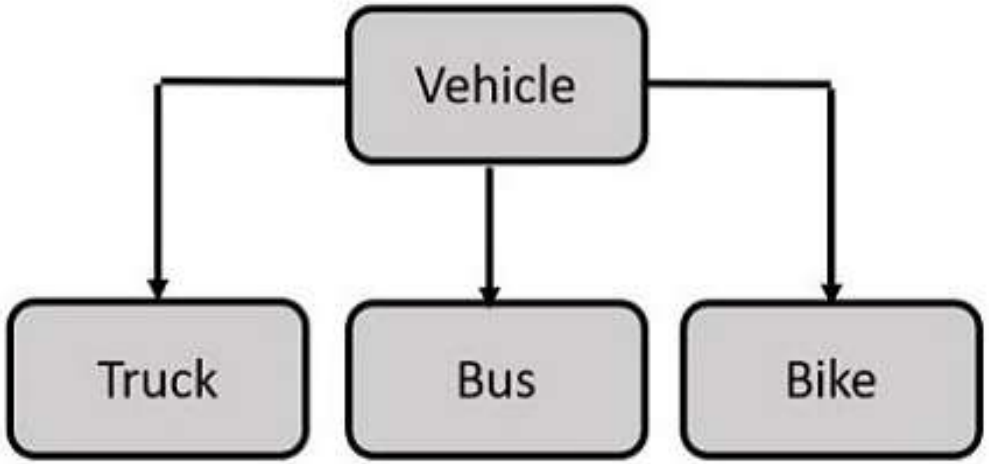
Python Inheritance

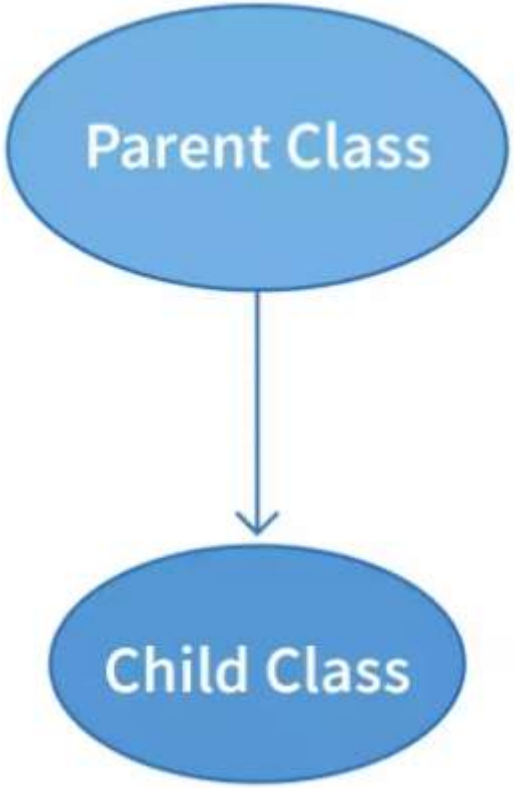
11.4.1.4 create a class hierarchy;

11.4.2.2 explain the concept of inheritance with examples;

11.4.3.2 solve applied problems of various subject areas.

Inheritance allows us to define a class that inherits all the methods and properties from another class





Parent class is the class being inherited from, also called base class.

Child class is the class that inherits from another class, also called derived class.

Python Inheritance Syntax

```
# define a superclass
class super_class:
    # attributes and method definition

# inheritance
class sub_class(super_class):
    # attributes and method of super_class
    # attributes and method of sub_class
```

Here, we are inheriting the sub_class from the super_class.

Example: Python Inheritance

```
class Animal:

    # attribute and method of the parent class
    name = ""

    def eat(self):
        print("I can eat")

# inherit from Animal
class Dog(Animal):

    # new method in subclass
    def display(self):
        # access name attribute of superclass using self
        print("My name is ", self.name)

# create an object of the subclass
labrador = Dog()

# access superclass attribute and method
labrador.name = "Rohu"
labrador.eat()

# call subclass method
labrador.display()
```

