In [ ]:
```
Object --> Object is an Instance of a class.
```

In [8]:
```
class Animal:
    # class Attribute(all instance can access)
    color = "black"

    # constructor
    def __init__(self, name):
        # instances attribute
        self.name = name

    def make_sound(self):
        return f"{self.name} makes a sound"

    def get_animal_count(self):
        count = 100
        return count

    @classmethod
    def get_color(cls):
        return cls.color


animal_obj = Animal("Lion")
print(animal_obj.color)
print(animal_obj.make_sound())
print(animal_obj.get_animal_count())
print(Animal.get_color())
```

```
black
Lion makes a sound
100
black
```

In [ ]:
```
implicit ---> a program is handle internally
explicit ---> a instruction is handle by externally
```

In [ ]:
```
# exercise

#write a five program using as object of the class
```

Inheritance

Definition

The Process of inherting the property of parent to child class is called the Inheritance

Type of Inheritance

In [ ]:
```
# Single Inheritance
# Multiple Inheritance
# Multi-level Inheritance
```

```
# Hierarchical Inheritance
# Hybrid Inheritance
```

Single Inheritance

In [21]:
```
class Dog(Animal):

    def __init__(self,name, age):
        super().__init__(name) # call parent constructor
        self.age = age

    def fetch(self):
        return f"{self.name} fetch all ball"


my_dog = Dog("Rex",3)
print(my_dog.__dict__) # magic method
print(my_dog.__) # magic functions
print(my_dog.fetch())
print(my_dog.make_sound())
print(my_dog.color)
```

```
{'name': 'Rex', 'age': 3}
Rex fetch all ball
Rex makes a sound
black
black
```

In [23]:
```
# Multiple Inheritance

class Fly:
    def fly(self):
        return f"I can fly"

class Swim:
    def swim(self):
        return f"I can swim"

class Duck(Fly,Swim):
    def make_sound(self):
        return f"I can swim and fly"

d = Duck()
print(d.fly())
print(d.swim())
print(d.make_sound())
```

```
I can fly
I can swim
I can swim and fly
```

In [24]:
```
# Multi-level Inheritance

class Fly:
    def fly(self):
        return f"I can fly"

class Swim(Fly):
    def swim(self):
        return f"I can swim"
```

```
class Duck(Swim):
    def make_sound(self):
        return f"I can swim and fly"

d1 = Duck()
print(d1.fly())
print(d1.swim())
print(d1.make_sound())
```

```
I can fly
I can swim
I can swim and fly
```

In [26]:
```
# Hierarchical Inheritance

class Fly:
    def fly(self):
        return f"I can fly"

class Swim(Fly):
    def swim(self):
        return f"I can swim"

class Duck(Fly):
    def make_sound(self):
        return f"I can swim and fly"

d1 = Duck()
print(d1.fly())
print(d1.make_sound())
```

```
I can fly
I can swim and fly
```

In [37]:
```
# Hybrid Inheritance

class Fly:
    def fly(self):
        return f"I can fly"

class Swim(Fly):
    def swim(self):
        return f"I can swim"

class Duck(Swim,Fly):
    def make_sound(self):
        return f"I can swim and fly"

class Bird(Swim):
    def jump(self):
        return f"I can jumpy"

d2 = Duck()
print(d2.fly())
print(d2.make_sound())
print(d2.make_sound())
```

```
I can fly
I can swim and fly
I can swim and fly
```

In [ ]:
```
# MRO
```