PYTHON-INHERITANCE

What is Inheritance

Inheritance is the ability to ‘inherit’ features or attributes from already written classes into newer classes we make. These features and attributes are defined data structures and the functions we can perform with them, a.k.a. Methods. It promotes code reusability, which is considered one of the best industrial coding practices as it makes the codebase modular.

In python inheritance, new class/es inherits from older class/es. The new class/es copies all the older class's functions and attributes without rewriting the syntax in the new class/es. These new classes are called derived classes, and old ones are called base classes.

Types of Inheritance in Python

1. Single Inheritance
2. Multiple Inheritance
3. Multi Level Inheritance
4. Hierarchical Inheritance in Python
5. Hybrid Inheritance in Python

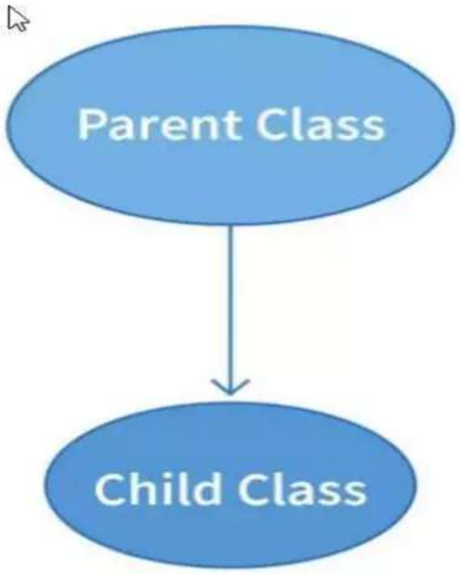
Parent1 -> Child1 : Single Inheritance

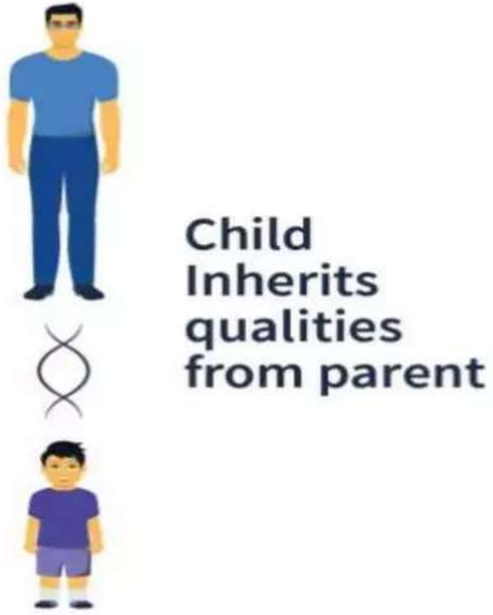
Parent1 -> Child1 -> Child2 : Multi – Level Inheritance

Parent1 -> Child2 <- Parent2 : Multiple Inheritance

Single Inheritance in python

Single Inheritance is the simplest form of inheritance where a single child class is derived from a single parent class. Due to its candid nature, it is also known as Simple Inheritance.



