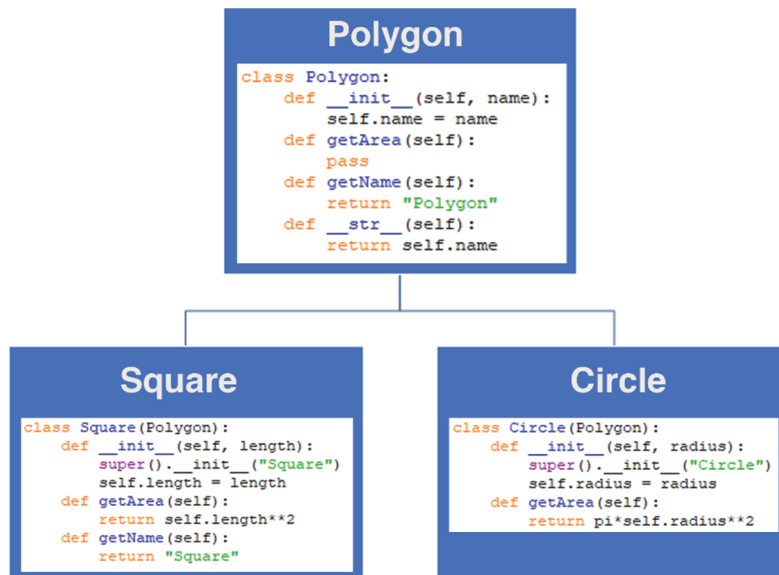


Polymorphism

The child classes in Python inherits methods and attributes from the parent class. We can redefine certain methods and attributes specifically for the child class using a feature called **Method Overriding**.



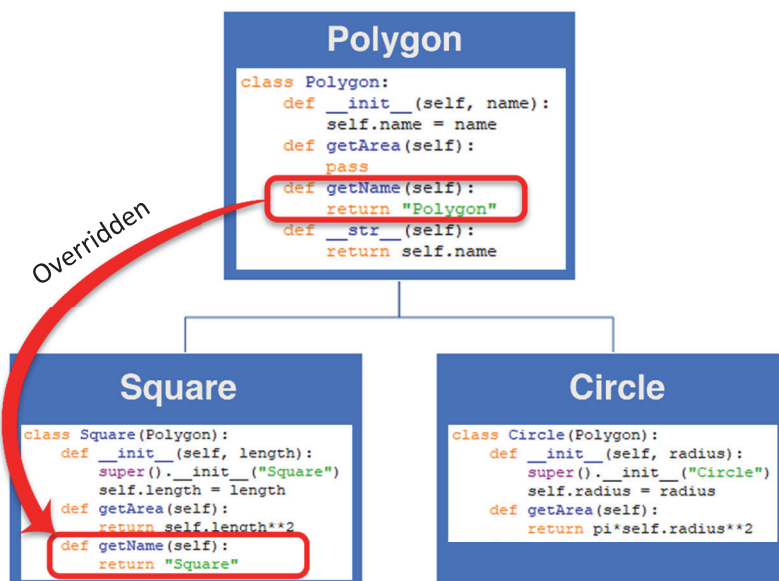
Polymorphism allows us to access these overridden methods and attributes that have the same name as defined in the parent class. For example, if we create an object called `squareObj` from the `Square` class:

```
squareObj = Square(7)
```

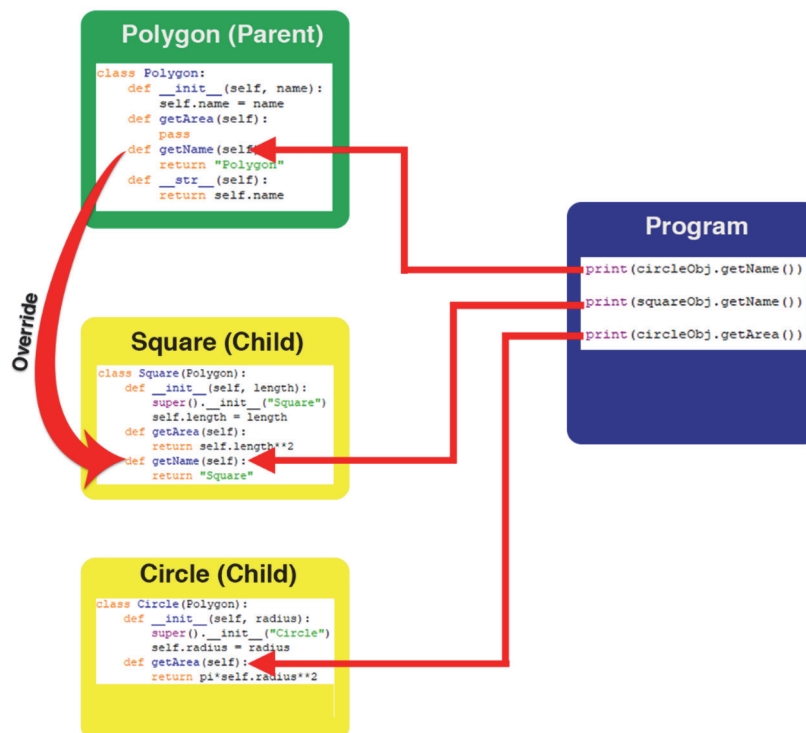
Then call the `getName()` method.

```
print(squareObj.getName())
```

The interpreter automatically recognizes that the `getName()` method for `squareObj` is overridden.



So the `getName()` method defined in the child class (Square) is used. The `getName()` method defined in the Square class is overriding the `getName()` method defined in the parent (Polygon). Similarly with `getArea()`.

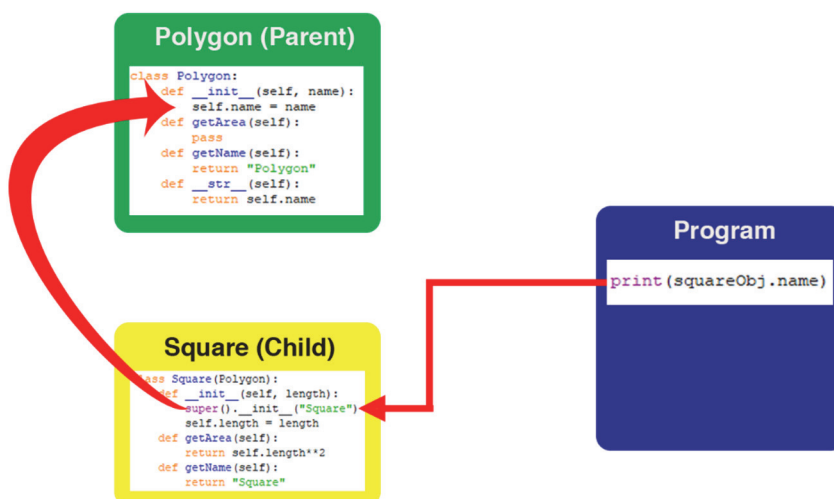


You'll also notice that we have called the `getName()` method from the circle class. In this case, there is no method defined in the circle class, so the interpreter will call the `getName()` method from the parent.

If we reference the name attribute from squareObj

```
print(squareObj.name)
```

You'll notice that name is defined in the parent not in the child class(Square). To access this we need the `super()` function in the child class to allow access.



Have a look at polyclass2.py