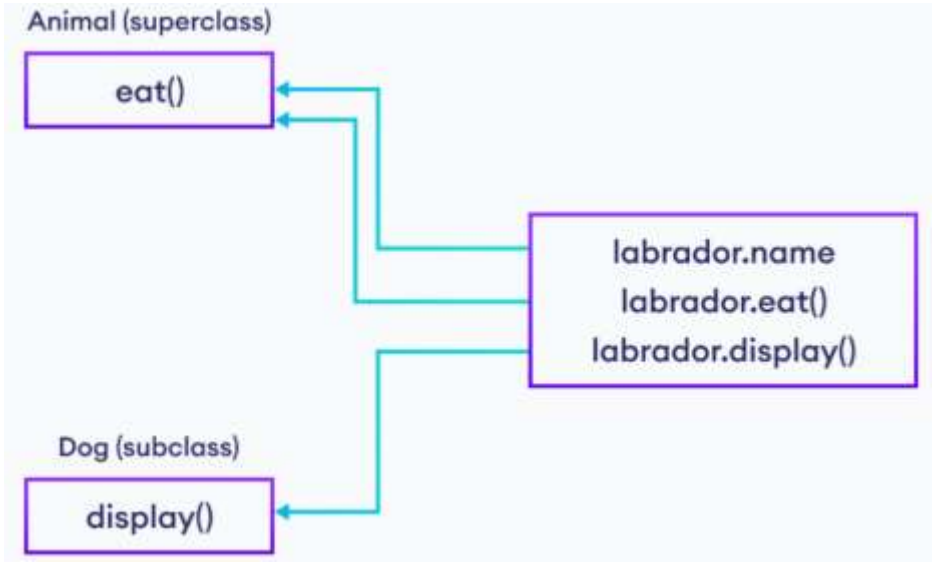
In the example, we have derived a subclass Dog from a superclass Animal.

we are using labrador (object of Dog) to access name and eat() of the Animal class.

This is possible because the subclass inherits all attributes and methods of the superclass.

Output

```
I can eat
My name is  Rohu
```



Method Overriding in Python Inheritance

In the previous example, we see the object of the subclass can access the method of the superclass. However, what if the same method is present in both the superclass and subclass?

In this case, the method in the subclass overrides the method in the superclass. This concept is known as method overriding in Python.

```
class Animal:

    # attributes and method of the parent class
    name = ""

    def eat(self):
        print("I can eat")

# inherit from Animal
class Dog(Animal):

    # override eat() method
    def eat(self):
        print("I like to eat bones")

# create an object of the subclass
labrador = Dog()

# call the eat() method on the labrador object
labrador.eat()
```

In the example, the same method eat() is present in both the Dog class and the Animal class.
Now, when we call the eat() method using the object of the Dog subclass, the method of the Dog class is called.

This is because the eat() method of the Dog subclass overrides the same method of the Animal superclass.

Output

```
I like to eat bones
```

The super() Function in Inheritance

if we need to access the superclass method from the subclass, we use the super() function. For example,

```
class Animal:

    name = ""

    def eat(self):
        print("I can eat")

# inherit from Animal
class Dog(Animal):

    # override eat() method
    def eat(self):

        # call the eat() method of the superclass using super()
        super().eat()

        print("I like to eat bones")

# create an object of the subclass
labrador = Dog()

labrador.eat()
```

Output

```
I can eat
I like to eat bones
```

https://www.w3schools.com/python/python_inheritance.asp

Task 1

- Define a base class **Person** with common attributes. From this base class, derive two classes: Student and Professor.

- The **Student class** should include specific attributes for **the year of study and major.** Equip it with methods to display student-specific information, including their year and major.

- The **Professor class**, also derived from Person, should introduce attributes for the **subject taught and the number_of_publications**. It should have methods to display professor-specific information, such as their subject and publication count.

Task 2

- Define an abstract **class Shape** with an abstract method **area** for calculating the shape's area.
- Create a **Rectangle class** derived from Shape, implementing the area method using rectangle attributes like **length and width** to calculate the area.
- Introduce a **Circle class** from Shape, implementing the area method with a **radius** attribute to compute the circle's area.
- Lastly, develop a **Triangle class** from Shape, implementing the area method using attributes such **as base and height** to determine the triangle's area.

Task 3

- Define a base class **Product** with **name and price** attributes, including methods **to display** and **update** the price.
- Create a **Book class** derived from Product, adding **author and ISBN** attributes, with methods **to show** book details and **update the author.**
- Introduce a **Food class** from Product, incorporating **expiration_date and ingredients**, with methods for **displaying food details** and **updating the expiration date**.
- Lastly, form a **Clothing** class from Product, with **size and material** attributes, including methods to **present** clothing details and **adjust the size.**

Task 4

- Define a base class Employee with basic attributes **name, id_number** and method **display_info** for workers.
- Create derived classes **Manager, Engineer, and Intern**, each extending the base class with new attributes and methods specific to each role.
- **Manager:** Add additional attributes **department and num_of_subordinates**. Implement a method **display_info** to display information about the manager, and a method **manage** to demonstrate managerial activities.
- **Engineer:** Include attributes **field_of_expertise and years_of_experience.** Override the method **display_info** to show information about the engineer, and implement a method **develop** to showcase engineering work.
- **Intern:** Introduce attributes **mentor and duration**. Override the method **display_info** to present information about the intern, and add a method **learn** to illustrate the intern's learning process.