GKTCS INNOVATIONS

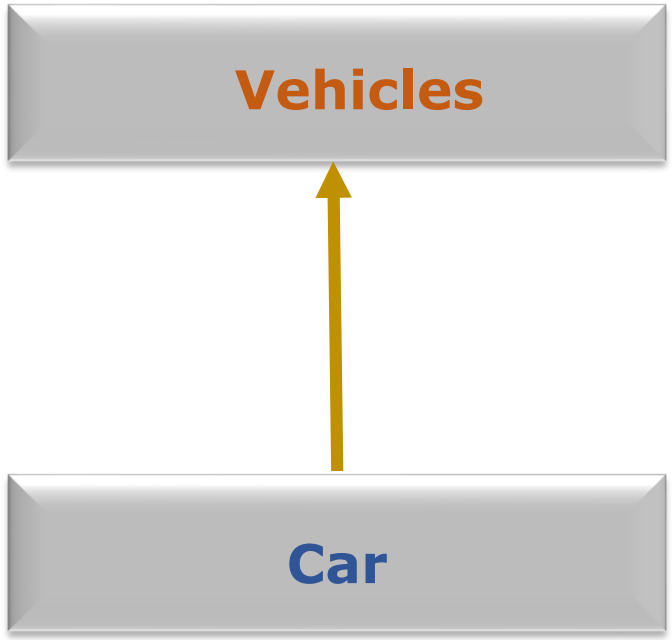
Inheritance
Polymorphism
Overloading
Overriding

What is Inheritance?

Inheritance is a powerful feature in object oriented programming.

It is the capability of one class to derive or inherit the properties from some another class.

Every Car is a vehicles. To show this relationship, we take an example.



In this representation, we use an arrow towards the base class as a UML (Unified Modeling Language) convention.

GKTCS INNOVATIONS

Vehicles can be called any of the following:

- ❑ Super Class
- ❑ Parent Class
- ❑ Base Class

And car is:

- ❑ Sub Class
- ❑ Child Class
- ❑ Derived Class

Inheritance Syntax

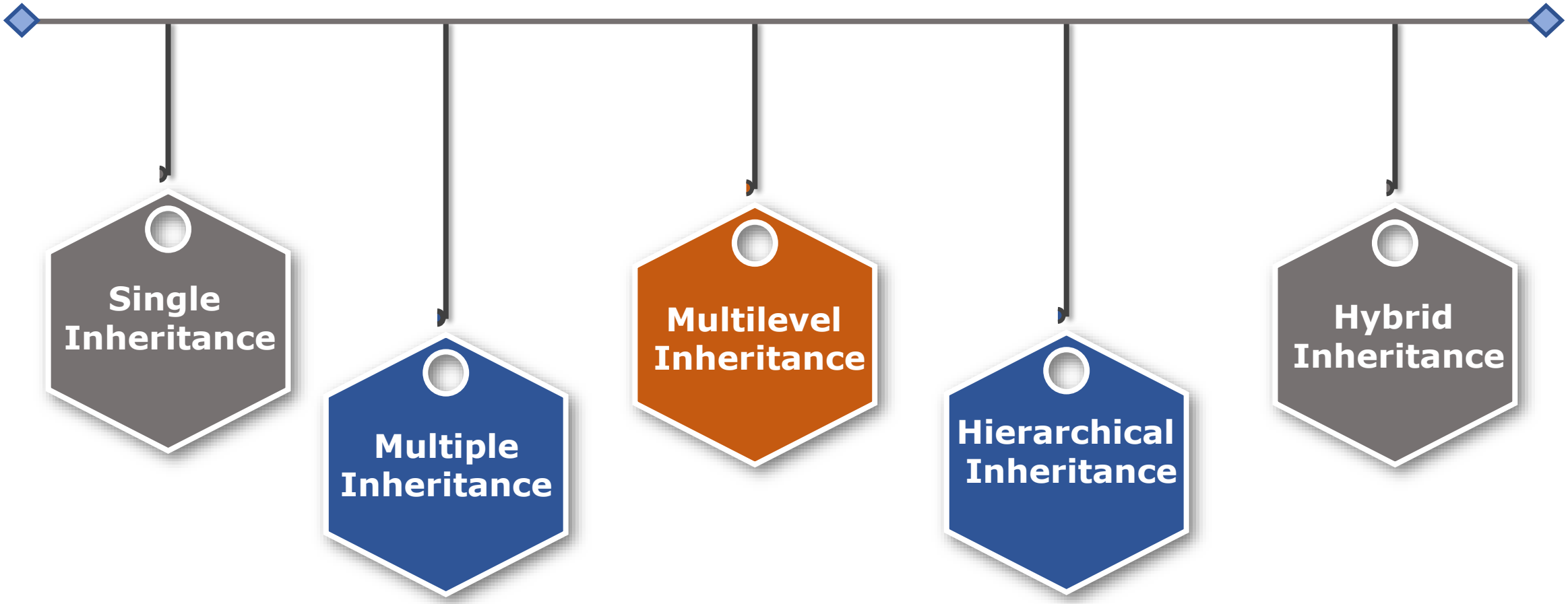
GKTCS INNOVATIONS

```
>>> class Person:
        pass
>>> class Car(Vehicles):
        pass
>>> issubclass(Car,Vehicles)
```

