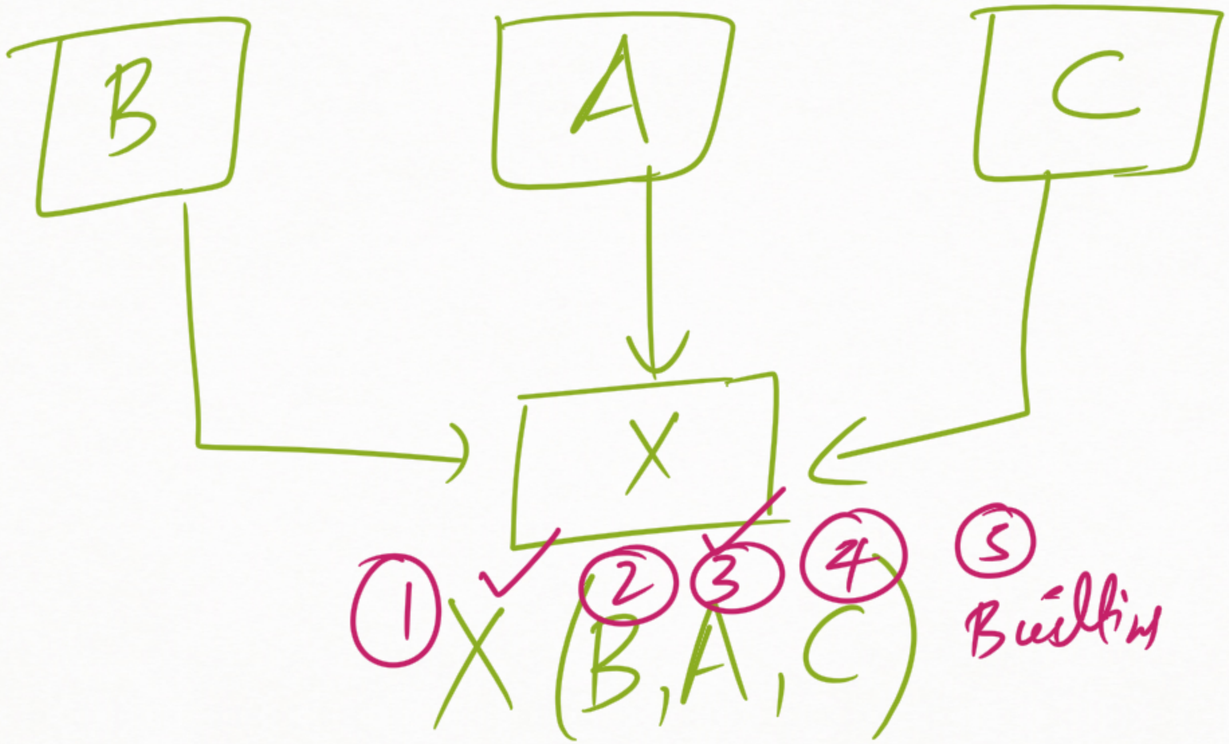
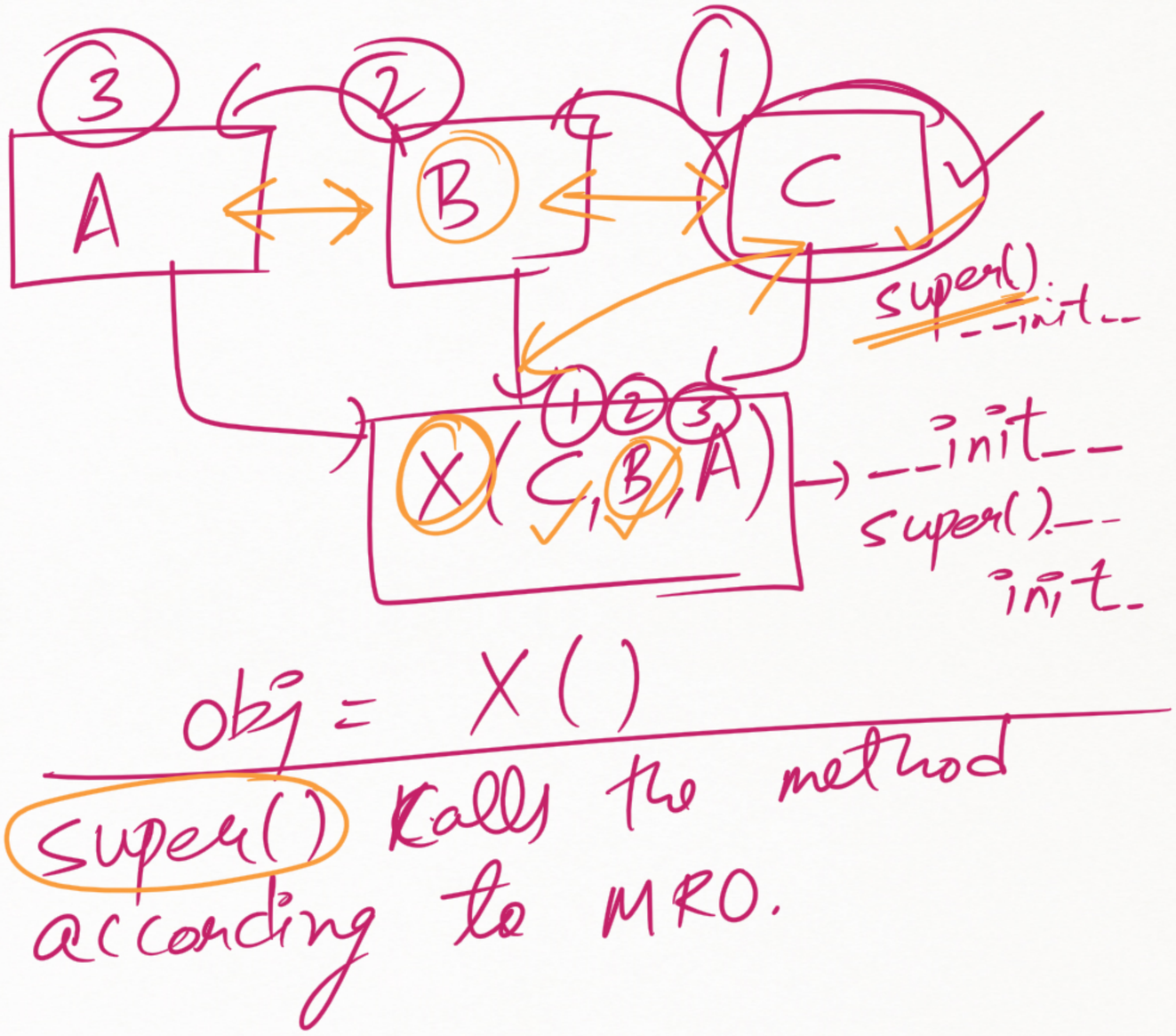