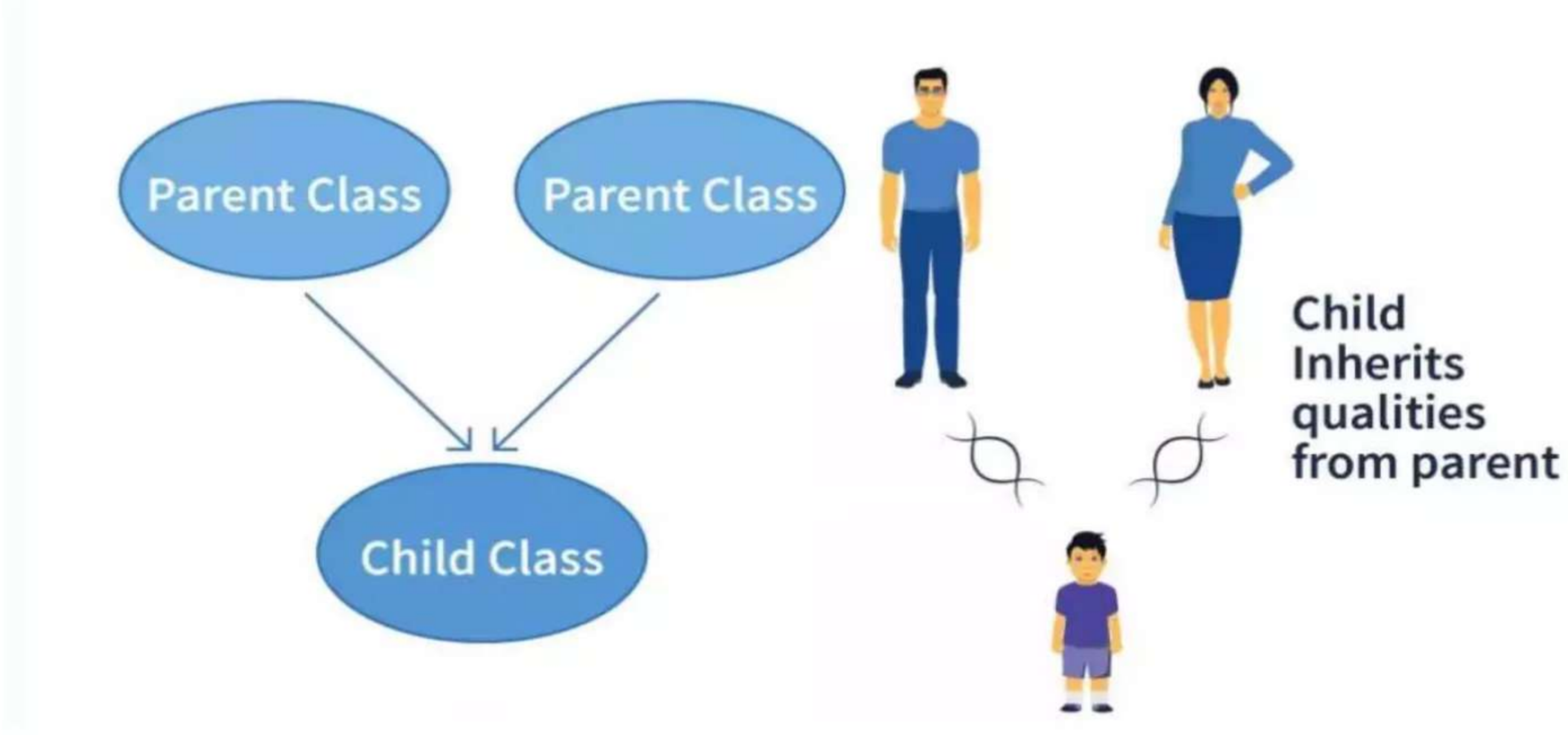


Single Inheritance Example

```
class parent:                          # parent class
    def func1(self):
        print("Hello Parent")

class child(parent):
  # child class
    def func2(self):                        # we include the parent class
        print("Hello Child")           # as an argument in the child
                                       # class

# Driver Code
test = child()                         # object created
test.func1()                           # parent method called via child object
test.func2()                           # child method called

Output :
 Hello Parent
 Hello Child
```

Multiple Inheritance in Python

In multiple inheritance, a single child class is inherited from two or more parent classes. It means the child class has access to all the parent classes' methods and attributes.

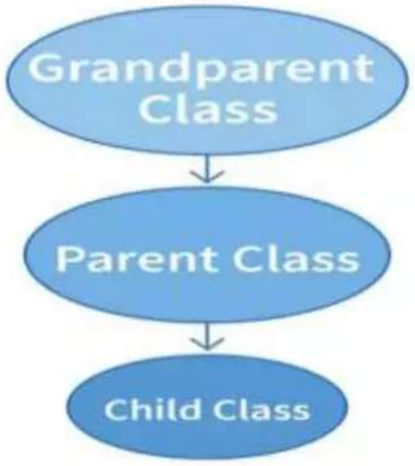


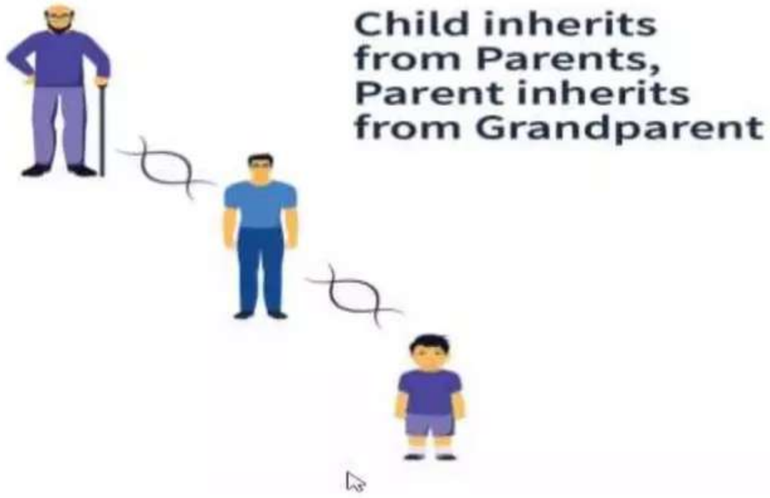
Multiple Inheritance Example

```
class parent1:                          # first parent class
    def func1(self):
        print("Hello Parent1")

class parent2:                          # second parent class
    def func2(self):
        print("Hello Parent2")

class parent3:                          # third parent class
    def func2(self):                        # the function name is same as parent2
        print("Hello Parent3")

class child(parent1, parent2, parent3):     # child class
    def func3(self):                        # we include the parent classes
        print("Hello Child")            # as an argument comma separated

# Driver Code
test = child()          # object created
test.func1()            # parent1 method called via child
test.func2()            # parent2 method called via child instead of parent3
test.func3()            # child method called

Output:
Hello Parent1
 Hello Parent2
 Hello Child
```

