Data Science & AI & NIC - Param

Python-For Data Science

OOPs

Lecture No.- 04

Recap of Previous Lecture

Topic Object-Oriented Programming Part -03

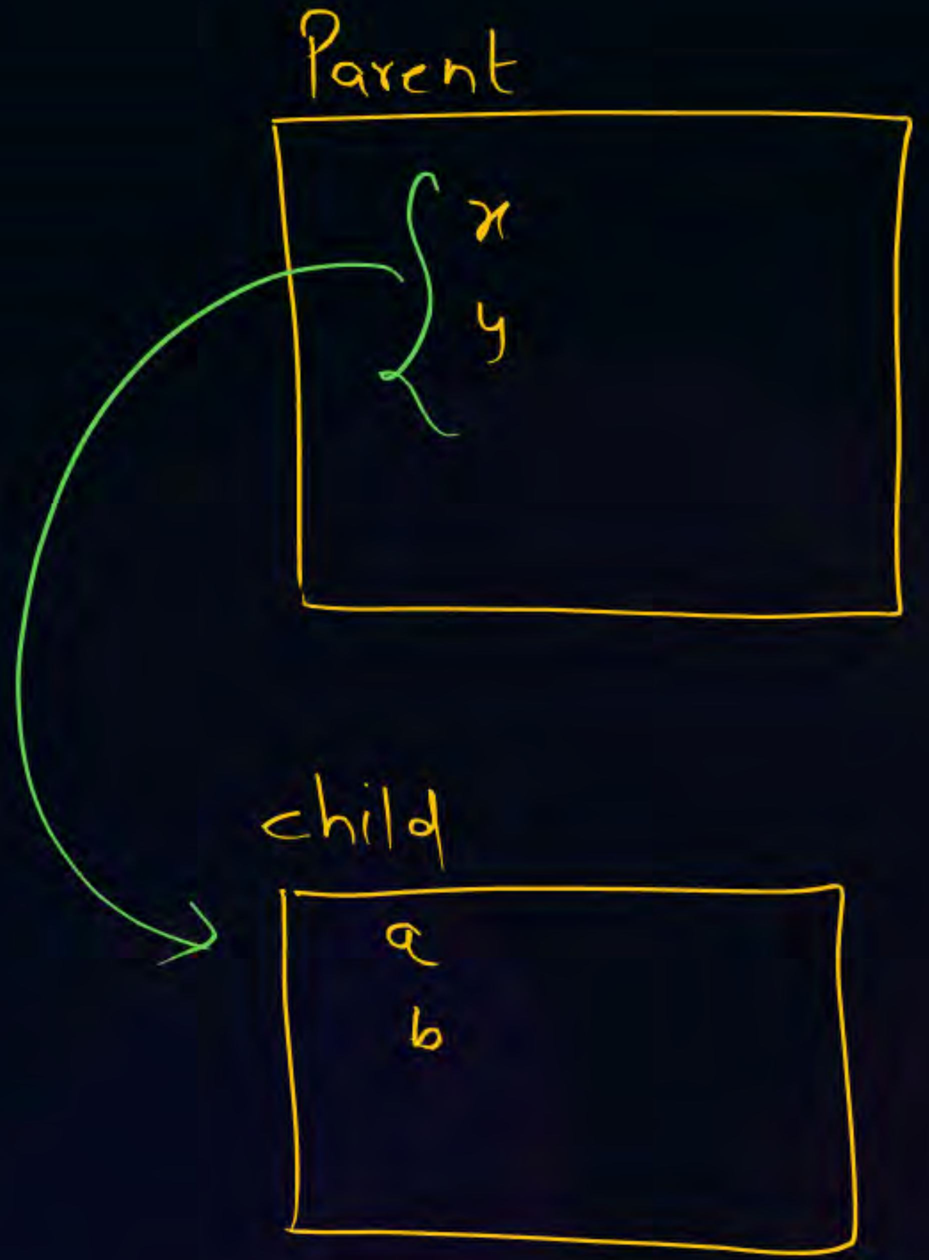
inheritance

Topics to be Covered

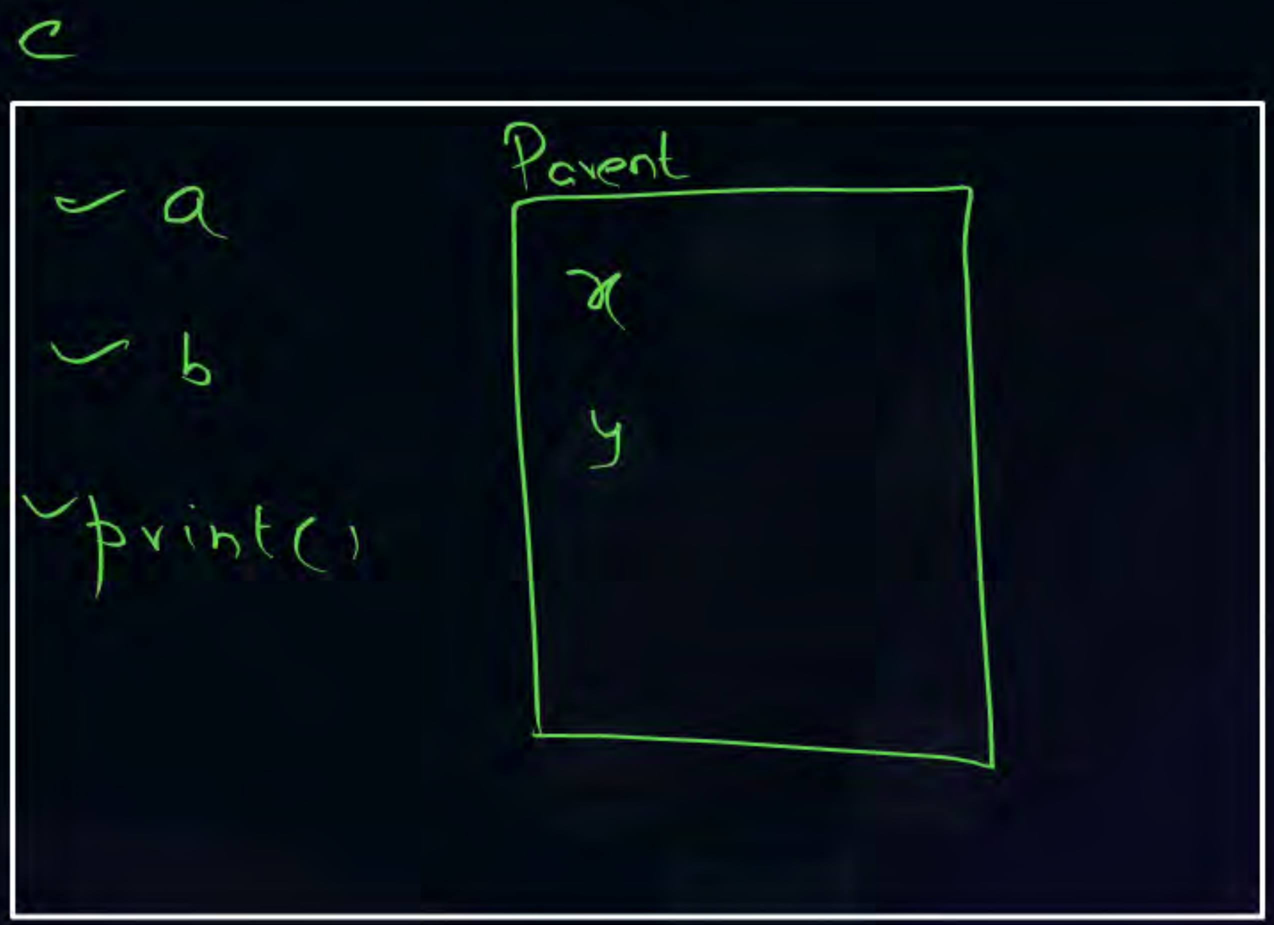
Topic Object-Oriented Programming Part -04

