Python Inheritance Basics

Inheritance is one of the key features of object-oriented programming (OOP) that allows a class (child or subclass) to inherit attributes and methods from another class (parent or superclass). This allows for code reuse and can help create a hierarchy of classes.

Key Concepts:

- **Parent Class** (or **Superclass**): The class whose properties and methods are inherited by another class.
- **Child Class** (or **Subclass**): The class that inherits the properties and methods of another class.
- **Method Overriding**: A subclass can provide its own implementation of methods defined in the parent class.

Basic Syntax for Inheritance

To inherit from a parent class, the child class is defined by specifying the parent class in parentheses.

```
class ParentClass:
    # Parent class code
    pass

class ChildClass(ParentClass):
    # Child class code
    pass
```

```
# Parent class
class Animal:
    def __init__(self, name):
        self.name = name

    def speak(self):
        print(f"{self.name} makes a sound.")

# Child class inheriting from Animal
class Dog(Animal):
    def __init__(self, name, breed):
        super().__init__(name)  # Call the parent class constructor
        self.breed = breed

    def speak(self):
        print(f"{self.name} barks.")
```

```
# Creating an object of the Dog class
dog = Dog("Buddy", "Golden Retriever")
dog.speak()  # Output: Buddy barks.
```

Output:-

Buddy barks.

Explanation:

- `Dog` inherits from `Animal`.
- `Dog` has a method `speak` that overrides the `speak` method in `Animal`.

Inheritance with Method Overriding

A subclass can override methods from the parent class to provide its own implementation.

```
# Parent class
class Animal:
    def sound(self):
        print("Some generic animal sound")

# Child class
class Dog(Animal):
    def sound(self):
        print("Woof! Woof!")

# Child class
class Cat(Animal):
    def sound(self):
        print("Meow!")

# Creating objects of the subclasses
dog = Dog()
cat = Cat()

dog.sound()  # Output: Woof! Woof!
cat.sound()  # Output: Meow!
```

Output:-

Woof! Woof!

Meow!

Explanation:

- Both `Dog` and `Cat` override the `sound` method from the parent class `Animal`.

Using `super()` in Inheritance

`super()` is used to call methods from the parent class, which is useful when you want to extend the functionality of a parent method rather than override it completely.

```
# Parent class
class Person:
    def __init__(self, name):
        self.name = name

    def introduce(self):
        print(f"Hello, my name is {self.name}.")

# Child class
class Student(Person):
    def __init__(self, name, course):
        super().__init__(name)  # Call the constructor of the parent class
        self.course = course

    def introduce(self):
        super().introduce()  # Call the parent class method
        print(f"I am studying {self.course}.")

# Creating an object of the Student class
student = Student("Alice", "Computer Science")
student.introduce()
# Output:
# Hello, my name is Alice.
# I am studying Computer Science.
```

Output:-

Hello, my name is Alice.

I am studying Computer Science.