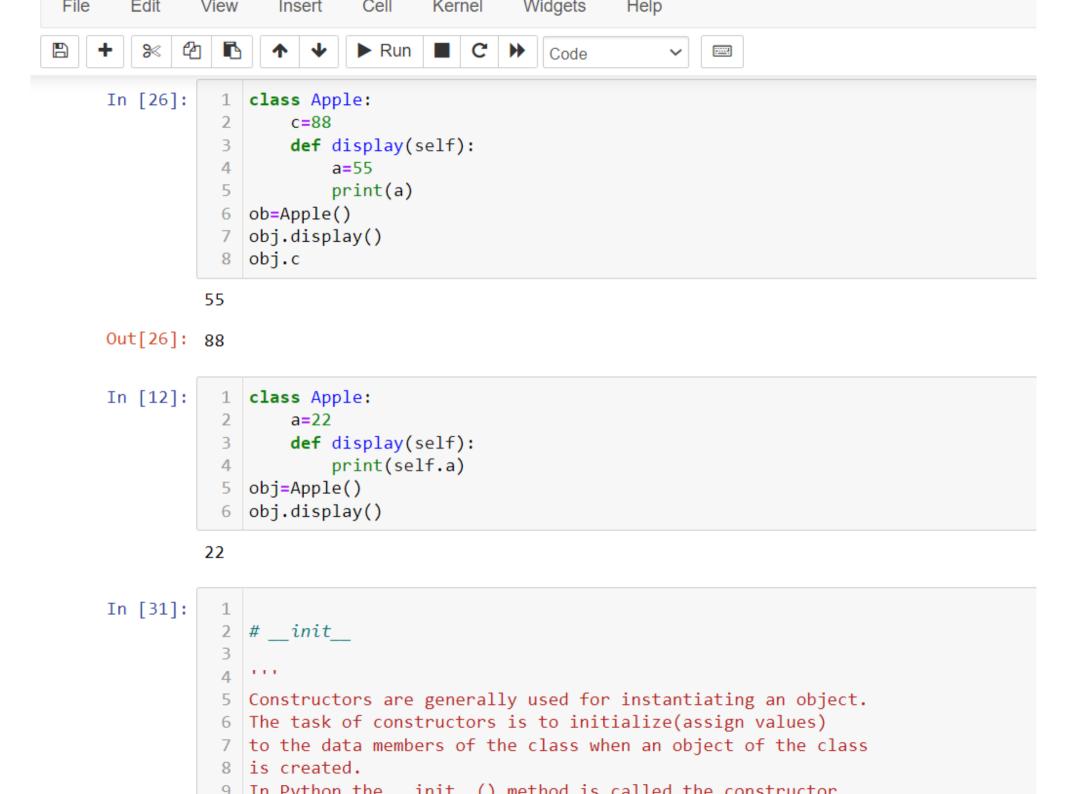
```
In [1]:
             ## oops-concept
             ## (1) class and objects
          3
             # class(template)
          5
             1.we will define class by using 'class'
             2.blue print to create a objects
             3.collection of objects is called class
          9
         10
             #ex fruits
         11
             # object
             # physical entity(real)
             . . .
         14
             1.an instance of a class
         16
             2.memory is created when it declared
         17
         18
             #ex apple, orange, mango
         19
             # attribute (variable) data members
         21
             age=20
             color='blue'
         23
         24
         25
         26
             # method(behaviour) or functions
         28
             eat()
             sleep()
             1.1.1
```

```
2
                 ▶ Run
                                                     [222]
                                   Code
   26
   27
      # method(behaviour) or functions
   28
   29
      eat()
      sleep()
   30
       1 1 1
   31
   32
   33
      # self keyword
       1 1 1
  34
   35
      we can access the attributes and methods of the class(current class only)
   36
      # class Class name:
   37
            #constructor
   38
            #attributes
   39
            #methods
  40
   41
   42
   43
   44
  45
      class Apple:
   46
                                 # class , class name
          print('jai kisan')
                                 # attributes/data members
   47
          def display(self):
                                 #method(self)
   48
               a = 55
   49
               b=66
   50
   51
               print(a,b)
      obj=Apple()
                                #object name, method name()
   52
      obj.display()
                                #declaring object
```

