(self):
        print("My Name is "+self.name)

man = Human("Ahmed")
man.speak()
```



Class Method

- is a method that shared by all instances of the Class

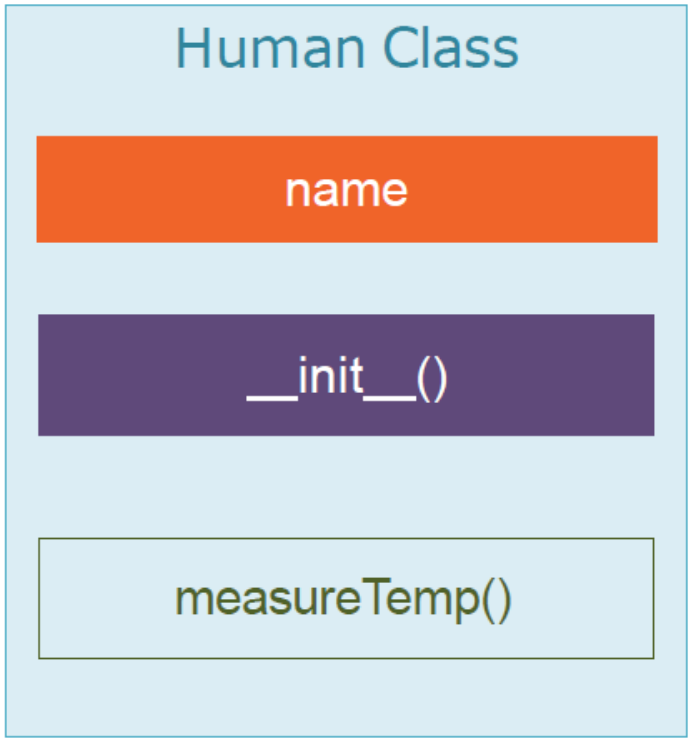
```
class Human:
    faults=0
    def_init_(self, name):
        self.name = name
    @classmethod
    def makeFaults(cls):
        cls.faults +=1
        print(cls.faults)
Human.makeFaults() #1
man = Human("Ahmed")
man.makeFaults()    #2
```



Static Method

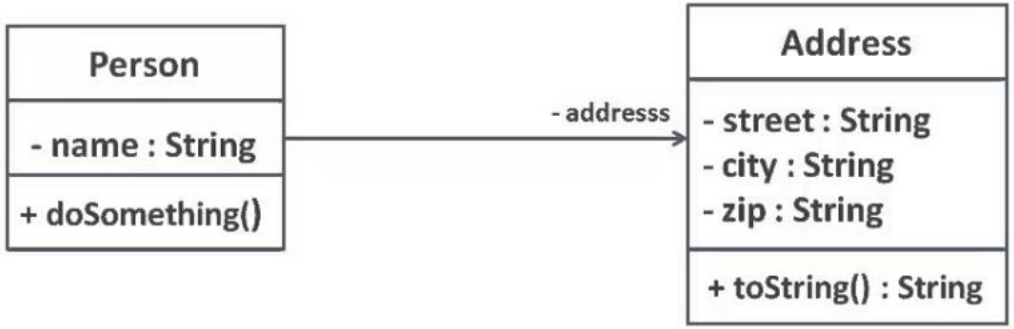
- is a normal function that have logic that related to the Class

```
class Human:
    def_init_(self, name):
        self.name = name

    @staticmethod
    def measureTemp(temp):
        if (temp == 37):
            return "Normal"
        return "Not Normal"

Human.measureTemp(38) # Not Normal
```

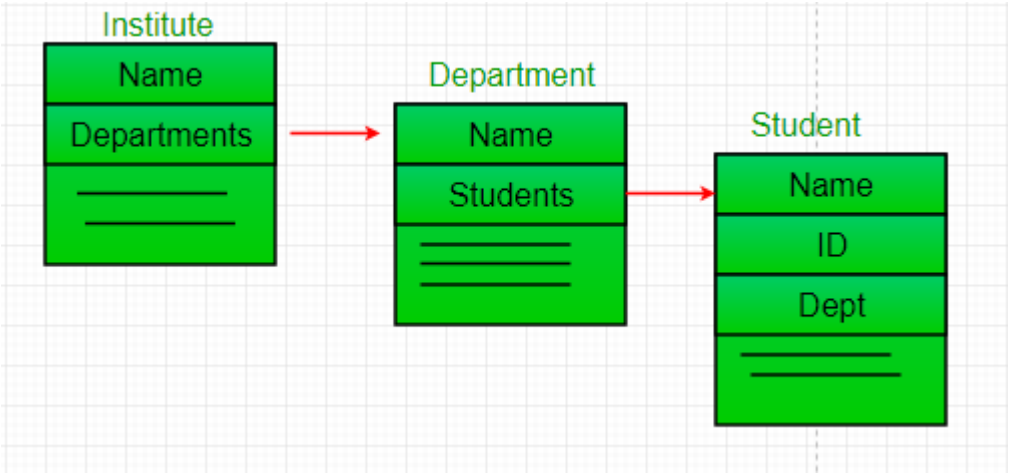


Class Relationships

- **Association :**
  - **object use object**

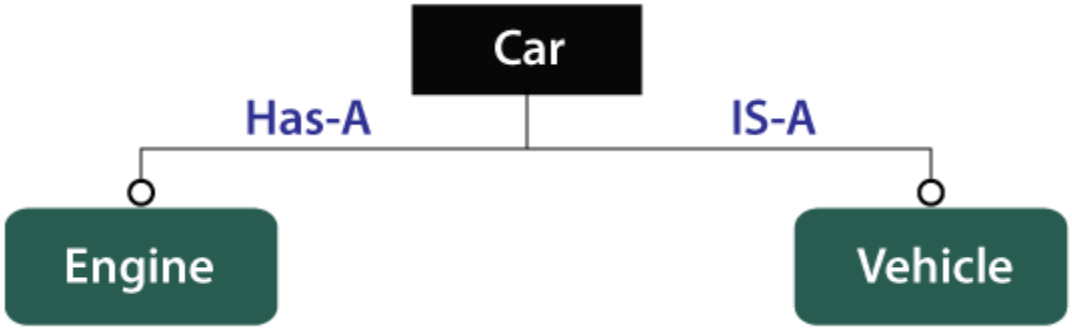


- **Aggregation**
  - **object contain object**



Class Relationships

- **Composition**
  - object **consist** of group of object
- **Inheritance**
  - object **is a** object



Exercises