```
class Book:
    def __init__(self, title, author, year):
        self.title = title
        self.author = author
        self.year = year

    def get_details(self):
        return f"Title: {self.title}, Author: {self.author}, Year Published: {self.year}"


book = Book("1984", "George Orwell", 1949)
print(book.get_details())
```

```
Title: 1984, Author: George Orwell, Year Published: 1949
```

```
class Animal:
    def speak(self):
        print("Some generic sound")

class Dog(Animal):
    def speak(self):
        print("Woof")

dog = Dog()
dog.speak()
```

```
Woof
```

```
class BankAccount:
    def __init__(self, balance=0):
        self.__balance = balance  # Private variable

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

    def withdraw(self, amount):
        if amount <= self.__balance:
            self.__balance -= amount
        else:
            print("Insufficient balance")

    def get_balance(self):
        return self.__balance


account = BankAccount(100)
account.deposit(50)
account.withdraw(120)
print(account.get_balance())
```

```
30
```

```
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, length, width):
        self.length = length
        self.width = width

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


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

print("Circle area:", circle.area())
print("Rectangle area:", rectangle.area())
```

```
print("Rectangle area:", rectangle_area())
```

```
Circle area: 78.5
Rectangle area: 24
```

```
class MathOperations:
    @staticmethod
    def multiply(x, y):
        return x * y


print(MathOperations.multiply(4, 5))
```

```
20
```

```
class Calculator:
    def add(self, *args):
        if len(args) == 2:
            return sum(args)
        elif len(args) == 1 and isinstance(args[0], list):
            return sum(args[0])


calc = Calculator()
print(calc.add(2, 3))
print(calc.add([1, 2, 3]))
```

```
5
6
```

```
class Employee:
    company_name = "TechCorp"

    def __init__(self, name, position):
        self.name = name
        self.position = position

    def get_details(self):
        return f"{self.name} works as {self.position} at {self.company_name}"


emp1 = Employee("Alice", "Developer")
emp2 = Employee("Bob", "Manager")
print(emp1.get_details())
print(emp2.get_details())
```

```
Alice works as Developer at TechCorp
Bob works as Manager at TechCorp
```

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

    def __repr__(self):
        return f"Car(make={self.make}, model={self.model}, year={self.year})"


car = Car("Toyota", "Corolla", 2020)
print(car)
```

```
Car(make=Toyota, model=Corolla, year=2020)
```

```
class Vehicle:
    def __init__(self, make, model):
        self.make = make
        self.model = model

class Truck(Vehicle):
    def __init__(self, make, model, payload_capacity):
        super().__init__(make, model)
        self.payload_capacity = payload_capacity

    def get_details(self):
        return f"Make: {self.make}, Model: {self.model}, Payload Capacity: {self.payload_capacity} kg"
```

```
truck = Truck("Ford", "F-150", 3000)
print(truck.get_details())
```

```
Make: Ford, Model: F-150, Payload Capacity: 3000 kg
```