Object-Oriented Programming (OOPs) - Detailed Notes

What is OOPs?

Object-Oriented Programming (OOPs) is a programming style based on objects and classes. It helps to make the code reusable, modular, and easier to maintain.

1. Class

A class is a blueprint for creating objects. It defines a set of attributes and methods that the created objects will have.

Example:

```
class Car:
    def __init__(self, brand, model):
        self.brand = brand
        self.model = model
```

2. Object

An object is an instance of a class. It represents a real-world entity.

Example:

```
car1 = Car("Toyota", "Camry")
```

3. Encapsulation

Encapsulation is the technique of hiding internal data by wrapping it inside a class. It helps in data protection and abstraction.

Example:

```
class BankAccount:
    def __init__(self):
```

```
        self.__balance = 0

    def deposit(self, amount):
        self.__balance += amount
```

4. Abstraction

Abstraction hides complex implementation and shows only the necessary details.

Example:

```
from abc import ABC, abstractmethod

class Animal(ABC):
    @abstractmethod
    def sound(self):
        pass
```

5. Inheritance

Inheritance allows a class (child) to acquire properties and methods of another class (parent).

Example:

```
class Animal:
    def speak(self):
        print("Animal speaks")

class Dog(Animal):
    def bark(self):
        print("Dog barks")
```

6. Polymorphism

Polymorphism means the same method name behaves differently based on the object.

Example:

```
class Bird:
    def sound(self):
        print("Chirp")


class Dog:
    def sound(self):
        print("Bark")


def animal_sound(animal):
    animal.sound()
```

Benefits of OOPs:

- Code reusability using inheritance.
- Better organization using classes and objects.
- Easier maintenance and debugging.
- Real-world modeling becomes simpler.