Multilevel Inheritance in Python

In multilevel inheritance, we go beyond just a parent-child relation. We introduce grandchildren, great-grandchildren, grandparents, etc. We have seen only two levels of inheritance with a superior parent class/es and a derived class/es, but here we can have multiple levels where the parent class/es itself is derived from another class/es.



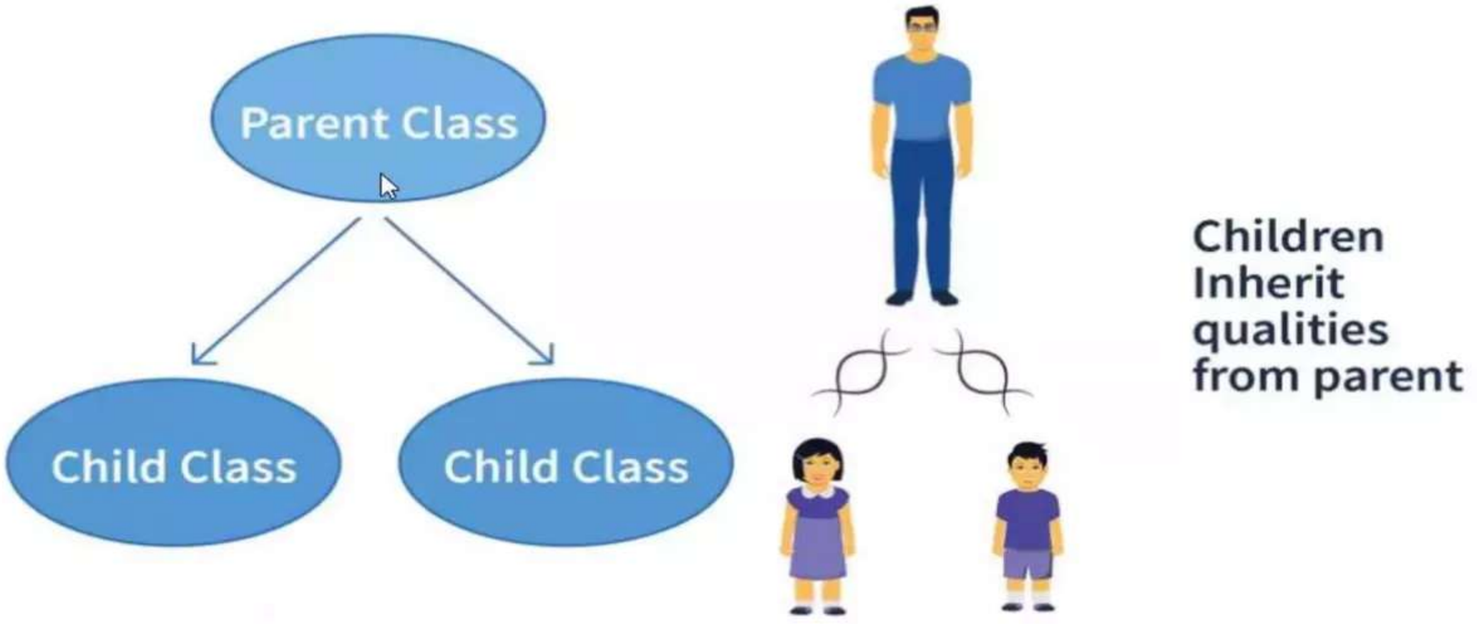


Example for MultiLevel Inheritance

```
class grandparent:                          # first level
    def func1(self):
        print("Hello Grandparent")

class parent(grandparent):                  # second level
    def func2(self):
        print("Hello Parent")

class child(parent):                        # third level
    def func3(self):
        print("Hello Child")


# Driver Code
test = child()                              # object created
test.func1()                                # 3rd level calls 1st level
test.func2()                                # 3rd level calls 2nd level
test.func3()                                # 3rd level calls 3rd level

output:
Hello Grandparent
Hello Parent
Hello Child
```

Hierarchical Inheritance

Hierarchical Inheritance is the right opposite of multiple inheritance. It means that, there are multiple derived child classes from a single parent class.