28



```
in [31]:
          1
             # init
          3
          4
            Constructors are generally used for instantiating an object.
             The task of constructors is to initialize(assign values)
             to the data members of the class when an object of the class
             is created.
             In Python the init () method is called the constructor
             and is always called when an object is created
         10
         11
         12
             does'nt support multiple constructor
         13
         14
         15
             class Name:
                                           # class declaration
         16
                 def init (self,a,b,c):
                                             #constructor
                     self.l=a
         17
                               # variables
                     self.m=b
         18
         19
                     self.n=c
                 def display(self):
         20
         21
                     print(self.1)
                     print(self.m)
         22
         23
                     print(self.n)
         24
             obj=Name(25,26,27)
             obj.display()
```

27

```
In [10]:
           1 # # inheritance
           2 # # single parent child
             # # multiple
           4 # # multilevel
           5 # # hierarchical
           6
                 # single-level-inheritance
              class Parent:
                  def display(self):
          10
                      print('this is a parent class')
          11
          12
          13
             class Child(Parent):
                 def display1(self):
          14
                      print('this is a child class')
          15
          16 obj=Child()
             obj.display()
             obj.display1()
```

this is a parent class this is a child class

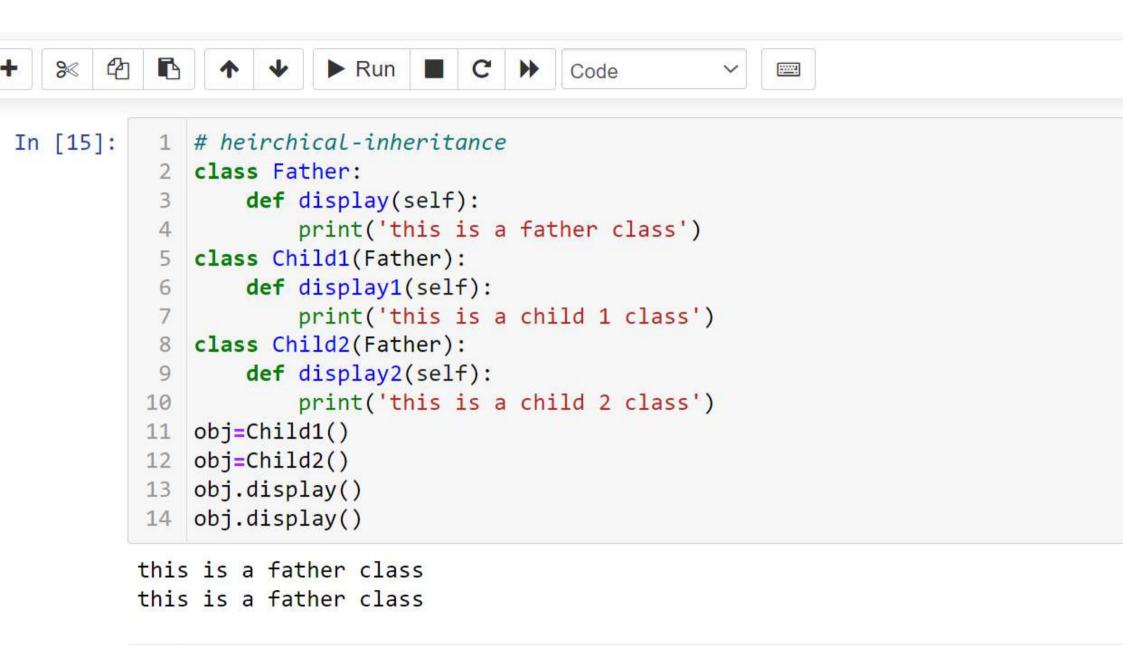
```
In [3]:
          1 ### multilevel-inheritance
            class Grandfather:
                def display(self):
                     print('this is a grand father class')
            class Father(Grandfather):
                def display1(self):
                     print('this is a father class')
            class Child(Father):
                 def display2(self):
                     print('this is a child class')
         10
         11
         12 obj=Child()
         13 obj.display()
         14 obj.display1()
         15 obj.display2()
        this is a grand father class
```