If you search for a method in X it will be searched Accordingly to MRO until it finds that method

In [63]:
```
## MULTIPLE INHERITANCE

class A:
    def __init__(self):
        print('inside A')
        self.A = 'Apple'

    def fun1(self):
        print('fun inside A')

class B:
    def __init__(self):
        print('inside B')
        self.B = 'Blackboard'
        super().__init__()

    def fun(self):
        print('fun inside B')

class C:
    def __init__(self):
        print('inside C')
        self.C = 'Camera'
        super().__init__()

    def fun1(self):
        print('fun inside C')

class X(C,B,A):
    def __init__(self):
        print('inside X')
        super().__init__()

    def fun(self):
        print('fun inside X')
```

In [64]:
```
obj = X()
```



```
[73]: class Z:
          def __init__(self):
              print('inside Z')

          def fun(self):
              print('fun of Z')

      class M(Z):
          def __init__(self):
              print('inside M')
          def fun(self):
              print('fun of M')

      class N(Z):
          def __init__(self):
              print('inside N')
          def fun(self):
              print('fun of N')

      class A(M):
          def fun(self):
              print('fun inside A')

      class B(N):
          def __init__(self):
              print('inside B')
          def fun(self):
              print('fun inside B')

      class C(A,B):
          def __init__(self):
              print('inside C')
              super().__init__()

          def fun(self):
              print('fun inside C')
```

```
[74]: objRef = C()
```