Full stack web development using python

Inheritance

Saurabh Shukla (MySirG)

Agenda

1. Inheritance
2. What is inheritance?
3. How to extend a class?
4. Example
5. Types of Inheritance

Inheritance

The main concept of OOP is encapsulation, which is achieved by class.

The intent of OOP is incomplete without the concept of Inheritance

Inheritance

Inheritance is another key concept of OOP.

Reuse of code is major aspect of OOP, and can be achieved through Inheritance.

What is Inheritance?

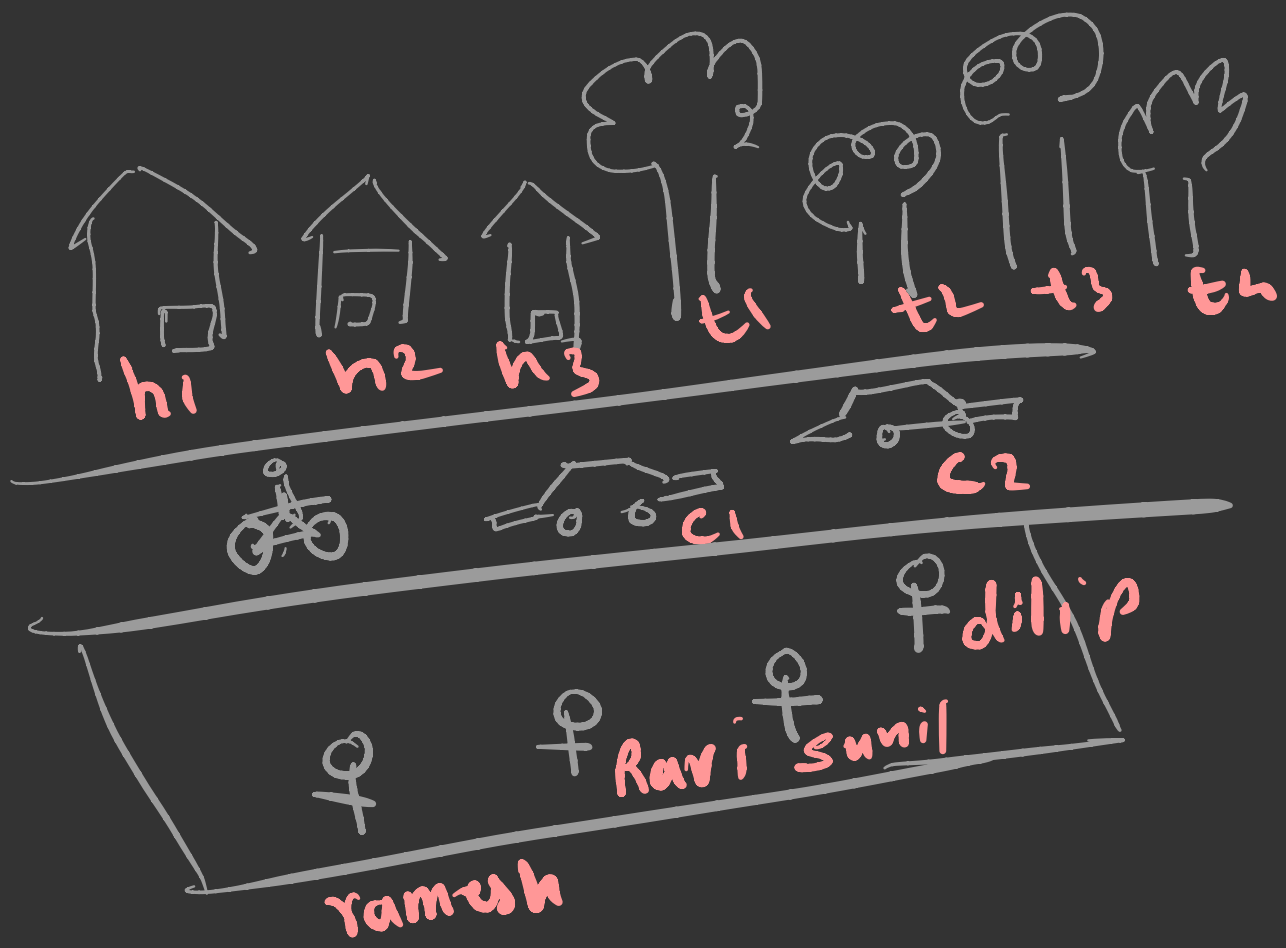
Defining a new class by extending the features of an existing class is called Inheritance.