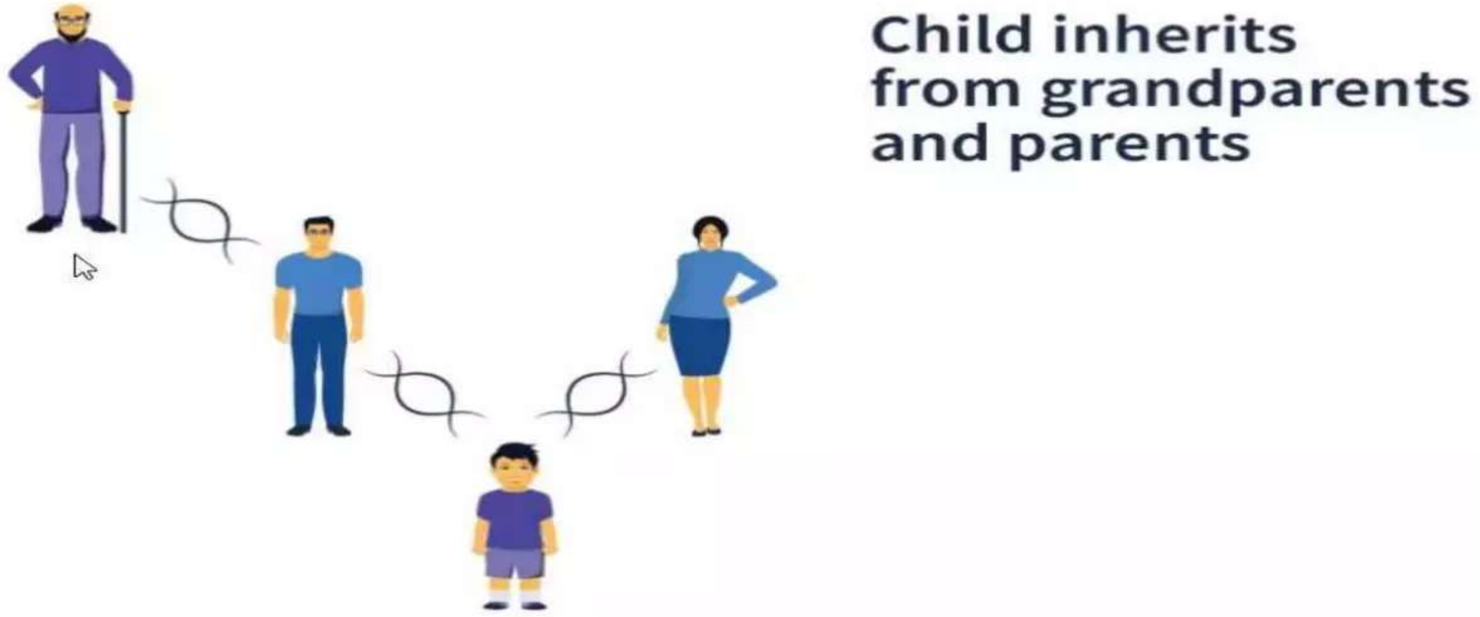


Example for Hierarchical Inheritance

```
class parent:                                   # parent class
    def func1(self):
        print("Hello Parent")

class child1(parent):                           # first child class
    def func2(self):
        print("Hello Child1")


class child2(parent):                           # second child class
    def func3(self):
        print("Hello Child2")


# Driver Code
test1 = child1()                                 # objects created
test2 = child2()

test1.func1()                                   # child1 calling parent method
test1.func2()                                   # child1 calling its own method

test2.func1()                                   # child2 calling parent method
test2.func3()                                   # child2 calling its own method

Output:
Hello Parent
Hello Child1
Hello Parent
Hello Child2
```

Hybrid Inheritance in Python

Hybrid Inheritance is the mixture of two or more different types of inheritance. Here we can have many relationships between parent and child classes with multiple levels.



