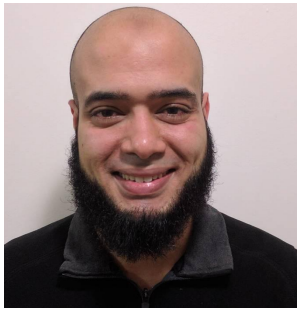
Python Programming

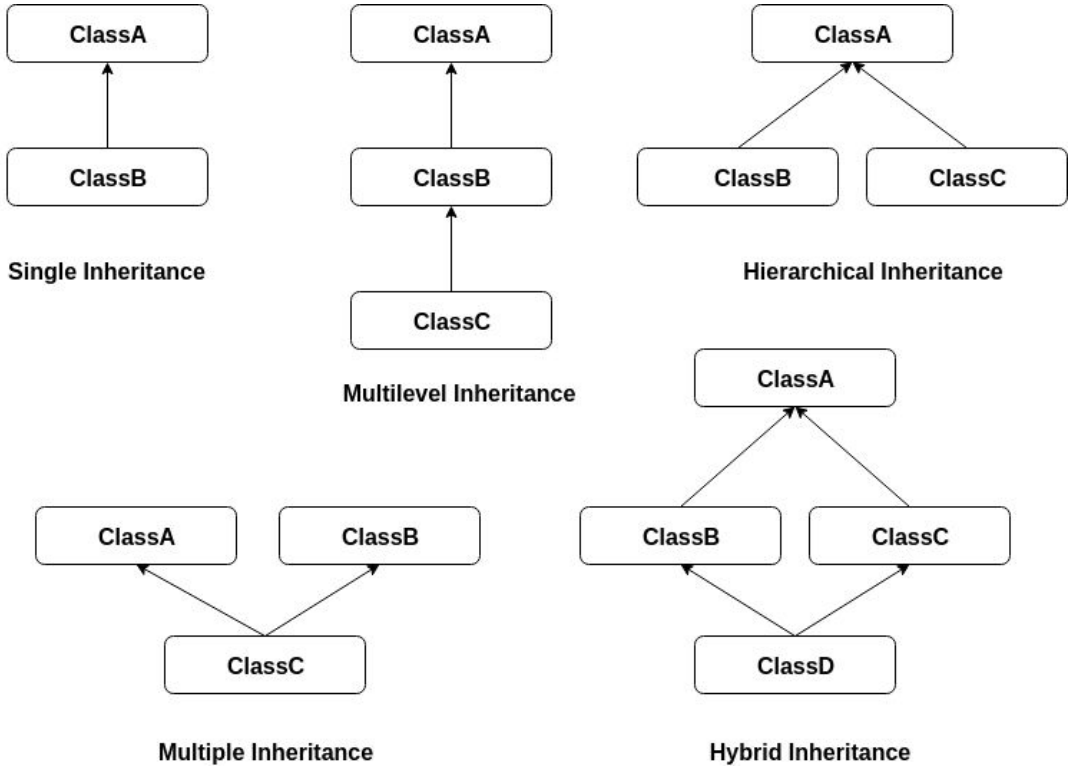
Inheritance
Multiple Inheritance

Mostafa S. Ibrahim
Teaching, Training and Coaching since more than a decade!

Artificial Intelligence & Computer Vision Researcher
PhD from Simon Fraser University - Canada
Bachelor / Msc from Cairo University - Egypt
Ex-(Software Engineer / ICPC World Finalist)

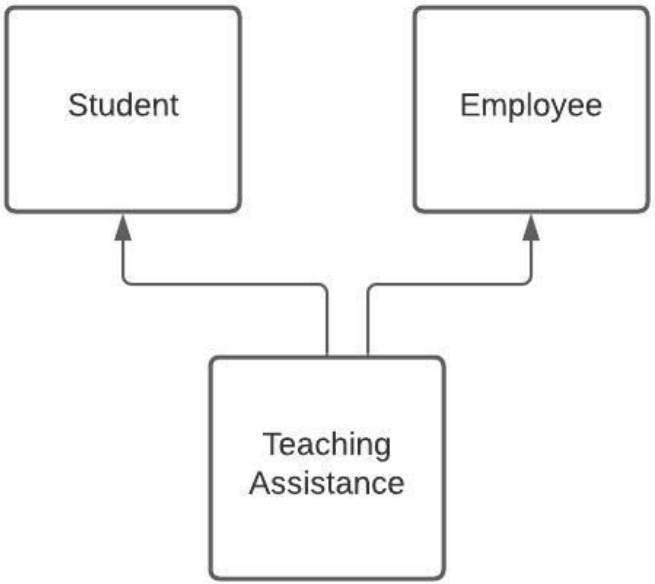


5 Inheritance relations types



Img Src

Multiple Inheritance



Basic multiple inheritance

```python
class ParentA:
    def __init__(self, a):
        self.a = a
        print('init ParentA')

    def fA(self):
        print('fA')
```

```python
class ParentB:
    def __init__(self, b):
        self.b = b
        print('init ParentB')

    def fB(self):
        print('fB')
```

```python
class ChildC(ParentA, ParentB):
    def __init__(self, a, b, c):
        ParentA.__init__(self, a)
        ParentB.__init__(self, b)
        self.c = c
        print('init ChildC')

    def fC(self):
        print('fC')
```

```python
if __name__ == '__main__':
    c = ChildC(1, 3, 5)
    c.fA()
    c.fB()
    c.fC()
```

```
init ParentA
init ParentB
init ChildC
fA
fB
fC
```

Same function name

- If you are language designer, how to solve this confusion?
- What might be the answer?

