Agenda

- Revision
- Class Variable vs Instance Variable
- Other Dunder/Magic Methods
- Inheritance
- Private Variables
- Multiple Inheritance
- MRO

Revision

Instructor Details

- WhatsApp Number: 8586836627
- Git Link : https://github.com/SachinScaler/July24Python2

class
↳ objects
↓
instance Variables

__init__ → Used for creating instance Variable

self → parameter to store instance

a1 = A(1, 2)

How can we access Instance Variables

① Inside Class

self.inst_Variable

② Outside Class

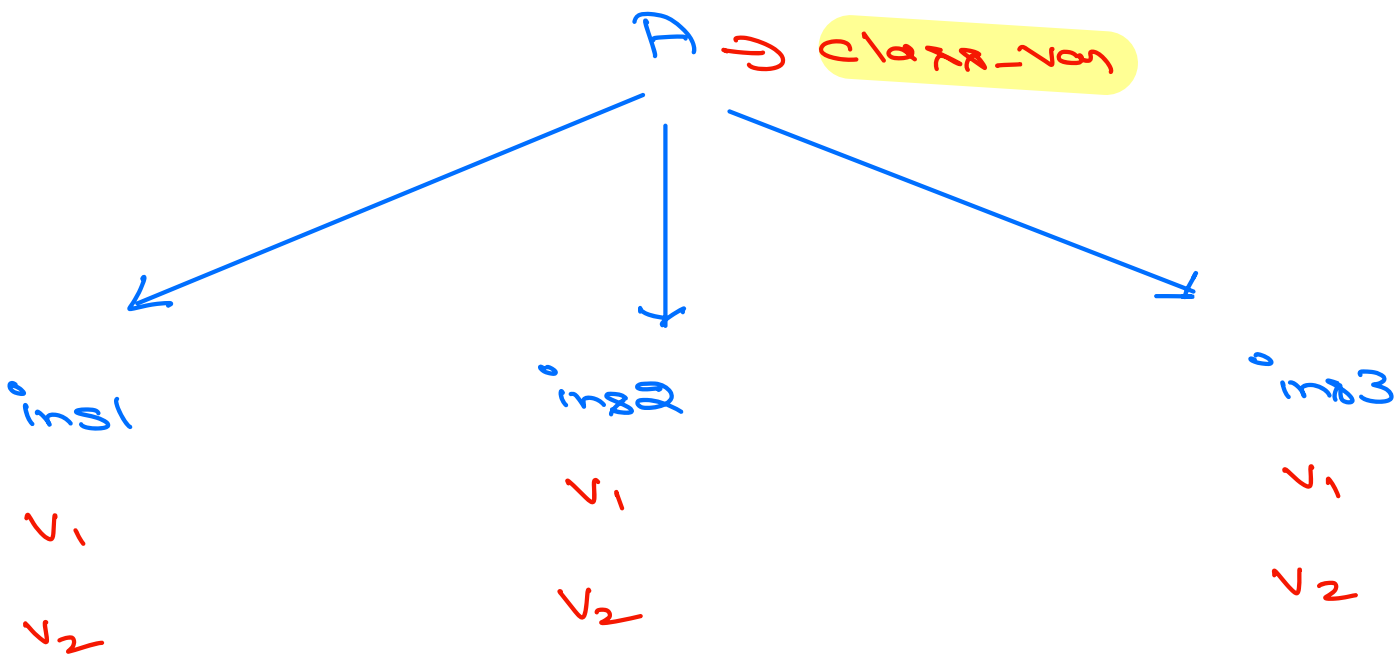
instance_name.inst_Variable

Class Variable vs Instance Variable

Class/Static Variable:

- Is Common Variable for all the instances
- Accessed
  - classname.Variable_name
    or
  - self/instance_name.Var_name

```
class A
    class_var = value
    def __init__(self, v1, v2)
        self.v1 = v1
        self.v2 = v2
```



Dunder / Magic Methods

__init__

* __ are called Dunders
* Methods starting with __ are also known as magic methods since they serve special purpose

