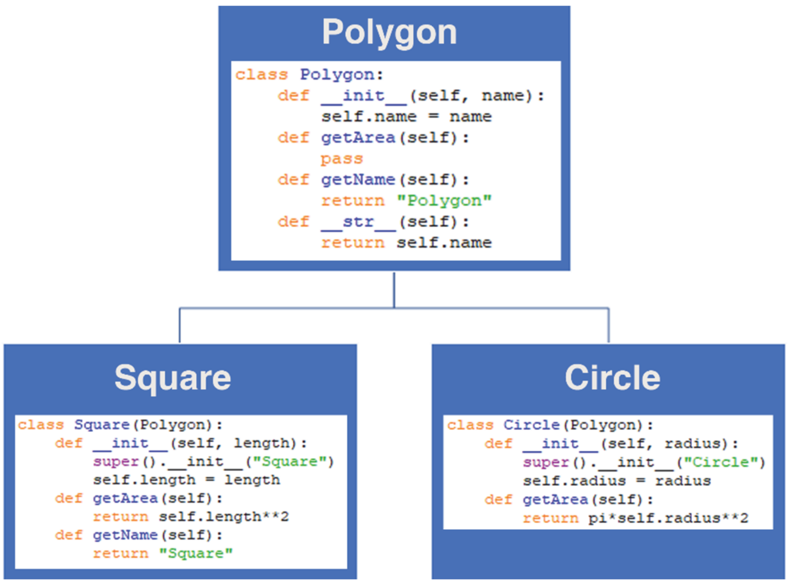
Polymorphism

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**.



```
class Polygon:
    def __init__(self, name):
        self.name = name
    def getArea(self):
        pass
    def getName(self):
        return "Polygon"
    def __str__(self):
        return self.name
```

```
class Square(Polygon):
    def __init__(self, length):
        super().__init__("Square")
        self.length = length
    def getArea(self):
        return self.length**2
    def getName(self):
        return "Square"
```

```
class Circle(Polygon):
    def __init__(self, radius):
        super().__init__("Circle")
        self.radius = radius
    def getArea(self):
        return pi*self.radius**2
```

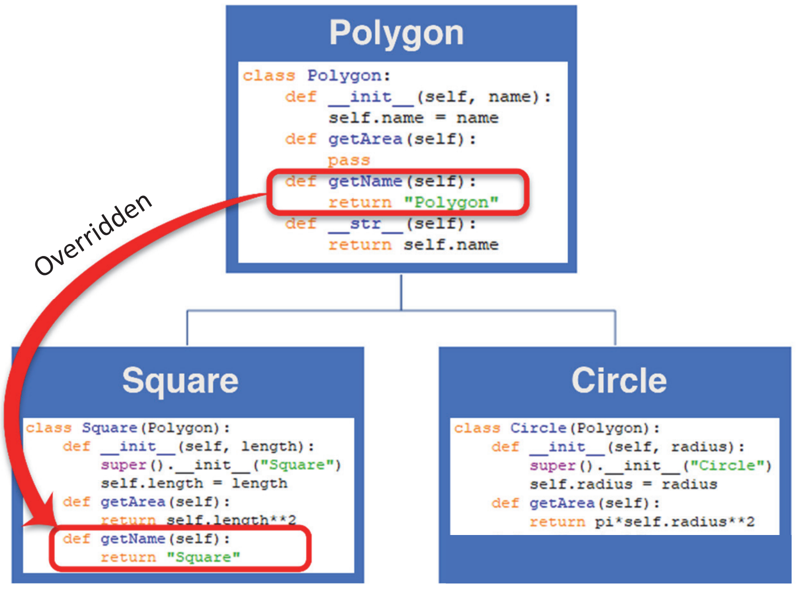
Polymorphism allows us to access these overridden methods and attributes that have the same name as defined in the parent class. For example, if we create an object called squareObj from the Square class:

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

Then call the `getName()` method.

