# Inheritance

Parent

Child –I Child- II

## Type:

1. Single
2. Hierarchical
3. Multiple
4. Multi-level
5. Hybrid

**Example** <https://github.com/bsdharshini/OOPS-2/blob/main/a_inherit_exmple.py>

**Vehicle** [Fuel, Model, Wheel] **Common attr**

**Car Bike Bus**

A/c type top speed seat

Sunroof Torque no.of.tyres **[specific attr]**

## Super() function/method:

* This fn bring down the parent attr and method to child class
* So child class have full access in spite of the modifier

Eg: <https://github.com/bsdharshini/OOPS-2/blob/main/c_super.py>

## Multiple Inheritance:

Inherit property from multiple parents

**Method resolution order:**

Eg: <https://github.com/bsdharshini/OOPS-2/blob/main/f_multiple_inheritance.py>

# Polymorphism:

Ability of the method/thing to be in different form.

Python does not support method overloading by default

## Method overriding:

* If wanted to change or modify parent class method in child class then use method overriding
* Once the parent class method is overridden by child class using object then the parent class cannot be accessed separately
* Name and parameter should be same

Eg: <https://github.com/bsdharshini/OOPS-2/blob/main/d_method_overriding.py>

## Operator overloading:

1. Some operator does 2 or more job
2. Eg: 1+4 = 5 or ‘a’+ ‘b’ = ab. Does both addition and concatation
3. Eg: list = [1,2,3], len(list) or s= ‘sha’, len(s)

Refer to python official doc for operator overloading fn

Eg: <https://github.com/bsdharshini/OOPS-2/blob/main/g_operator_overloading.py>

# Object class

Every class in python by default inherits from object class

**Methods provided by object class:**

1. \_\_new\_\_ = create an object. We will not override this method
2. \_\_init\_\_ = Initialize the object. Will be override
3. \_\_str\_\_ = Change the print messageCan be overridden.

Eg: <https://github.com/bsdharshini/OOPS-2/blob/main/e_object_class__str__.py>