Polymorphism Using Methods

In this lesson, we will implement polymorphism using methods.

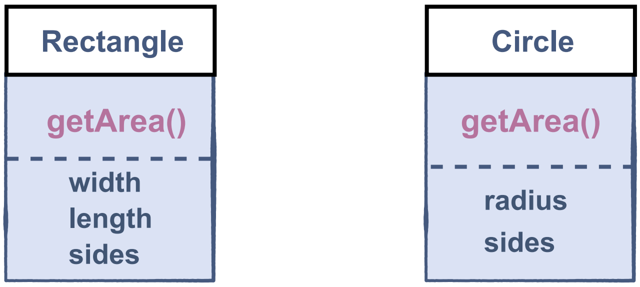
We'll cover the following

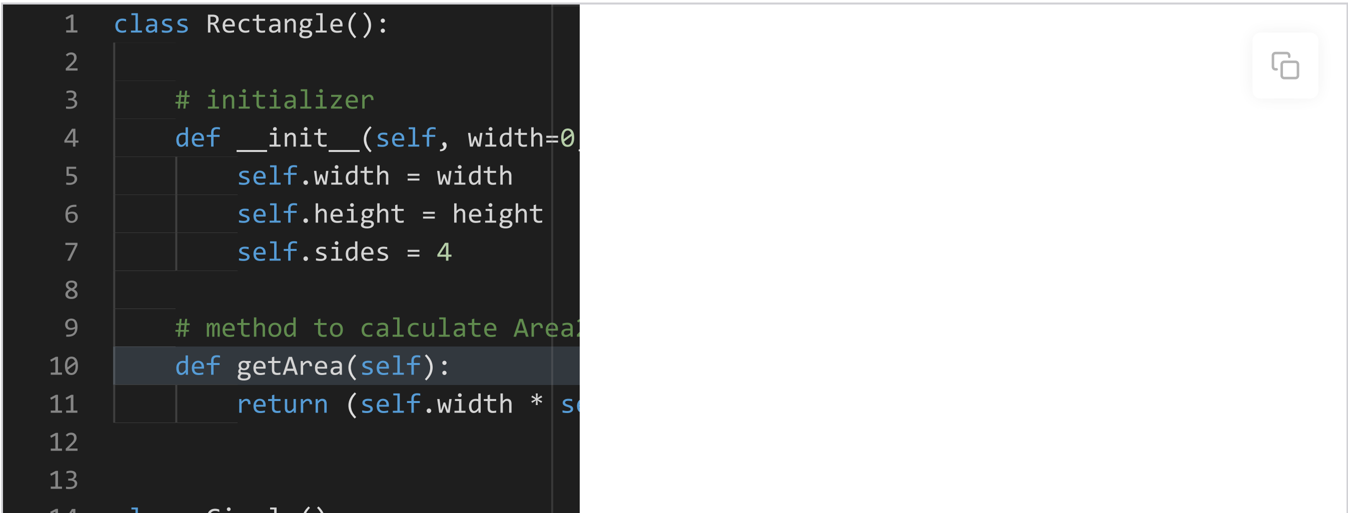
- Example
    - Explanation

We have learned how polymorphism is useful in making code manageable. In this lesson, we will learn how to implement polymorphism using methods. In the next lesson, we will implement it using inheritance.

Example

Here, we consider two shapes that are defined as classes: *Rectangle and Circle*. These classes contain the **getArea()** method which calculates the area for the respective shape depending on the values of their properties.