Here, class Car inherits from class Vehicles. We use the function issubclass() to confirm that car is a subclass of person.

Types of Inheritance

GKTCS INNOVATIONS



Single Inheritance

GKTCS INNOVATIONS

Child class inherits from only one parent class

Example:

```
class Animal:
    def speak(self):
        print("Animal Speaking")

#child class Lion inherits the base class Animal
class Lion(Animal):
    def roar(self):
        print("Lion roaring")

d = Lion()
d.roar()
d.speak()
```

Output:

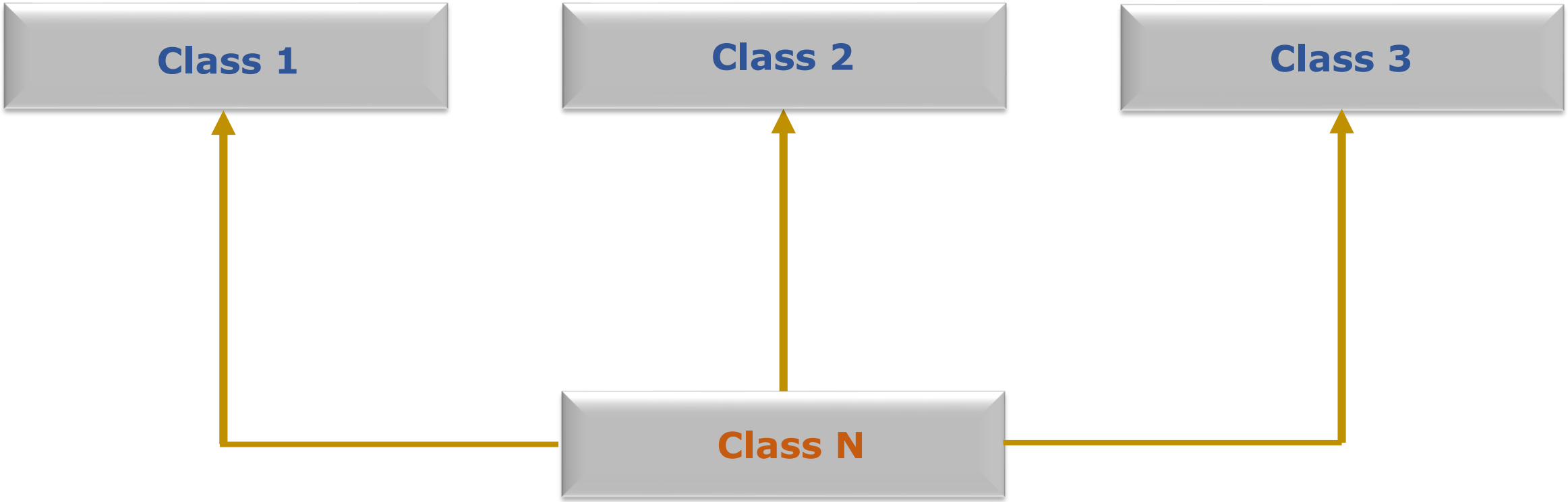
GKTCS INNOVATIONS

Lion roaring
Animal Speaking

Multiple inheritance

GKTCS INNOVATIONS

Inherit multiple base classes in the child class



Example:

GKTCS INNOVATIONS

```
class Calculation1:
    def Addition(self,x,y):
        return x+y;

class Calculation2:
    def Multiplication(self,x,y):
        return a*b;

class Derived(Calculation1,Calculation2):
    def Division(self,a,b):
        return a/b;

d = Derived()
print(d.Addition(2,4))
print(d.Multiplication(2,4))
print(d.Division(2,4))
```

Output:

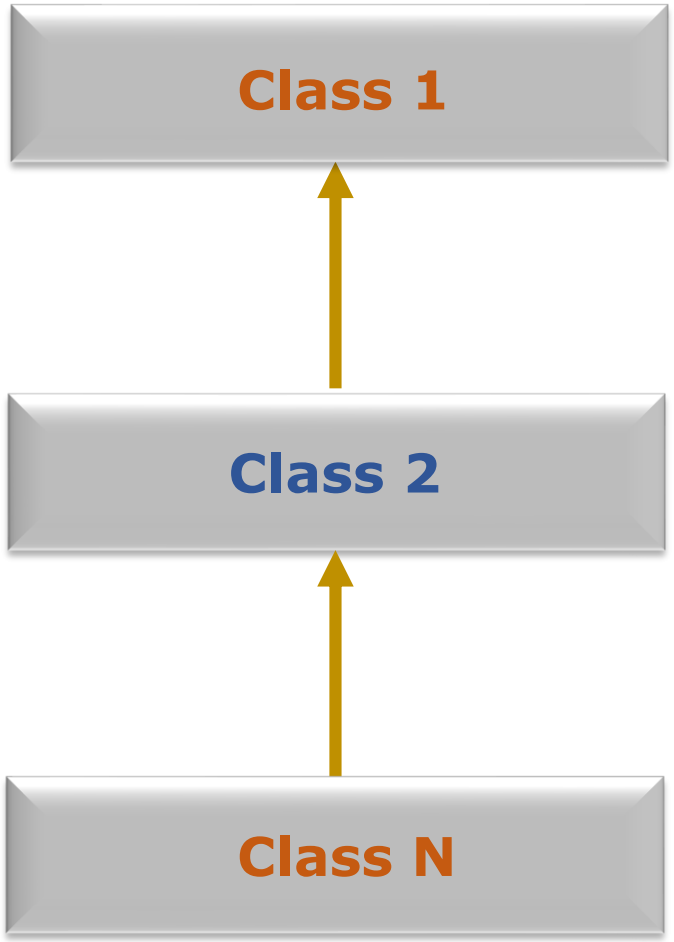
GKTCS INNOVATIONS

```
6
8
0.5
```

Multi-Level inheritance

GKTCS INNOVATIONS

In Multi-level inheritance derived class inherits from another derived class. There is no limit for level.



Example:

GKTCS INNOVATIONS

```
class Animal:
    def speak(self):
        print("Animal Speaking")

#The child class Lion inherits the base class Animal
class Lion(Animal):
    def roar(self):
        print("Lion roaring")

#The child class BabyLion inherits another child class Lion
class BabyLion(Lion):
    def eat(self):
        print("Eating meat...")

d = BabyLion()
d.roar()
d.speak()
d.eat()
```

Output:

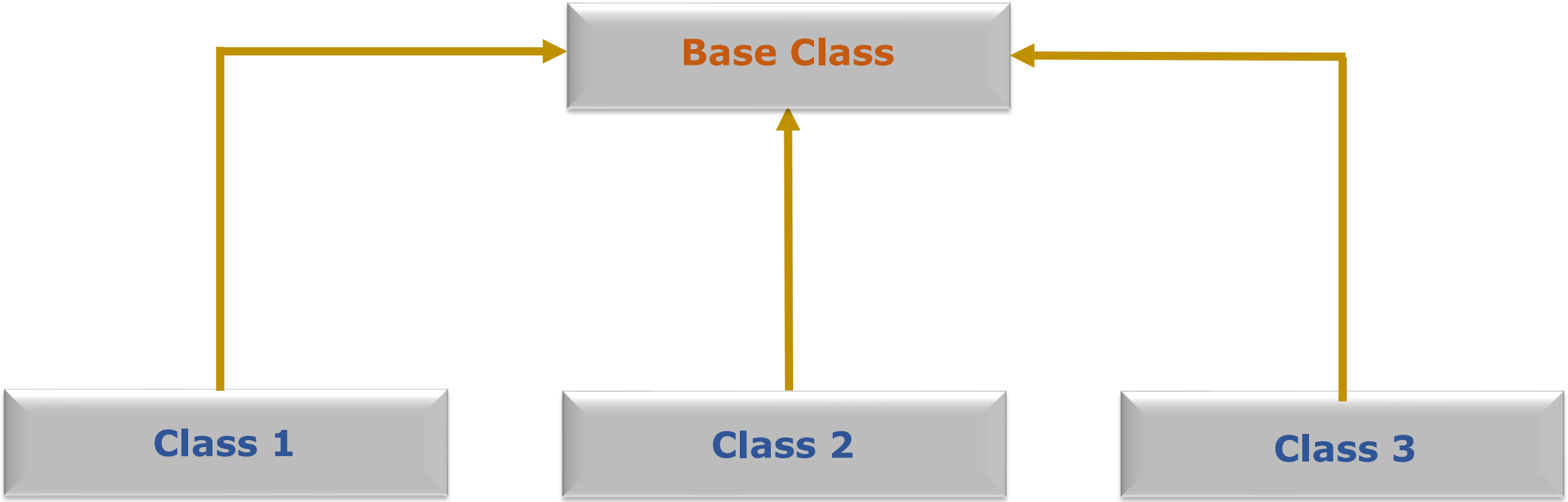
GKTCS INNOVATIONS

Lion roaring
Animal Speaking
Eating meat...

Hierarchical inheritance

GKTCS INNOVATIONS

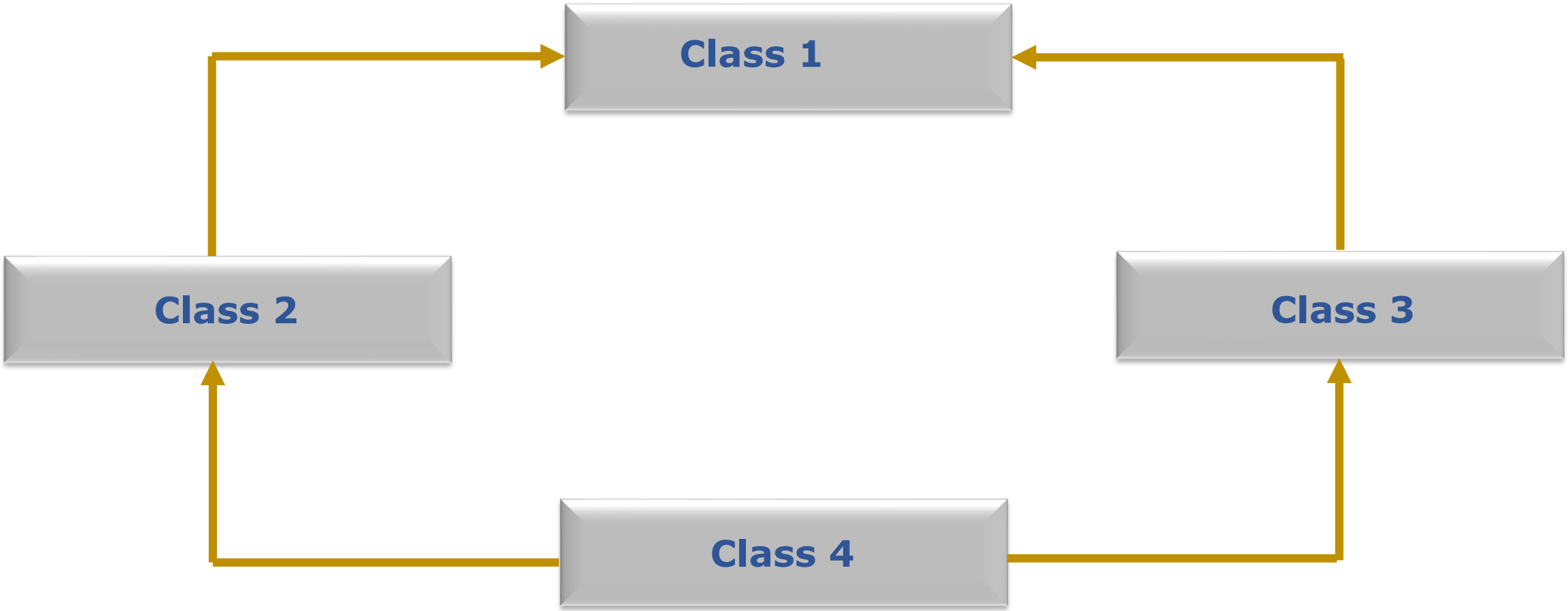
In hierarchical inheritance more than one derived classes are created from a single base class.



