Lecture

Inheritance

Methods - Syntax

Methods are inherited automatically

Inheritance

The subclass inherits the methods from the superclass

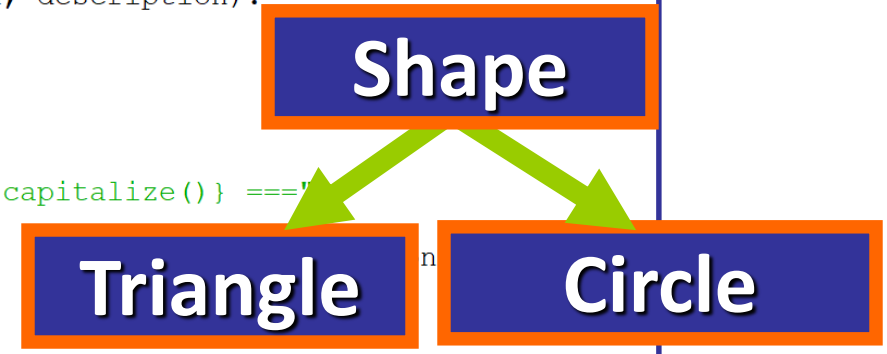
```
class Superclass:
    # Body

class Subclass(Superclass):
    # Body
```

Inheritance

```python
class Shape:

    def __init__(self, color, is_polygon, description):
        self.color = color
        self.is_polygon = is_polygon
        self.description = description

    def display_data(self):
        print(f"\n=== {self.description.capitalize()} ===")
        print("Color:", self.color)
        print("Is the shape a polygon?", "Yes" if self.is_polygon else "No")

class Triangle(Shape):

    def __init__(self, color, vertices, base, height):
        Shape.__init__(self, color, True, "Triangle")
        self.vertices = vertices
        self.base = base
        self.height = height

class Circle(Shape):

    def __init__(self, color, radius):
        Shape.__init__(self, color, False, "Circle")
        self.radius = radius
```

Inheritance

```python
class Shape:

    def __init__(self, color, is_polygon, description):
        self.color = color
        self.is_polygon = is_polygon
        self.description = description

    def display_data(self):
        print(f"\n=== {self.description.capitalize()} ===")
        print("Color:", self.color)
        print("Is the shape a polygon?",

class Triangle(Shape):

    def __init__(self, color, vertices, base, height):
        Shape.__init__(self, color, True, "Triangle")
        self.vertices = vertices
        self.base = base
        self.height = height

class Circle(Shape):

    def __init__(self, color, radius):
        Shape.__init__(self, color, False, "Circle")
        self.radius = radius
```



Inheritance

```python
class Shape:

    def __init__(self, color, is_polygon, description):
        self.color = color
        self.is_polygon = is_polygon
        self.description = description

    def display_data(self):
        print(f"\n=== {self.description.capitalize()} ===")
        print("Color:", self.color)
        print("Is the shape a polygon?", "Yes" if self.is_polygon else "No")

class Triangle(Shape):

    def __init__(self, color, vertices, base, height):
        Shape.__init__(self, color, True, "Triangle")
        self.vertices = vertices
        self.base = base
        self.height = height

class Circle(Shape):

    def __init__(self, color, radius):
        Shape.__init__(self, color, False, "Circle")
        self.radius = radius
```

Inheritance

```
triangle = Triangle("red", [(-2, 0), (2, 0), (0, 7)], 4, 7)
circle = Circle("blue", 6.3)

triangle.display_data()
circle.display_data()
```

Inheritance

```
triangle = Triangle("red", [(-2, 0), (2, 0), (0, 7)], 4, 7)
circle = Circle("blue", 6.3)

triangle.display_data()
circle.display_data()
```

Inheritance

```
triangle = Triangle("red", [(-2, 0), (2, 0), (0, 7)], 4, 7)
circle = Circle("blue", 6.3)

triangle.display_data()
circle.display_data()
```

```
=== Triangle ===
Color: red
Is the shape a polygon? Yes

=== Circle ===
Color: blue
Is the shape a polygon? No
```

Inheritance

```python
class Shape:

    def __init__(self, color, is_polygon, description):
        self.color = color
        self.is_polygon = is_polygon
        self.description = description

    def display_data(self):
        print(f"\n=== {self.description.capitalize()} ===")
        print("Color:", self.color)
        print("Is the shape a polygon?", "Yes" if self.is_polygon else "No")
```

```python
class Triangle(Shape):

    def __init__(self, color, vertices, base, height):
        Shape.__init__(self, color, True, "Triangle")
        self.vertices = vertices
        self.base = base
        self.height = height
```

```python
class Circle(Shape):

    def __init__(self, color, radius):
        Shape.__init__(self, color, False, "Circle")
        self.radius = radius
```

Inheritance

```python
class Shape:

    def __init__(self, color, is_polygon, description):
        self.color = color
        self.is_polygon = is_polygon
        self.description = description

    def display_data(self):
        print(f"\n=== {self.description.capitalize()} ===")
        print("Color:", self.color)
        print("Is the shape a polygon?", "Yes" if self.is_polygon else "No")
```

```python
class Triangle(Shape):

    def __init__(self, color, vertices, base, height):
        Shape.__init__(self, color, True, "Triangle")
        self.vertices = vertices
        self.base = base
        self.height = height
```

```python
class Circle(Shape):

    def __init__(self, color, radius):
        Shape.__init__(self, color, False, "Circle")
        self.radius = radius
```