Ex:

① __str__ :

1) Change the way print function works

② __add__ : +

③ __lt__ : a < b

④ __call__ : A()

* Full List can be viewed with dir(int)

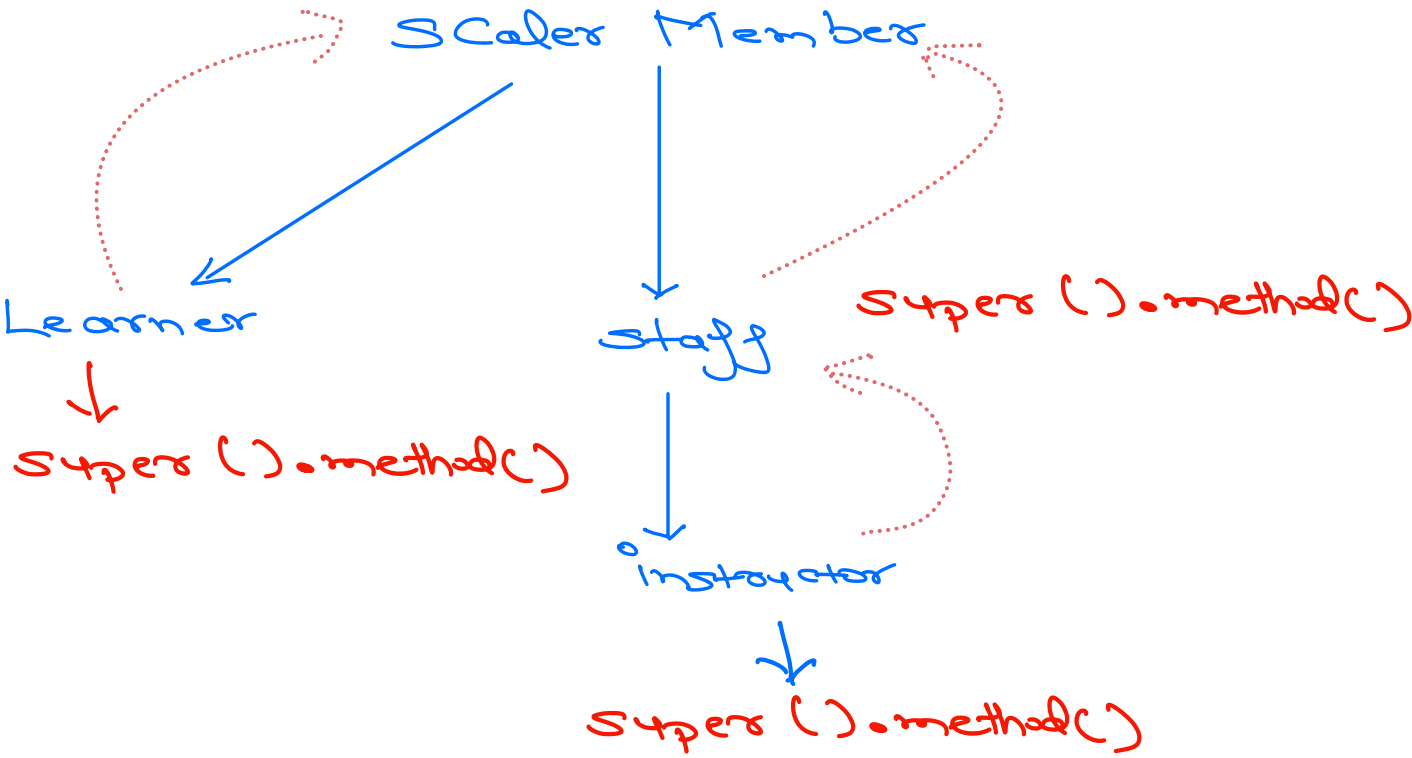
Inheritance

* Child inherits parents Public class Variable and Methods
* In case of same name of Variable or method present in both, child object gives preference to its own.

```
class A:
    pass

class B(A):
    pass
```

* B is child of A



* How to call parents methods from inside of child ?