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

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

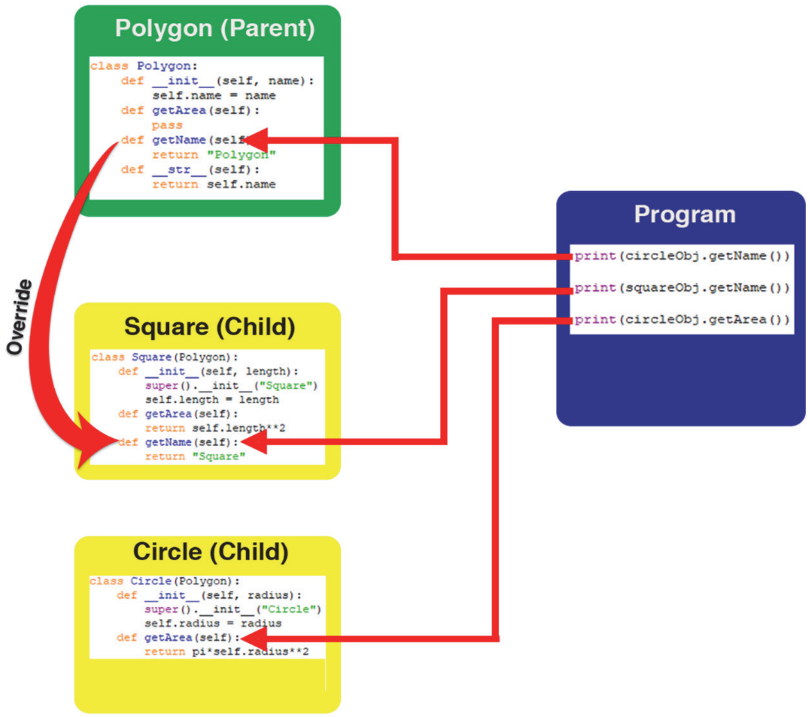


```
class Polygon:
    def __init__(self, name):
        self.name = name
    def getArea(self):
        pass
    def getName(self):
        return "Polygon"
    def __str__(self):
        return self.name
```

```
class Square(Polygon):
    def __init__(self, length):
        super().__init__("Square")
        self.length = length
    def getArea(self):
        return self.length**2
    def getName(self):
        return "Square"
```

```
class Circle(Polygon):
    def __init__(self, radius):
        super().__init__("Circle")
        self.radius = radius
    def getArea(self):
        return pi*self.radius**2
```

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()`.



```
class Polygon:
    def __init__(self, name):
        self.name = name
    def getArea(self):
        pass
    def getName(self):
        return "Polygon"
    def __str__(self):
        return self.name
```

```
class Square(Polygon):
    def __init__(self, length):
        super().__init__("Square")
        self.length = length
    def getArea(self):
        return self.length**2
    def getName(self):
        return "Square"
```

```
class Circle(Polygon):
    def __init__(self, radius):
        super().__init__("Circle")
        self.radius = radius
    def getArea(self):
        return pi*self.radius**2
```

```
print(circleObj.getName())
print(squareObj.getName())
print(circleObj.getArea())
```

You'll also notice that we have called the `getName()` method from the circle class object circleObj.

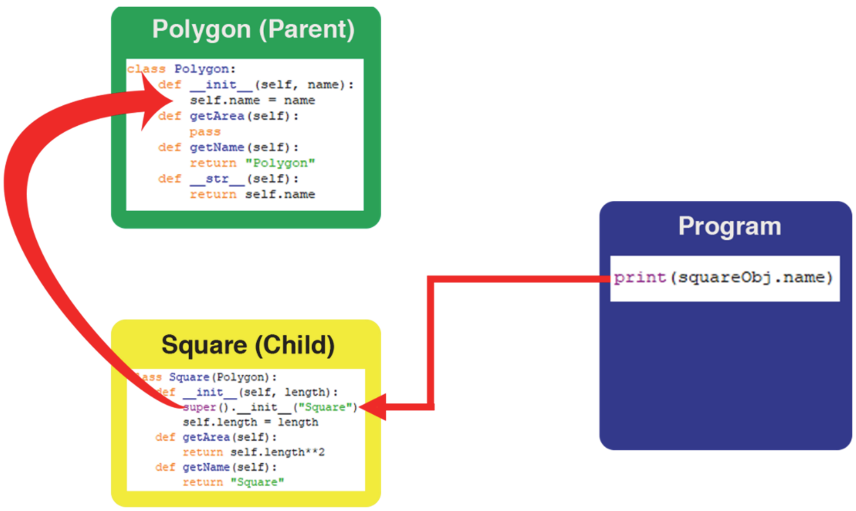
```
print (circleObj.getName()
```

In this case, there is no `getName()` 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 class (Polygon) not in the child class (Square). To access this attribute from the child, we need the `super()` function in the child class to call the `__init__()` constructor method defined in the parent class (Polygon).