Hybrid inheritance

GKTCS INNOVATIONS

Hybrid inheritance is a combination of multiple inheritance and multilevel inheritance.



Polymorphism

GKTCS INNOVATIONS

- Polymorphism means many forms or multiple form. In programming polymorphism means the same name of function (but different parameters) that is used for different types.
- Polymorphism simply means that we can call the same method name with different parameters, and depending on the parameters, it will do different things.

Example:

GKTCS INNOVATIONS

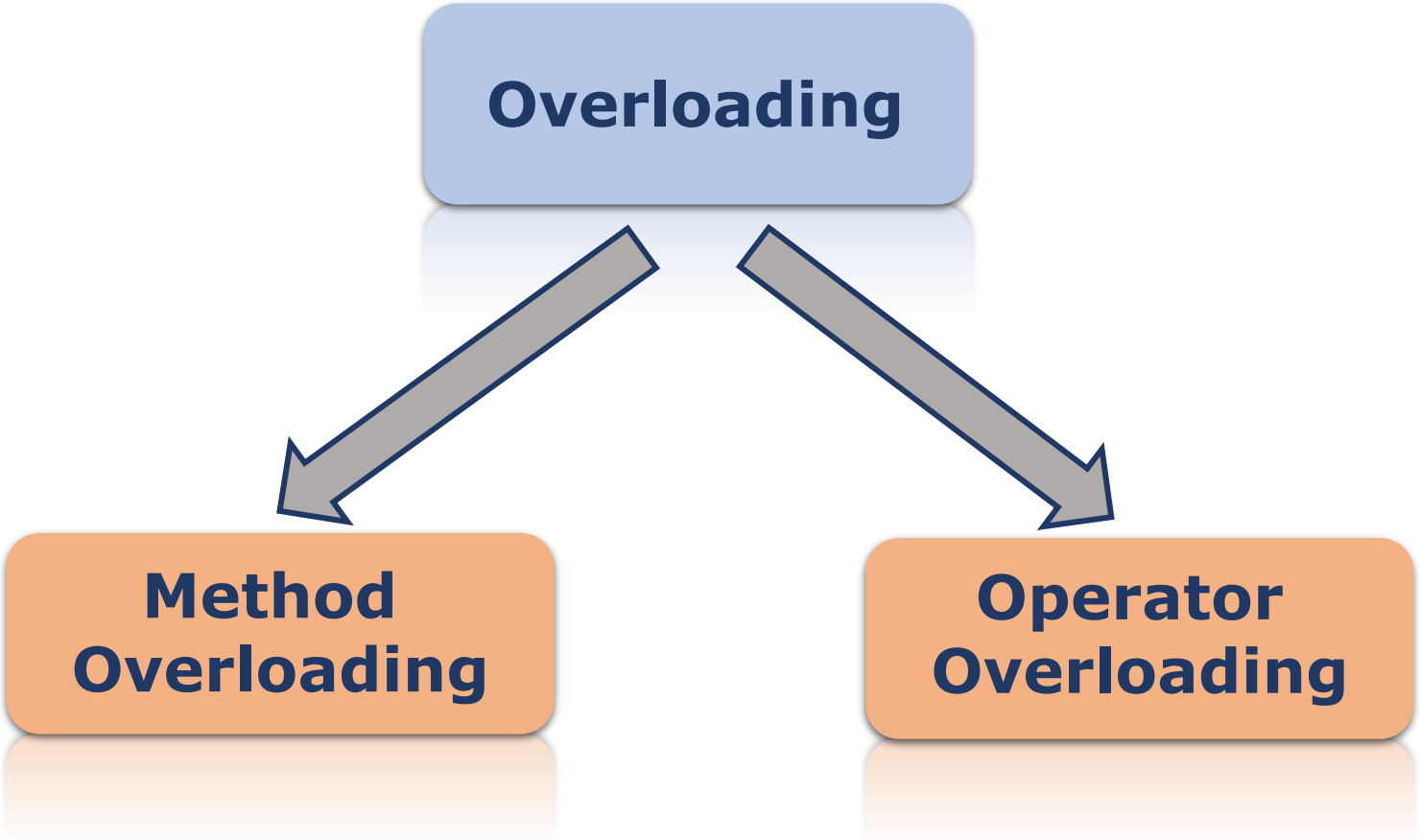
```
len("GKTCS")                    # returns 5 as result
len([5,2,8,4,45,75,3,92,33])    # returns 9 as result
```