Example for Hybrid Inheritance

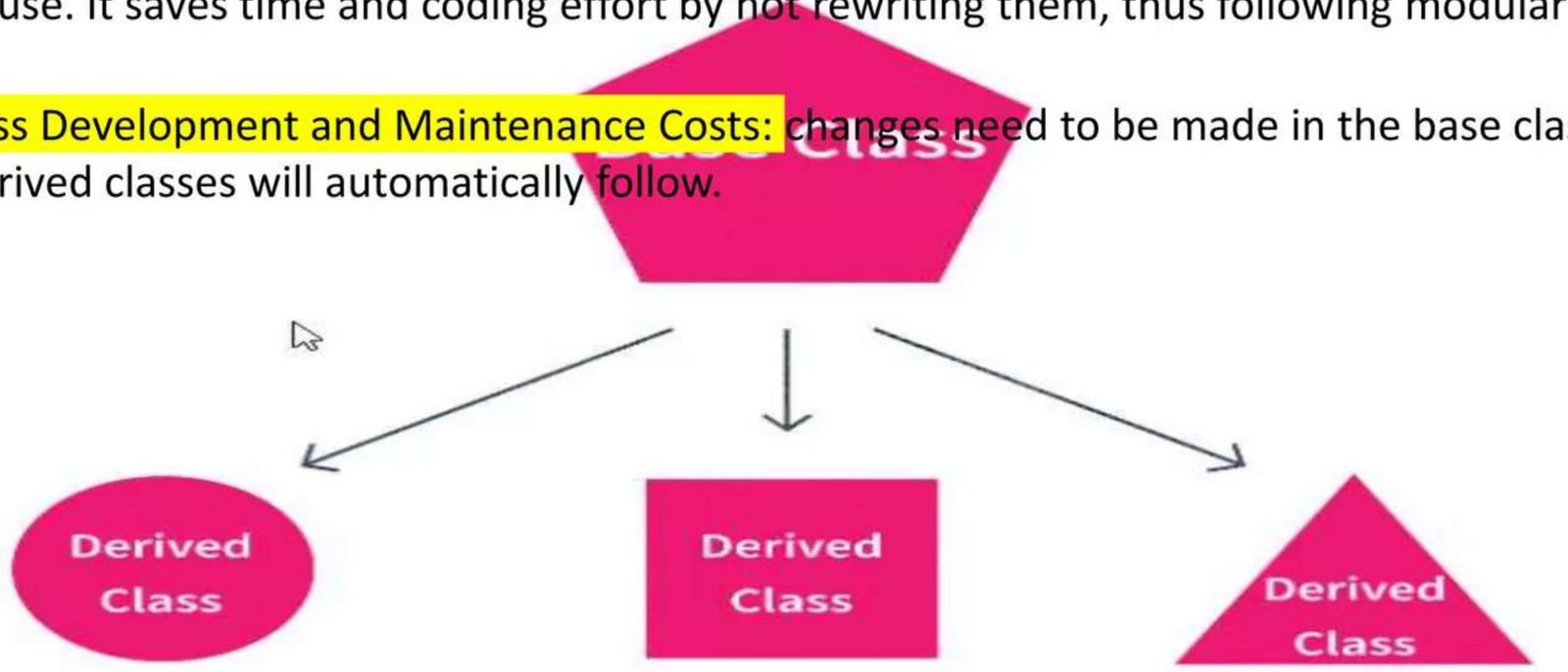
```
class parent1:                                  # first parent class
    def func1(self):
        print("Hello Parent")

class parent2:                                  # second parent class
    def func2(self):
        print("Hello Parent")

class child1(parent1):                          # first child class
    def func3(self):
        print("Hello Child1")

class child2(child1, parent2):                  # second child class
    def func4(self):
        print("Hello Child2")

# Driver Code
test1 = child1()                                # object created
test2 = child2()

test1.func1()                                   # child1 calling parent1 method
test1.func3()                                   # child1 calling its own method

test2.func1()                                   # child2 calling parent1 method
test2.func2()                                   # child2 calling parent2 method
test2.func3()                                   # child2 calling child1 method
test2.func4()                                   # child2 calling its own method

Output :
Hello Parent1
Hello Child1
Hello Parent1
Hello Parent2
Hello Child1
Hello Child2
```

Advantage of Inheritance in Python

Modular Codebase:
Increases modularity, i.e., breaking down codebase into modules, making it easier to understand. Here, each class we define becomes a separate module that can be inherited separately by one or many classes.

Code Reusability: the child class copies all the attributes and methods of the parent class into its class and use. It saves time and coding effort by not rewriting them, thus following modularity paradigms.

Less Development and Maintenance Costs: changes need to be made in the base class; all derived classes will automatically follow.



Disadvantage of Inheritance in Python

Decreases the Execution Speed: loading multiple classes because they are interdependent

Tightly Coupled Classes: this means that even though parent classes can be executed independently, child classes cannot be executed without defining their parent classes.