Common noun Tree, House Person, Car

Proper noun Ramesh, Ravi, Sunil, dilip

t1, t2, t3, t4, h1, h2, h3

C1, C2



Proper noun
(Object)

Common Noun
(class)

Suppose we have a Person class

Person (name, age)

now we want to make a class Student.

- Defining Student from the scratch is rework and increase the cost of development.
- In real world every student is a Person.
- Person is already defined, we can reuse it to define Student.
- By adding attributes like rollno we can make a Person Student.

You may think of adding new attributes to the Person class but this will loose identity of Person.

The best way to do modification in Person to make it Student without disturbing Person class, is inheritance

How to extend a class ?

```
class Person:
    # attributes
```

Base class
Super class
Parent class

```
class Student(Person):
    # new attributes
```

Derived class
subclass
child class

Example

```
class Person :
    def __init__(self, n, a):
        self.name = n
        self.age = a
    def showName(self):
        print(self.name)
    def showAge(self):
        print(self.age)
```

```
class Person :
    def __init__(self,n,a):
        self.name = n
        self.age = a
    def showName(self):
        print(self.name)
    def showAge(self):
        print(self.age)

class Student (Person):

    def setRollno(self,r):
        self.rollno = r
    def showRollno(self):
        print(self.rollno)
```

```
class Person :
    def __init__(self,n,a):
        self.name = n
        self.age = a
    def showName(self):
        print(self.name)
    def showAge(self):
        print(self.age)

class Student (Person):

    def setRollno(self,r):
        self.rollno = r
    def showRollno(self):
        print(self.rollno)
```

```
S1 = Student("Rahul", 20)
```

↑

You cannot pass rollno.

```
class Person :
    def __init__(self,n,a):
        self.name = n
        self.age = a
    def showName(self):
        print(self.name)
    def showAge(self):
        print(self.age)

class Student (Person):
    def __init__(self, r):
        self.rollno = r

    def setRollno(self, r):
        self.rollno = r
    def showRollno(self):
        print(self.rollno)
```

Let's define __init__() method in Student class as well.

```
class Person :
    def __init__(self,n,a):
        self.name = n
        self.age = a
    def showName(self):
        print(self.name)
    def showAge(self):
        print(self.age)

class Student (Person):
    def __init__(self, r):
        self.rollno = r

    def setRollno(self, r):
        self.rollno = r
    def showRollno(self):
        print(self.rollno)
```

```
S1 = Student(205)
```

↑

you can pass only rollno

```
class Person :
    def __init__(self,n,a):
        self.name = n
        self.age = a
    def showName(self):
        print(self.name)
    def showAge(self):
        print(self.age)
```

```
S1= Student("Rahul",20,205)
S1.showRollno()
S1.showName()
S1.showAge()
```

```
class Student(Person):
    def __init__(self,n,a,r):
        super().__init__(n,a)
        self.rollno = r

    def setRollno(self,r):
        self.rollno=r
    def showRollno(self):
        print(self.rollno)
```

You can also write

```
Person.__init__(self,n,a)
```

When an attribute is accessed via derived class object, if attribute is not found in derived class, the search proceeds to look in the base class.