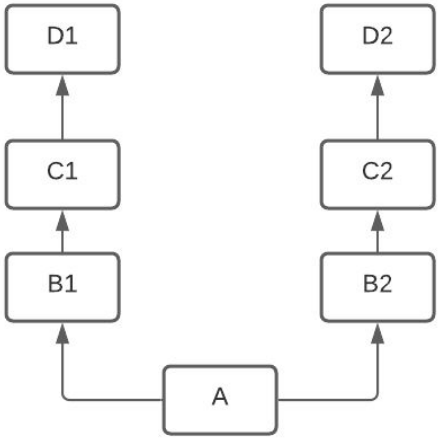
```
class ParentA:
    def f(self):
        print('ParentA')

class ParentB:
    def f(self):
        print('ParentB')

class ChildC1(ParentA, ParentB):
    pass

class ChildC2(ParentB, ParentA):
    pass


ChildC1().f()   # ??
ChildC2().f()   # ??
```

Method resolution order (MRO)

- A graph algorithm is used to find a proper ordering (C3 linearization).
  - In the previous case: we depends on the parents order left to right
- In complex hierarchy, things get complicated :(

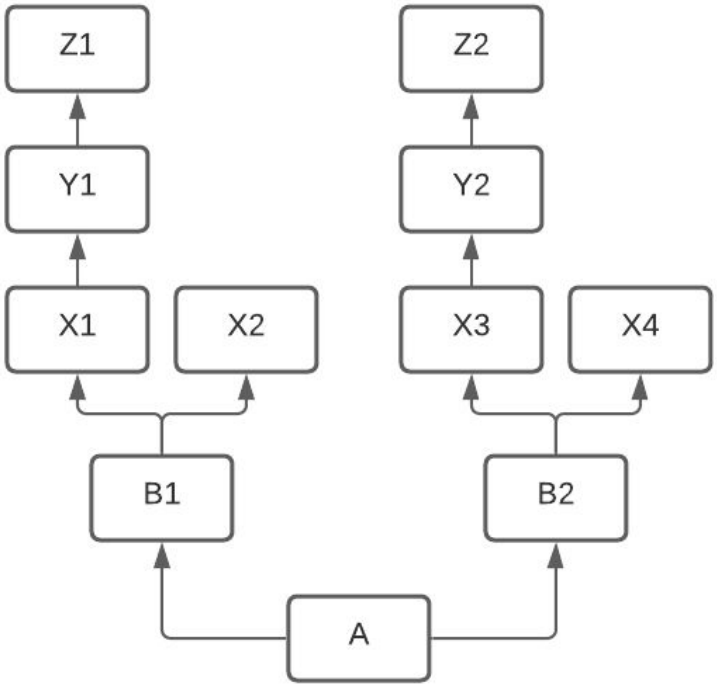
```
class ChildC2(ParentB, ParentA):
    pass

if __name__ == '__main__':
    print(ChildC1.__mro__)  # (ChildC1, ParentA, ParentB, object)

    print(ChildC2.__mro__)  # (ChildC2, ParentB, ParentA, object)
    ChildC2().f()           # ParentB is the left one
```

MRO in multilevel



- What is the classes order of A.__mro__?
- A, B1, C1, D1, B2, C2, D2
  - It ends with object - skip for now
- Assume Classes: C1, C2, and B2 has method F
  - Let's call A().f(), which will be called?
  - C1 as it appeared first!
  - *Note: code is provided*
- Useful rules for mro:
  - Child class comes before its parents
  - For multiple parents: order left to right (of inheriting)
  - In multilevel:
    - Finish every branch in order from child to parent

MRO in multilevel and **Simple** multiple inheritance



- What is the classes order of A.__mro__?
- A, B1, X1, Y1, Z1, X2, B2, X3, Y2, Z2, X4
- Let's view as a hierarchy
- A,
  - B1,
    - X1,
      - Y1, Z1
    - X2,
  - B2,
    - X3,
      - Y2, Z2
    - X4
- In graph they it will make sense with DFS

MRO Exceptions

- If MRO couldn’t find consistent order, relative to its current algorithm, it will fail
- If you got this error, u typically is doing nonsense in the hierarchy
- 1) Draw it
- 2) Spot what is weird

```
class A:
        print('init A')

class B(A):
    def __init__(self):
        print('init B')

class C(A, B):
    def __init__(self):
        print('init D')


C()
"""
TypeError: Cannot create a consistent
method resolution order (MRO) for bases A, B


Note: class C(B, A):
    will work
"""
```

More

- In practice, we avoid multiple inheritance
  - One more video about super with inheritance
- So learn the concept and play with it, but don’t dig deep
- MRO is using an algorithm named C3 Linearization
  - It determines the order in complex hierarchies
  - You may understand it when you study graph theory, but no such big need

"Acquire knowledge and impart it to the people."

"Seek knowledge from the Cradle to the Grave."