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

    def introduce(self):
        print(f"Hi my name is {self.name} and I am {self.age} years old")

student1 = Student("John", 20)
student1.introduce()
```

```
Hi my name is John and I am 20 years old
```

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

    def display(self):
        print(f"Book Title: {self.title}")
        print(f"Author: {self.author}")

book1 = Book("Python Programming", "John Doe")
book1.display()
```

```
class Vehicle:
    def start(self):
        print("Vehicle starting...")

class Car(Vehicle):
    def start(self):
        print("Car starting...")

car = Car()
car.start()
```

```
Car starting...
```

```
class Animal:
    def sound(self):
        return "some generic animal sound"

class Dog(Animal):
    def sound(self):
        return "bark"

dog = Dog()
print(dog.sound())
```

```
bark
```

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

    def get_age(self):
        return self._age

    def set_age(self, age):
        if age >= 0:
            self._age = age
        else:
            raise ValueError("Age cannot be negative")

person = Person(30)
print(person.get_age())
person.set_age(35)
print(person.get_age())
```

```
30
35
```

```
class Shape:
    def area(self):
        return "Area not defined"

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

    def area(self):
        return 3.14 * self.radius ** 2

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())
print(rectangle.area())
```

```
78.5
24
```

```
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("Tesla", "Model S", 2022)
print(car)
```

```
Car(make='Tesla', model='Model S', year=2022)
```

```
class Calculator:
    def add(self, *args):
        return sum(args) if args else 0

calc = Calculator()
print(calc.add(5))
print(calc.add(1, 2, 3, 4))
```

```
5
10
```

```
class Employee:
    company = "TechCorp"

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

    def display(self):
        print(f"Name: {self.name}, Company: {self.company}")

emp1 = Employee("Alice")
emp2 = Employee("Bob")

emp1.display()
emp2.display()
```

```
Name: Alice, Company: TechCorp
Name: Bob, Company: TechCorp
```

```
class BankAccount:
    interest_rate = 0.03  # Static variable

    def __init__(self, balance):
        self.balance = balance

    @classmethod
    def create_account(cls, balance):
        return cls(balance)

    @staticmethod
```

```
        def interest_rate():
            return BankAccount.interest_rate

account = BankAccount.create_account(1000)
print(account.balance)
print(BankAccount.interest_rate())
```

```
1000
<function BankAccount.interest_rate at 0x7c9bc64c1360>
```