In this case the function len() taking string as input in the first case and is taking list as input in the second case.

Overloading

GKTCS INNOVATIONS

Overloading is the ability of a function or operator to behave differently dependin g on the parameters passed on to the function or the operands on which the oper ator operates.



Method OR Function Overloading

GKTCS INNOVATIONS

Method overloading or function overloading is a type of polymorphism in which we can define a number of methods with the same name but with a different number of parameters as well as parameters can be of different types.

Example:

GKTCS INNOVATIONS

```
# Takes two argument and print their Addition
def addition(a, b):
    x = a + b
    print(x)

# Takes three argument and print their Addition
def addition(a, b, c):
    x = a + b + c
    print(x)

# below line shows an error
#addition(7, 2)

# This line will call the second product method
addition(2, 5, 1)
```

Output:

GKTCS INNOVATIONS

```
08
```

- In the above code we have defined two addition method, but we can only use the second addition method, as python does not supports method overloading.

- We may define many method of same name and different argument but we can only use the latest defined method. Calling the other method will produce an error. Like here calling **addition(7,2)** will produce an error as the latest defined addition method takes three arguments.

Operator Overloading

GKTCS INNOVATIONS

We can use '+' operator for adding numbers and at the same time to concatenate strings. It is possible because '+' operator is overloaded by both int class and str class.

Example:

GKTCS INNOVATIONS

```
# Addition of two numbers
print(3 + 2)

# Concatenate two strings
print("GKTCS" + "Innovations")

# Product of two numbers
print(3 * 2)

# Repeat the String
print("GKTCS"*3)
```

Output:

GKTCS INNOVATIONS

```
5
GKTCS Innovations
6
GKTCSGKTCSGKTCS
```

Overriding

GKTCS INNOVATIONS

Override means having two methods with the same name but doing different tasks. It means that one of the methods overrides the other.

The concept of Method overriding allows us to change or override the Parent Class function in the Child Class.

GKTCS INNOVATIONS

In Python, to override a method, you have to meet certain conditions

- You can’t override a method within the same class. It means you have to do it in the child class using the Inheritance concept.

- To override the Parent Class method, you have to create a method in the Child class with the same name and the same number of parameters.

Example:

GKTCS INNOVATIONS

```
# Python Method Overriding
class Employee:
    def message(self):
        print('This message is from Employee Class')

class Company(Employee):
    def message(self):
        print('This Company class is inherited from Employee’)

emp = Employee()
emp.message()
comp = Company()
comp.message()
```

Output:

GKTCS INNOVATIONS

```
'This message is from Employee Class'
'This Company class is inherited from Employee'
```