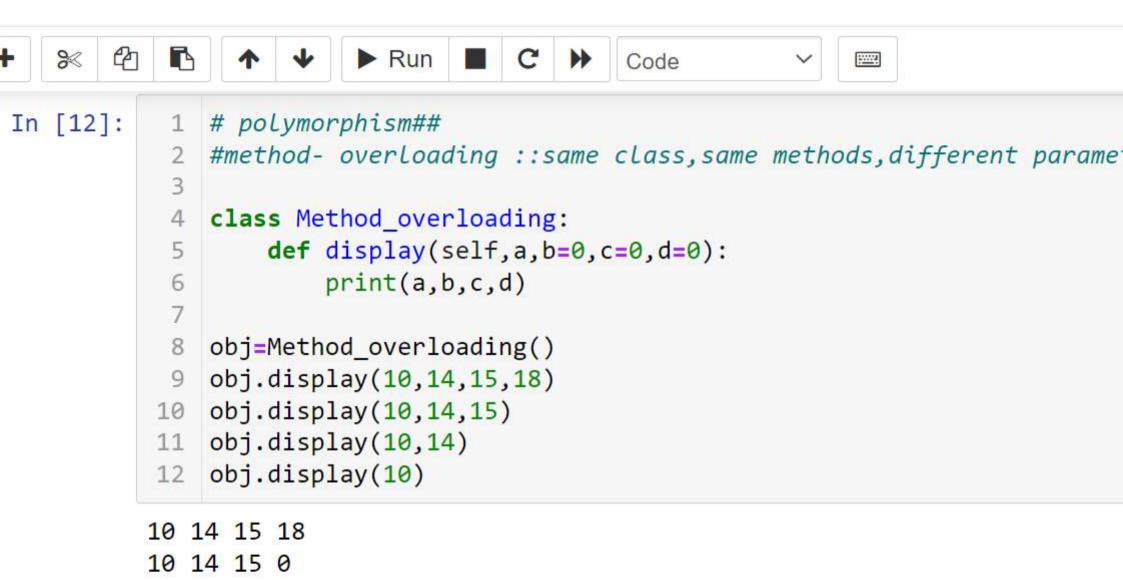
this is a father class

this is a child class

```
In [11]:
              #multiple-inheritance
           3 class Father:
                  def display(self):
                      print('this is a father class')
           5
              class Mother:
                  def display1(self):
                      print('this is a mother class')
              class Child(Father, Mother):
                  def display2(self):
          10
                      print('this is a child class')
          11
          12 | obj=Child()
          13 obj.display()
          14 obj.display1()
          15 obj.display2()
          16
```

