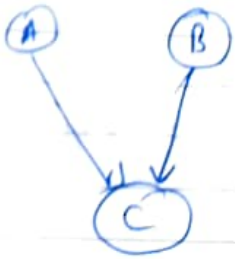
MRO

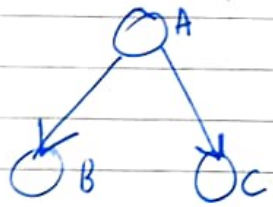
Single Inheritance

Multiple

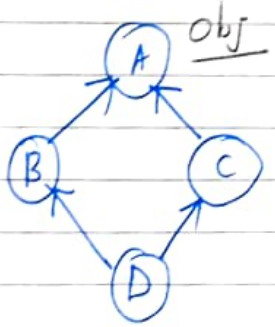


Multilevel

Hierarchical Inheritance



Q.



object class is the parent class.

A (mro) → A → object

B (mro) → B → A → "

C (mro) → C → A → "

D (mro) → D → B → C → A → O

Q.

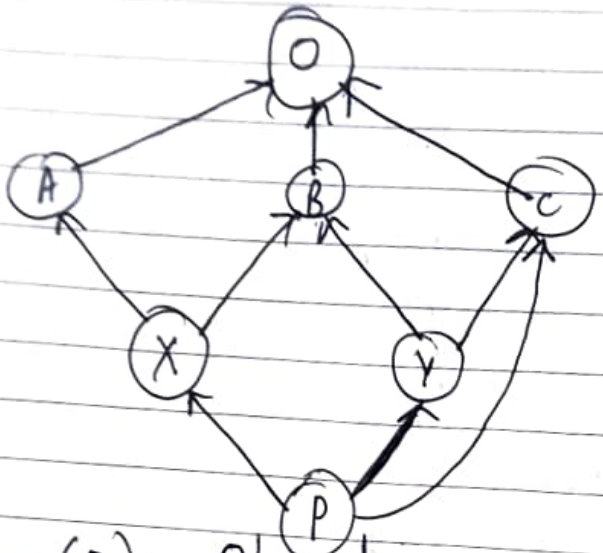
```
class P:
    def m1(self):
        print(Parent Method)

class C(P): pass
```

print (C.mro())

print (P.mro())

Q



mro(O) = Object

mro(B) = B, O → 3rd level 1 up

mro(A) = A, O ⇒ Use a formulae

mro(X) = X, A, B, O

mro(Y) = Y, B, C, O

mro(P) = P, X, Y, C, A, B, O ?

Wrong if you execute this code ...

C3 Algo. → Use it

Do not change order always → L → R.

3 parents

mro(P) = P + Merge (mro(X), mro(Y), mro(C), XYC)

parent list

mro(P) ⇒ P + Merge (XABO, YBCO, CO, XYC)

⇒ P + X + Merge (ABO, YBCO, CO, YC)

⇒ P + X + A + Merge (BO, YBCO, CO, YC)

leave as all options are explored

Ex. ABCDEF

Head Element → A

Tail " → BCDEF

⇒ P + X + A + Y + Merge again from 1st list ← CBO, BCO, CO, C)

⇒ P + X + A + Y + B + Merge (O, CO, CO, C)

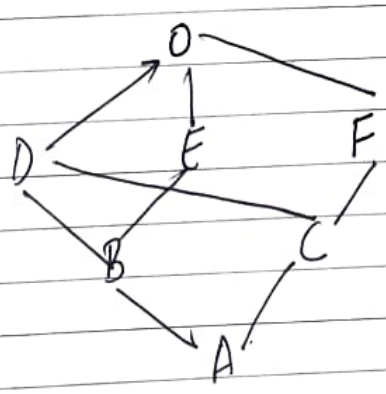
P + X + A + Y + B + C + Merge (O, O, O)

→ If first list head ele not present in the tail part of any other list then consider that ele. in the result & remove that ele. from all the lists

Final Answer ⇒ P + X + A + Y + B + C + O

⇒ PXAYBCO

Q)



mro(O) = O

mro(D) = D, O

mro(E) = E, O

mro(F) = F, O

mro(B) = BDEO

mro(C) = CDFO

mro(A) = A + Merge (BDEO, CDFO, BC)

mro

mro

=> A + merge (mrv(B), mrv(C), BC)

=> A + merge (BDEO, CDFO, BC)

=> A + B + merge (DEO, CDFO, C)

So ignore & move to the next list

=> A + B + C + merge (DEO, DFO)

=> A + B + C + D + merge (EO, FO)

=> A + B + C + D + E + merge (O, FO)

=> ABCDE + F + Merge (O, O)

=> ABCDEFO