Inheritance :

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

    def speak(self):
        pass

# Derived class (Child class) inheriting from Animal
class Dog(Animal):
    def __init__(self, name, breed):
        super().__init__(name, species="Dog")
        self.breed = breed

    def speak(self):
        return f"{self.name} barks!"

# Another derived class (Child class) inheriting from Animal
class Cat(Animal):
    def __init__(self, name, color):
        super().__init__(name, species="Cat")
        self.color = color

    def speak(self):
        return f"{self.name} meows!"

dog = Dog("Buddy", "Golden Retriever")
cat = Cat("Whiskers", "Gray")
print(f"{dog.name} is a {dog.species} of breed {dog.breed}. {dog.speak()}")
print(f"{cat.name} is a {cat.species} with {cat.color} fur. {cat.speak()}")
```

Abstraction:

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

    def introduce(self):
        print(f"Hi, I'm {self.name} and I am {self.age} years old.")
person1 = Person("Alice", 30)
person2 = Person("Bob", 25)

print(person1.name)  # Output: Alice
print(person2.age)   # Output: 25
person1.introduce()  # Output: Hi, I'm Alice and I am 30 years old.
person2.introduce()  # Output: Hi, I'm Bob and I am 25 years old.
```

Encapsulation:

```
class Student:
    def __init__(self, name, age):
        self.__name = name  # Private attribute
        self.__age = age    # Private attribute

    def get_name(self):
        return self.__name

    def set_name(self, name):
        if len(name) > 0:
            self.__name = name
    def get_age(self):
        return self.__age

    def set_age(self, age):
        if age >= 0:
            self.__age = age

    def display_info(self):
        print(f"Name: {self.__name}, Age: {self.__age}")
student1 = Student("Alice", 20)
student1.display_info()
student1.set_name("Bob")
student1.set_age(22)
student1.display_info()
```

Polymorphism:

```
class Shape:
    def area(self):
        pass

class Circle(Shape):
    def __init__(self, radius):
        self.radius = radius

    def area(self):
        return 3.14 * self.radius * self.radius

class Rectangle(Shape):
    def __init__(self, width, height):
        self.width = width
        self.height = height

    def area(self):
        return self.width * self.height

def print_area(shape):
    print(f"Area: {shape.area()}")

circle = Circle(5)
rectangle = Rectangle(4, 6)

print("Circle:")
print_area(circle)

print("Rectangle:")
print_area(rectangle)
```