Inheritance

```python
class Shape:

    def __init__(self, color, is_polygon, description):
        self.color = color
        self.is_polygon = is_polygon
        self.description = description

    def display_data(self):
        print(f"\n=== {self.description.capitalize()} ===")
        print("Color:", self.color)
        print("Is the shape a polygon?", "Yes" if self.is_polygon else "No")
```

```python
class Triangle(Shape):

    def __init__(self, color, vertices, base, height):
        Shape.__init__(self, color, True, "Triangle")
        self.vertices = vertices
        self.base = base
        self.height = height

class Circle(Shape):

    def __init__(self, color, radius):
        Shape.__init__(self, color, False, "Circle")
        self.radius = radius
```

Inheritance

```python
class Shape:

    def __init__(self, color, is_polygon, description):
        self.color = color
        self.is_polygon = is_polygon
        self.description = description

    def display_data(self):
        print(f"\n=== {self.description.capitalize()} ===")
        print("Color:", self.color)
        print("Is the shape a polygon?", "Yes" if self.is_polygon else "No")
```

```python
class Triangle(Shape):

    def __init__(self, color, vertices, base, height):
        Shape.__init__(self, color, True, "Triangle")
        self.vertices = vertices
        self.base = base
        self.height = height

class Circle(Shape):

    def __init__(self, color, radius):
        Shape.__init__(self, color, False, "Circle")
        self.radius = radius
```

Inheritance

```
triangle = Triangle("red", [(-2, 0), (2, 0), (0, 7)], 4, 7)
circle = Circle("blue", 6.3)

triangle.find_area()
circle.find_area()
```

Inheritance

```
triangle = Triangle("red", [(-2, 0), (2, 0), (0, 7)], 4, 7)
circle = Circle("blue", 6.3)

triangle.find_area()
circle.find_area()
```

Inheritance

```python
class Shape:

    def __init__(self, color, is_polygon, description):
        self.color = color
        self.is_polygon = is_polygon
        self.description = description

    def display_data(self):
        print(f"\n=== {self.description.capitalize()} ===")
        print("Color:", self.color)
        print("Is the shape a polygon?", "Yes" if self.is_polygon else "No")
```

```python
class Triangle(Shape):

    def __init__(self, color, vertices, base, height):
        Shape.__init__(self, color, True, "Triangle")
        self.vertices = vertices
        self.base = base
        self.height = height
```

```python
class Circle(Shape):

    def __init__(self, color, radius):
        Shape.__init__(self, color, False, "Circle")
        self.radius = radius
```

Inheritance

```python
class Shape:

    def __init__(self, color, is_polygon, description):
        self.color = color
        self.is_polygon = is_polygon
        self.description = description

    def display_data(self):
        print(f"\n=== {self.description.capitalize()} ===")
        print("Color:", self.color)
        print("Is the shape a polygon?", "Yes" if self.is_polygon else "No")
```

```python
class Triangle(Shape):

    def __init__(self, color, vertices, base, height):
        Shape.__init__(self, color, True, "Triangle")
        self.vertices = vertices
        self.base = base
        self.height = height
```

```python
class Circle(Shape):

    def __init__(self, color, radius):
        Shape.__init__(self, color, False, "Circle")
        self.radius = radius
```

Inheritance

```python
class Shape:

    def __init__(self, color, is_polygon, description):
        self.color = color
        self.is_polygon = is_polygon
        self.description = description

    def display_data(self):
        print(f"\n=== {self.description.capitalize()} ===")
        print("Color:", self.color)
        print("Is the shape a polygon?", "Yes" if self.is_polygon else "No")
```

```python
class Triangle(Shape):

    def __init__(self, color, vertices, base, height):
        Shape.__init__(self, color, True, "Triangle")
        self.vertices = vertices
        self.base = base
        self.height = height

class Circle(Shape):

    def __init__(self, color, radius):
        Shape.__init__(self, color, False, "Circle")
        self.radius = radius
```

Inheritance

```python
class Shape:

    def __init__(self, color, is_polygon, description):
        self.color = color
        self.is_polygon = is_polygon
        self.description = description

    def display_data(self):
        print(f"\n=== {self.description.capitalize()} ===")
        print("Color:", self.color)
        print("Is the shape a polygon?", "Yes" if self.is_polygon else "No")
```

```python
class Triangle(Shape):

    def __init__(self, color, vertices, base, height):
        Shape.__init__(self, color, True, "Triangle")
        self.vertices = vertices
        self.base = base
        self.height = height

class Circle(Shape):

    def __init__(self, color, radius):
        Shape.__init__(self, color, False, "Circle")
        self.radius = radius
```

Inheritance

```
>>> triangle.find_area()
Traceback (most recent call last):
  File "<pyshell#1>", line 1, in <module>
    triangle.find_area()
AttributeError: 'Triangle' object has no attribute 'find_area'
>>> circle.find_area()
Traceback (most recent call last):
  File "<pyshell#2>", line 1, in <module>
    circle.find_area()
AttributeError: 'Circle' object has no attribute 'find_area'
```

Inheritance

```
>>> triangle.find_area()
Traceback (most recent call last):
  File "<pyshell#1>", line 1, in <module>
    triangle.find_area()
AttributeError: 'Triangle' object has no attribute 'find_area'
>>> circle.find_area()
Traceback (most recent call last):
  File "<pyshell#2>", line 1, in <module>
    circle.find_area()
AttributeError: 'Circle' object has no attribute 'find_area'
```

Inheritance

The subclass inherits the methods from the superclass

```
class Superclass:
    # Body

class Subclass(Superclass):
    # Body
```

Now… An Example