In [ ]:
```
#multiple inheritance
class A:
   def funct1(self):
     print("ur in A")
class B:
   def funct1(self):
      print("ur in B")
class C(A,B):
   def funct1(self):
      print("ur in C")
objc=C()
objc.funct1()
```

```
ur in C
```

In [ ]:
```
#multiple inheritance-init method
class A:
   def __init__(self):
     print("init of A")
   def funct1(self):
     print("ur in A")
class B(A):
   def funct2(self):
      print("ur in B")
objc=B()
```

```
init of A
```

In [ ]:
```
#not invoking init method of parent
class Parent:
    def __init__(self):
        self.parent_attribute = 'I am a parent'
    def parent_method(self):
        print('parent class')
# Create a child class that inherits from Parent
class Child(Parent):
    def __init__(self):
        self.child_attribute = 'I am a child'
# Create instance of child
child = Child()
# Show attributes and methods of child class
print(child.child_attribute)
print(child.parent_attribute)
child.parent_method()
```

```
I am a child

---------------------------------------------------------------------------
----
AttributeError                            Traceback (most recent call l
ast)
<ipython-input-2-3b08e731fe08> in <module>()
     12 # Show attributes and methods of child class
     13 print(child.child_attribute)
---> 14 print(child.parent_attribute)
     15 child.parent_method()

AttributeError: 'Child' object has no attribute 'parent_attribute'
```

In [ ]:
```
#one way to invoke parent init
class Parent:
    def __init__(self):
        self.parent_attribute = 'I am a parent'
    def parent_method(self):
        print('parent class')
# Create a child class that inherits from Parent
class Child(Parent):
    def __init__(self):
        Parent.__init__(self)
        self.child_attribute = 'I am a child'
# Create instance of child
child = Child()
# Show attributes and methods of child class
print(child.child_attribute)
print(child.parent_attribute)
child.parent_method()
```

```
I am a child
I am a parent
parent class
```

In [ ]:
```python
#using super()
class Parent:
    def __init__(self):
        self.parent_attribute = 'I am a parent'
    def parent_method(self):
        print('parent class')
# Create a child class that inherits from Parent
class Child(Parent):
    def __init__(self):
        super().__init__()
        self.child_attribute = 'I am a child'
# Create instance of child
child = Child()
# Show attributes and methods of child class
print(child.child_attribute)
print(child.parent_attribute)
child.parent_method()
```

```
I am a child
I am a parent
parent class
```

In [ ]:
```python
#multiple inheritance
class B:
    def b(self):
        print('b')
class C:
    def c(self):
        print('c')
class D(B, C):
    def d(self):
        print('d')
d = D()
d.b()
d.c()
d.d()
```

```
b
c
d
```

In [ ]:
```
#multiple resolution order(MRO)
class B:
    def x(self):
        print('x: B')
class C:
    def x(self):
        print('x: C')
class D(B, C):
    pass
d = D()
d.x()
print(D.mro())
```

```
x: B
[<class '__main__.D'>, <class '__main__.B'>, <class '__main__.C'>, <cla
ss 'object'>]
```

In [ ]:
```
class First():
  def __init__(self):
    print ("First(): entering")
    super().__init__()
    print ("First(): exiting")

class Second():
  def __init__(self):
    print ("Second(): entering")
    super().__init__()
    print ("Second(): exiting")

class Third(First, Second):
  def __init__(self):
    print ("Third(): entering")
    super().__init__()
    print ("Third(): exiting")
print(Third.mro())
t1=Third()
```

```
[<class '__main__.Third'>, <class '__main__.First'>, <class '__main__.S
econd'>, <class 'object'>]
Third(): entering
First(): entering
Second(): entering
Second(): exiting
First(): exiting
Third(): exiting
```

In [ ]:

```
# Base class
class Parent:
      def func1(self):
          print("This function is in parent class.")
# Derived class1
class Child1(Parent):
      def func2(self):
          print("This function is in child 1.")
# Derivied class2
class Child2(Parent):
      def func3(self):
          print("This function is in child 2.")

object1 = Child1()
object2 = Child2()
object1.func1()
object1.func2()
object2.func1()
object2.func3()
```

```
This function is in parent class.
This function is in child 1.
This function is in parent class.
This function is in child 2.
```