Question 1: Fixing a Rectangle-Square Hierarchy

Refactor the `Square` class to adhere to the Liskov Substitution Principle. Ensure that a `Square` object can be used wherever a `Rectangle` object is expected without causing unexpected behavior

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

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

    def set_width(self, width):
        self.width = width

    def set_height(self, height):
        self.height = height

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

class Square(Shape):
    def __init__(self, side):
        self.side = side

    def set_side(self, side):
        self.side = side

    def area(self):
        return self.side * self.side

# Testing
shapes = [Rectangle(4, 5), Square(4)]
for shape in shapes:
    print(f"{shape.__class__.__name__} Area: {shape.area()}")
```

```
Rectangle Area: 20
Square Area: 16
```

Question 2: Fixing a Bird-Penguin Hierarchy

Refactor the `Bird` and `Penguin` classes to adhere to the Liskov Substitution Principle. Ensure that a `Penguin` object can be used wherever a `Bird` object is expected without causing unexpected behavior.

```
class Bird:
    def make_sound(self):
        print("Chirp!")

class FlyingBird(Bird):
    def fly(self):
        print("Flying in the sky")

class NonFlyingBird(Bird):
    pass  # Penguins and similar birds will inherit this

class Penguin(NonFlyingBird):
    def swim(self):
        print("Swimming in water")

# Testing
birds = [FlyingBird(), Penguin()]

for bird in birds:
    bird.make_sound()
    if isinstance(bird, FlyingBird):
        bird.fly()
    elif isinstance(bird, Penguin):
        bird.swim()
```

```
Chirp!
Flying in the sky
Chirp!
Swimming in water
```

Question 3: Fixing a File Reader-Writer Hierarchy

Refactor the `FileReader` and `FileWriter` classes to adhere to the Liskov Substitution Principle. Ensure that a `FileWriter` object can be used wherever a `FileReader` object is expected without causing unexpected behavior.

```
class FileReader:
    def __init__(self, filename):
        self.filename = filename

    def read(self):
        with open(self.filename, 'r') as file:
            return file.read()

class FileWriter:
    def __init__(self, filename):
        self.filename = filename

    def write(self, content):
        with open(self.filename, 'w') as file:
            file.write(content)

# Testing
reader = FileReader("test.txt")
writer = FileWriter("test.txt")

writer.write("Hello, Liskov!")
print(reader.read())  # Should print "Hello, Liskov!"
```

```
Hello, Liskov!
```