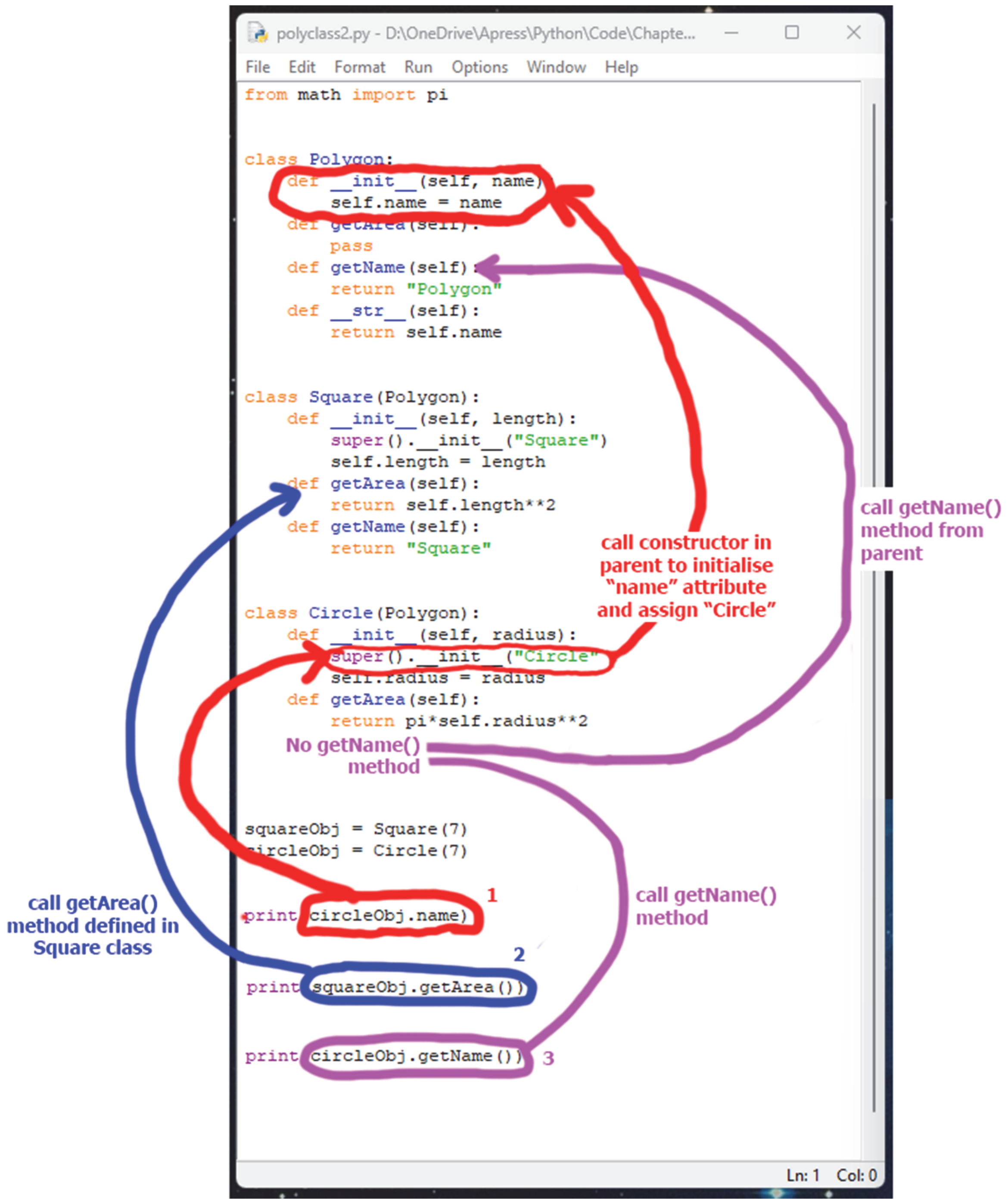


```
class Polygon:
    def __init__(self, name):
        self.name = name
    def getArea(self):
        pass
    def getName(self):
        return "Polygon"
    def __str__(self):
        return self.name
```

```
class Square(Polygon):
    def __init__(self, length):
        super().__init__("Square")
        self.length = length
    def getArea(self):
        return self.length**2
    def getName(self):
        return "Square"
```

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

Have a look at polyclass2.py


polyclass2.py - D:\OneDrive\Apress\Python\Code\Chapte...
File Edit Format Run Options Window Help
call constructor in parent to initialise "name" attribute and assign "Circle"
call getName() method from parent
No getName() method
call getName() method
call getArea() method defined in Square class
1
2
3
Ln: 1 Col: 0

```python
from math import pi


class Polygon:
    def __init__(self, name):
        self.name = name
    def getArea(self):
        pass
    def getName(self):
        return "Polygon"
    def __str__(self):
        return self.name


class Square(Polygon):
    def __init__(self, length):
        super().__init__("Square")
        self.length = length
    def getArea(self):
        return self.length**2
    def getName(self):
        return "Square"


class Circle(Polygon):
    def __init__(self, radius):
        super().__init__("Circle")
        self.radius = radius
    def getArea(self):
        return pi*self.radius**2


squareObj = Square(7)
circleObj = Circle(7)


print(circleObj.name)

print(squareObj.getArea())

print(circleObj.getName())
```