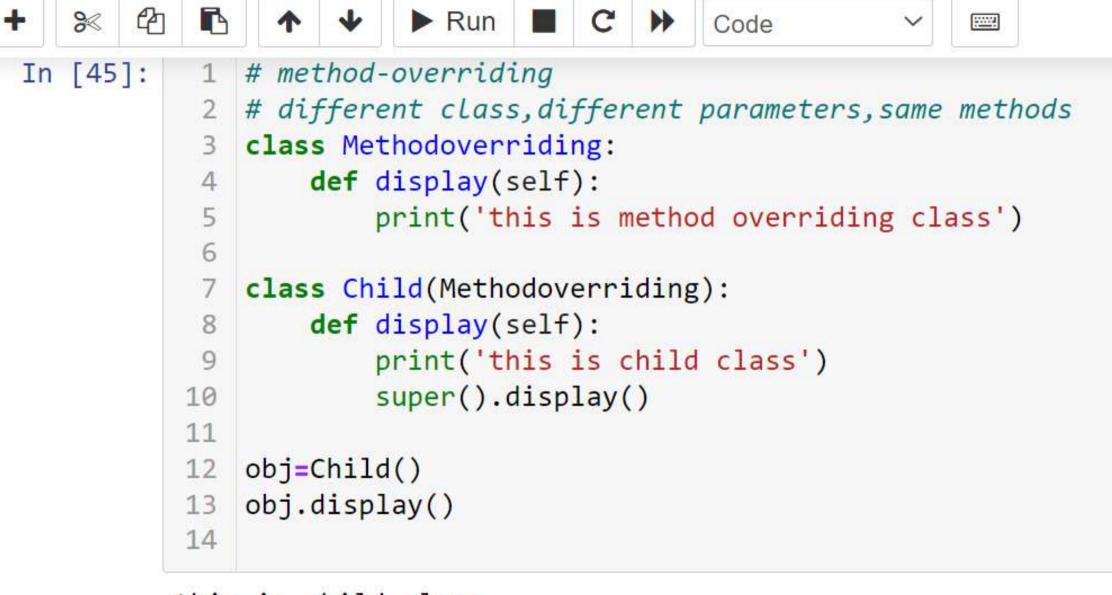
this is a father class this is a mother class this is a child class





10 14 0 0

10 0 0 0



this is child class this is method overriding class

```
[56]:
       1
          #encapsulation
        3
        4
          # binding of class (methods and variables(attributes))
          # public
          # and
        7 # private
          # protected
       9
       10
          ## is protected, they belongs to only family
       11
       12
       13
          class GrandFather:
      14
              def init (self,a):
      15
                   self. x=a
      16
       17
          class Father(GrandFather):
      18
              def display1(self):
                   print(self. x)
       19
       20
          class Child(Father):
       21
              def display2(self):
       22
       23
                   print("child", self. x)
       24
       25
          obj=Child('500 cr')
          obj.display2()
       26
```

child 500 cr

```
4
                    ▶ Run
                                                       ====
                                       Code
58]:
         ## private ,they don't belong to anyone
         class GrandFather:
             def init (self,a):
                  self. x=a
       4
       5
         class Father(GrandFather):
             def display1(self):
                  print(self. x)
       8
       9
         class Child(Father):
      11
             def display2(self):
                  print("child", self. x)
      12
      13
         obj=Child('500 cr')
      15 obj.display2()
     AttributeError
                                                Traceback (most recent call last)
     <ipython-input-58-8787a65aadc9> in <module>
          13
          14 obj=Child('500 cr')
     ---> 15 obj.display2()
     <ipython-input-58-8787a65aadc9> in display2(self)
          10 class Child(Father):
          11
                 def display2(self):
                     print("child", self. x)
     ---> 12
          13
          14 obj=Child('500 cr')
     AttributeError: 'Child' object has no attribute '_Child__x'
```

```
In [6]:
            #abstraction
          1
          2 #abs method there is no body
            #abs class can not create object
             #a class contain one or more abstract methods then it said to be a abc
          5
          6
          7
          8
             from abc import ABC, abstractmethod
          9
             class Parent(ABC):
         10
         11
         12
                 @abstractmethod
         13
         14
                 def display(self):
         15
                     pass
                 def display1(self):
         16
         17
                    print('this is parent')
         18
             class Child(Parent):
         19
                 def display(self,a):
         20
                     print('this is child',a)
         21
         22
         23
            obj=Child()
         24
             obj.display(1200)
         25
            obj.display1()
```

this is child 1200 this is parent