① super().methodname()

→ Calls immediate parent

② Parent.methodname(instance)

Private Properties

→ Access Specifier

Pending

① Multiple Inheritance

↓

② MRO Method Resolution Order

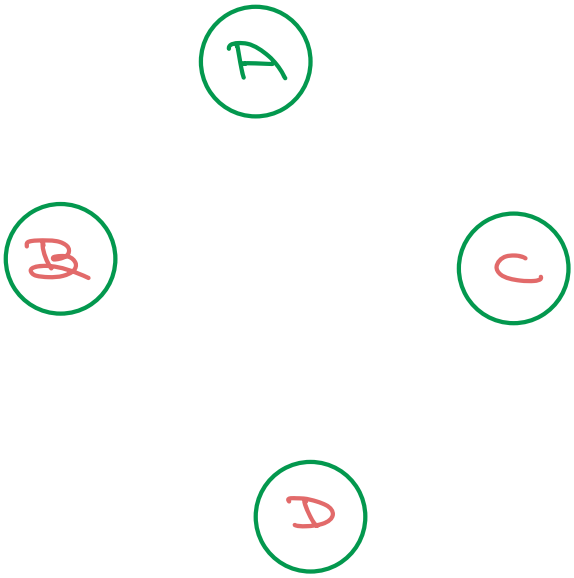
- Can't be accessed outside the Class
- Not inherited

But why?

How?

