Inheritance Examples

In [11]:
```
class Vehicle:
    #Parent Class
    def __init__(self, price):
        self.price = price

    def display_price(self):
        return f'Price = ${self.price}'

    def __str__(self):
        return self.display_price()

class NamedVehicle(Vehicle):
    #"Child/Derived class"
    def __init__(self, price, name):
        # Notice you must pass self if not using super()
        Vehicle.__init__(self, price)
        self.name = name

    # New method
    def display_name(self):
        return f'Vehicle = {self.name}'

    # Overridden method
    def __str__(self):
        # Reuse of __str__ of parent class
        return self.display_name() + ', ' + super().__str__()


obj = NamedVehicle(1200, 'BMW')
print(obj.display_name())
print(obj.display_price())
print(obj)
```

```
Vehicle = BMW
Price = $1200
Vehicle = BMW, Price = $1200
```

In [12]:
```
class Quadrilateral:
    def __init__(self, a, b, c, d):
        self.side1 = a
        self.side2 = b
        self.side3 = c
        self.side4 = d

    def perimeter(self):
        return self.side1 + self.side2 + self.side3 + self.side4

class Rectangle(Quadrilateral):
    def __init__(self, a,b):
        super().__init__(a, b, a, b)

    def area(self):
        return self.side1 * self.side2

class Square(Rectangle):
    def __init__(self, a):
        super().__init__(a, a)

    def area(self):
        return pow(self.side1, 2)
```

In [13]:
```
quad = Quadrilateral(5,10,2,3)
print(quad.perimeter())
```

```
20
```

In [14]:
```
try:
    print(quad.area())
except Exception as e:
    print(e)
```

```
'Quadrilateral' object has no attribute 'area'
```

In [15]:
```
quad = Quadrilateral(5,10,5,10)
try:
    print(quad.area())
except Exception as e:
    print(e)
```

```
'Quadrilateral' object has no attribute 'area'
```

In [16]:
```
rect = Rectangle(5,10)
print(rect.perimeter())
print(rect.area())
```

```
30
50
```

In [17]:
```
sq = Square(5)
print(sq.perimeter())
print(sq.area())
```

```
20
25
```

In [18]:
```
# Now let's modify the square
sq.side1 = 2
sq.side2 = 2
sq.side3 = 2
sq.side3 = 2 # Oops...bug!!!!
print(sq.perimeter()) #This will still proceed and not notice the bug
print(sq.area())
```

```
11
4
```

In [19]:
```
# Safer rewrite of Square
class Square(Rectangle):
    def __init__(self, a):
        super().__init__(a, a)

    def area(self):
        #assert will do nothing if the statement is true, and will throw
an AssertionError if it is false
        assert self.side1 == self.side2, 'side1 != side2'
        assert self.side1 == self.side3, 'side1 != side3'
        assert self.side1 == self.side4, 'side1 != side4'
        return pow(self.side1, 2)

    def set_side(self, a):
        self.side1 = a
        self.side2 = a
        self.side3 = a
        self.side4 = a
```

In [20]:
```
sq = Square(5)
print(sq.perimeter())
print(sq.area())

# Now let's modify the square
sq.side1 = 2
sq.side2 = 2
sq.side3 = 2
sq.side3 = 2 # Oops...bug!!!!
try:
    print(sq.perimeter())
    print(sq.area())
except Exception as e:
    print('ERROR:', e)
```

```
20
25
11
ERROR: side1 != side4
```

In [ ]:
```

```