Types of Inheritance

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

Single Inheritance

```
class A:
    ==

class B(A):
    ==
```

Multilevel Inheritance

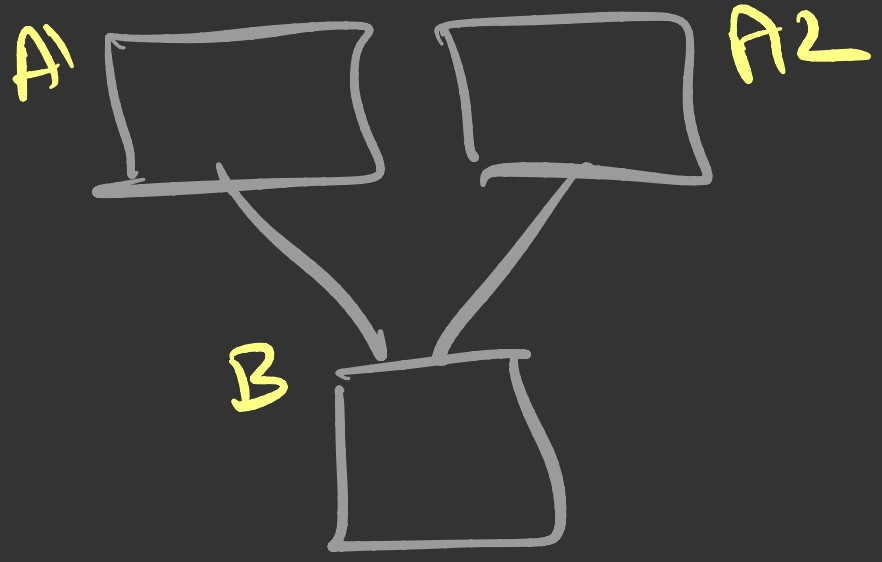
```
class A:
    ==

class B(A):
    ==

class C(B):
    ==
```

Multiple Inheritance

```
class A1:
    ...
class A2:
    ...
```



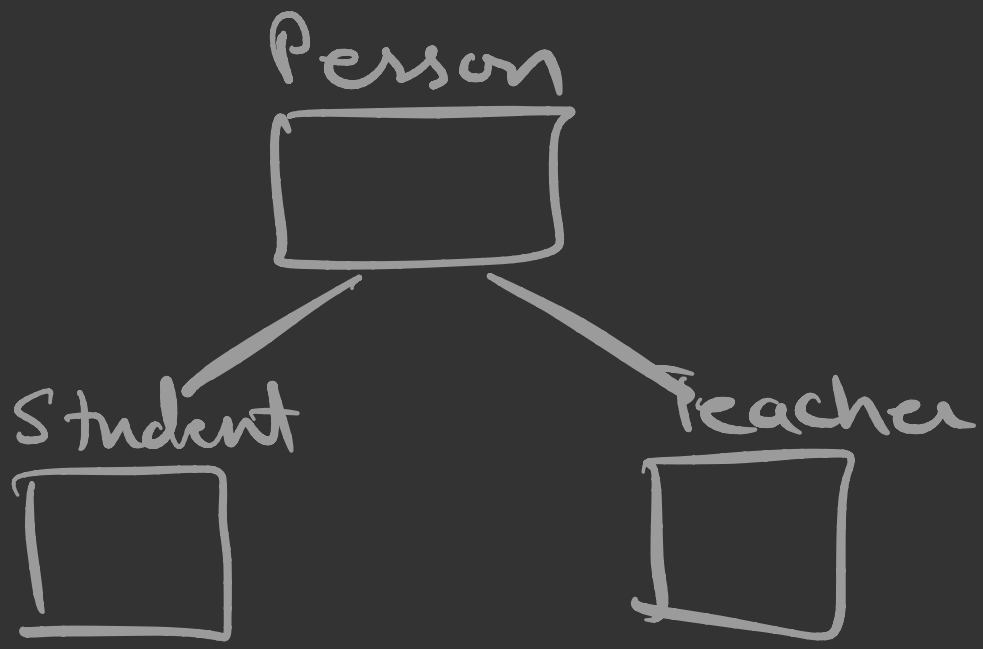

```
class B(A1, A2):
    ...
```

Hierarchical Inheritance

```
class A:
    ...

class B1(A):
    ...

class B2(A):
    ...
```



Hybrid Inheritance

```
class A :

class B1(A):

class B2(A):

class C(B1, B2):
```