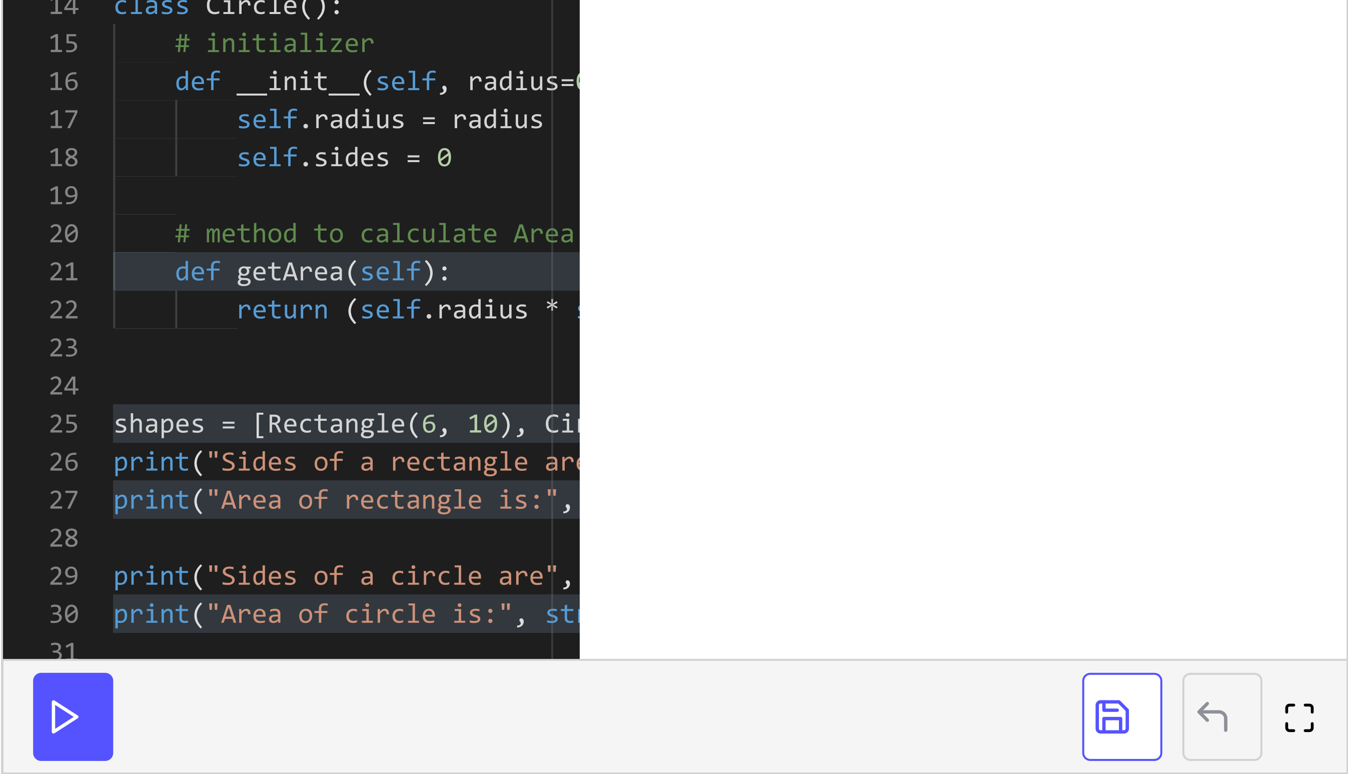


```
class Rectangle():

    # initializer
    def __init__(self, width=0
        self.width = width
        self.height = height
        self.sides = 4

    # method to calculate Area
    def getArea(self):
        return (self.width * s


```



```
14  class Circle():
15      # initializer
16      def __init__(self, radius=
17          self.radius = radius
18          self.sides = 0
19
20      # method to calculate Area
21      def getArea(self):
22          return (self.radius *
23
24
25  shapes = [Rectangle(6, 10), Ci
26  print("Sides of a rectangle are
27  print("Area of rectangle is:",
28
29  print("Sides of a circle are",
30  print("Area of circle is:", st
31
```

Explanation

- In the main function, at **line 25**, we have declared a list that has *two* objects in it.
- The *first* object is a Rectangle with `width` 6 and `height` 10, and the *second* object is a `Circle` of `radius` 7.
- Both the classes have the method `getArea()`, on **lines 10** and **21**, but the execution of this method is different for each class and this is how we have achieved polymorphism.
- Method calls on **lines 27** and **30** look identical, but different methods are called. Thus, we have achieved polymorphism.

This was one way of achieving polymorphism. In the next lesson, we will implement polymorphism using a more efficient and commonly used approach: **polymorphism using inheritance**.