Multiple Inheritance

MRO



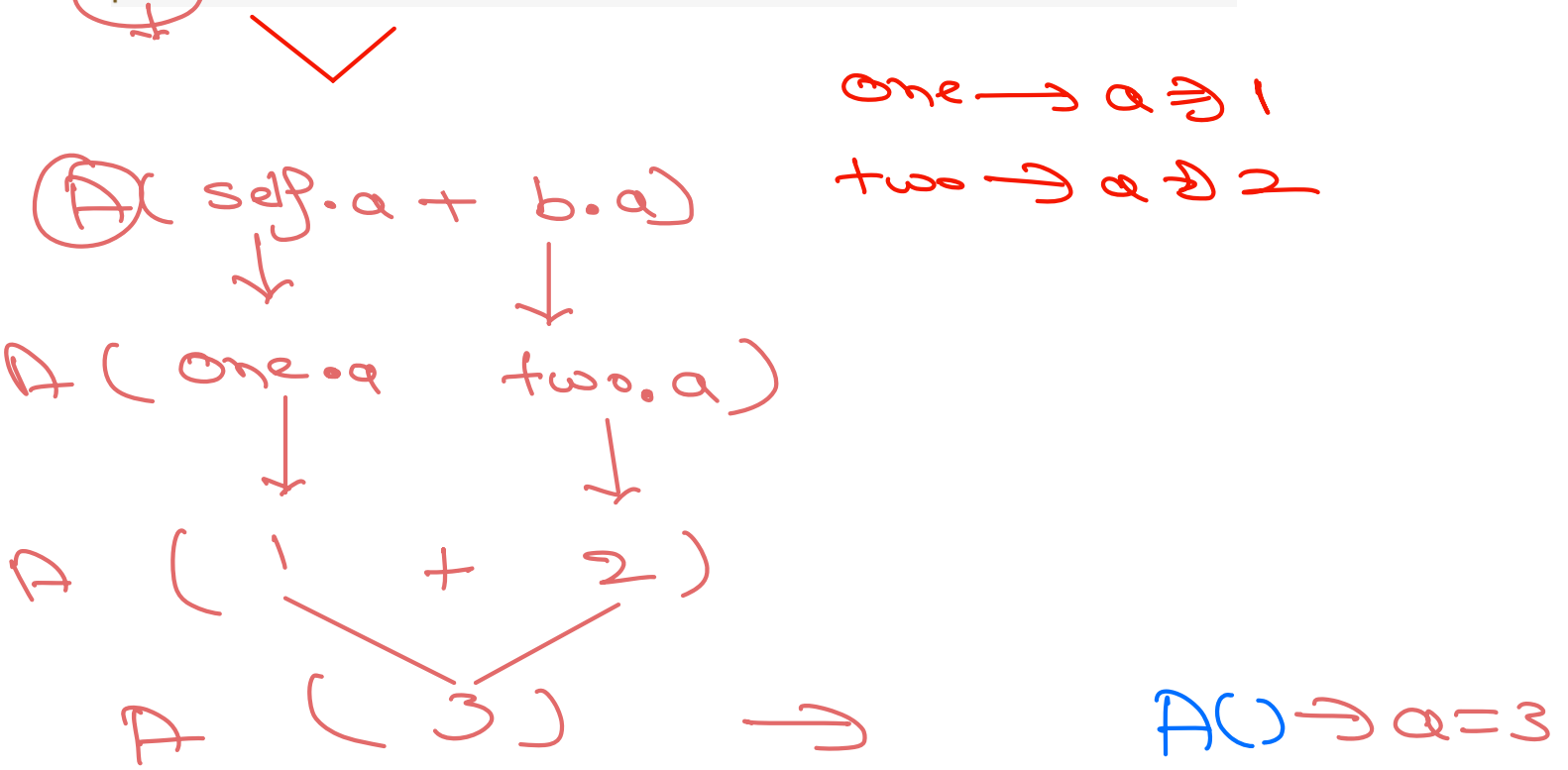
In Python : classname.__mro__

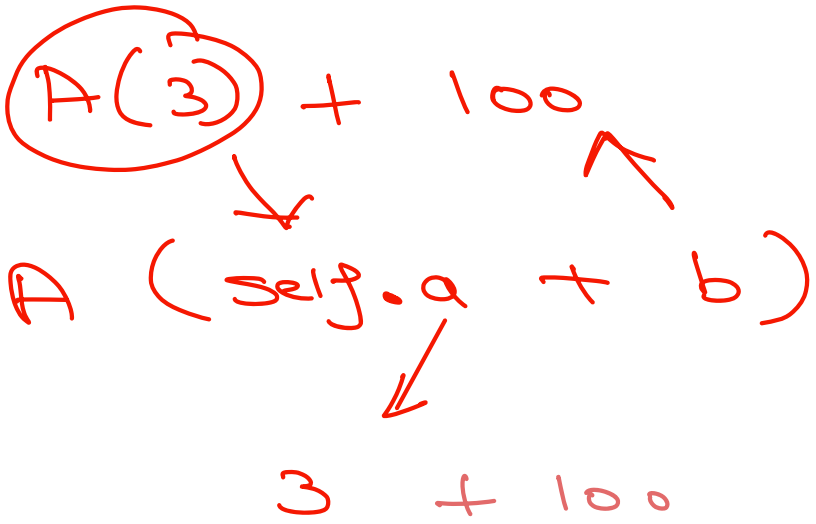
Rules for MRO

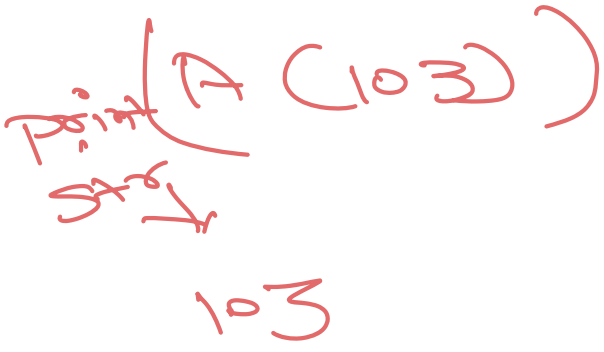
```python
class A:
  def __init__(self, a):
    self.a = a
  def __add__(self, b):
    if isinstance(b, int):
      return A(self.a + b) # ins3 + int
    return A(self.a + b.a) # ins3 = ins1 + ins2
  def __str__(self):
    return f"{self.a}"

one = A(1)
two = A(2)

# ins1 + ins2 + int
print(one + two + 100)
```




A(3) + 100
A (self.a + b)
3 + 100


Print(A(103))
103