Topic : Object-Oriented Programming

inheritance

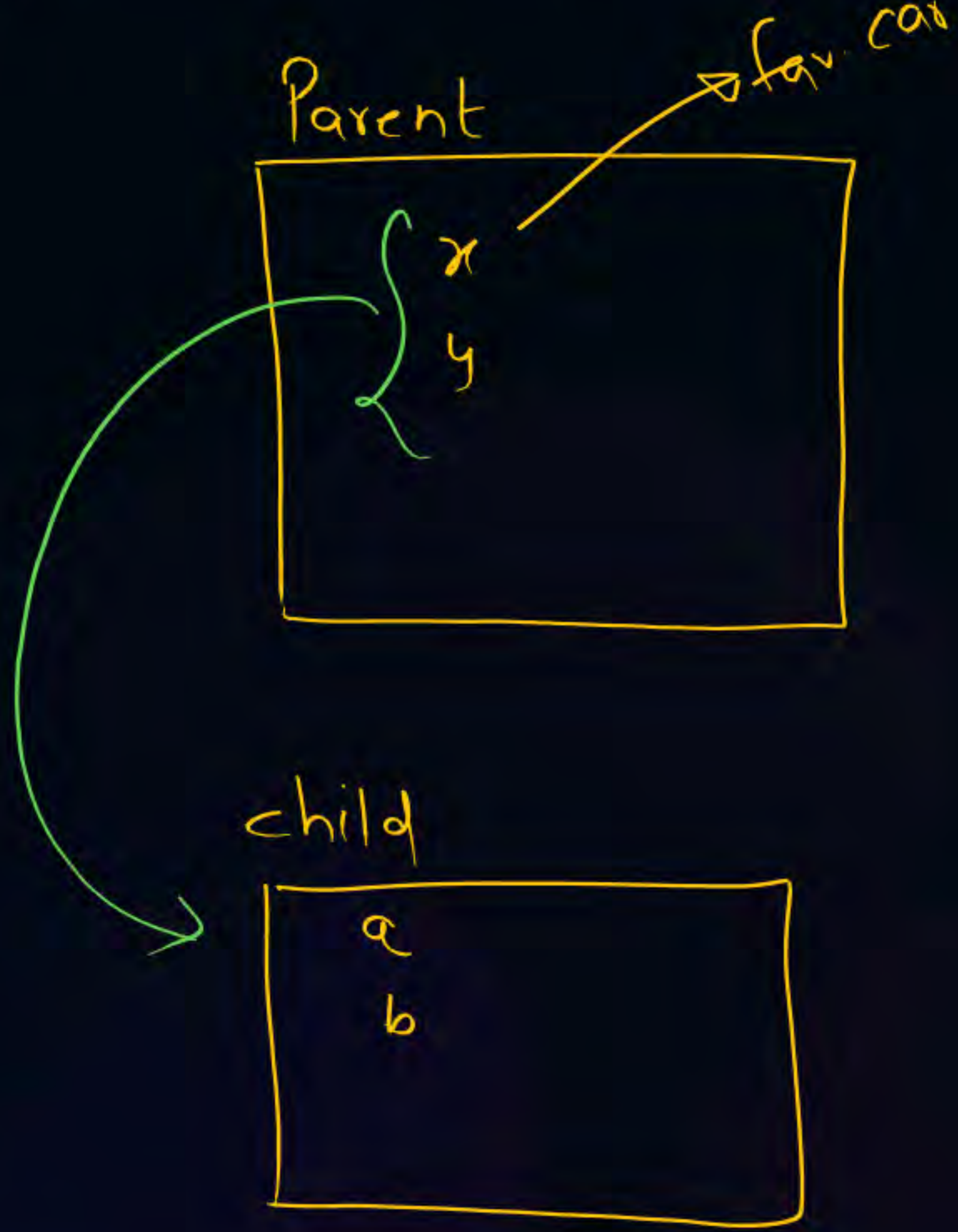


C = child( )

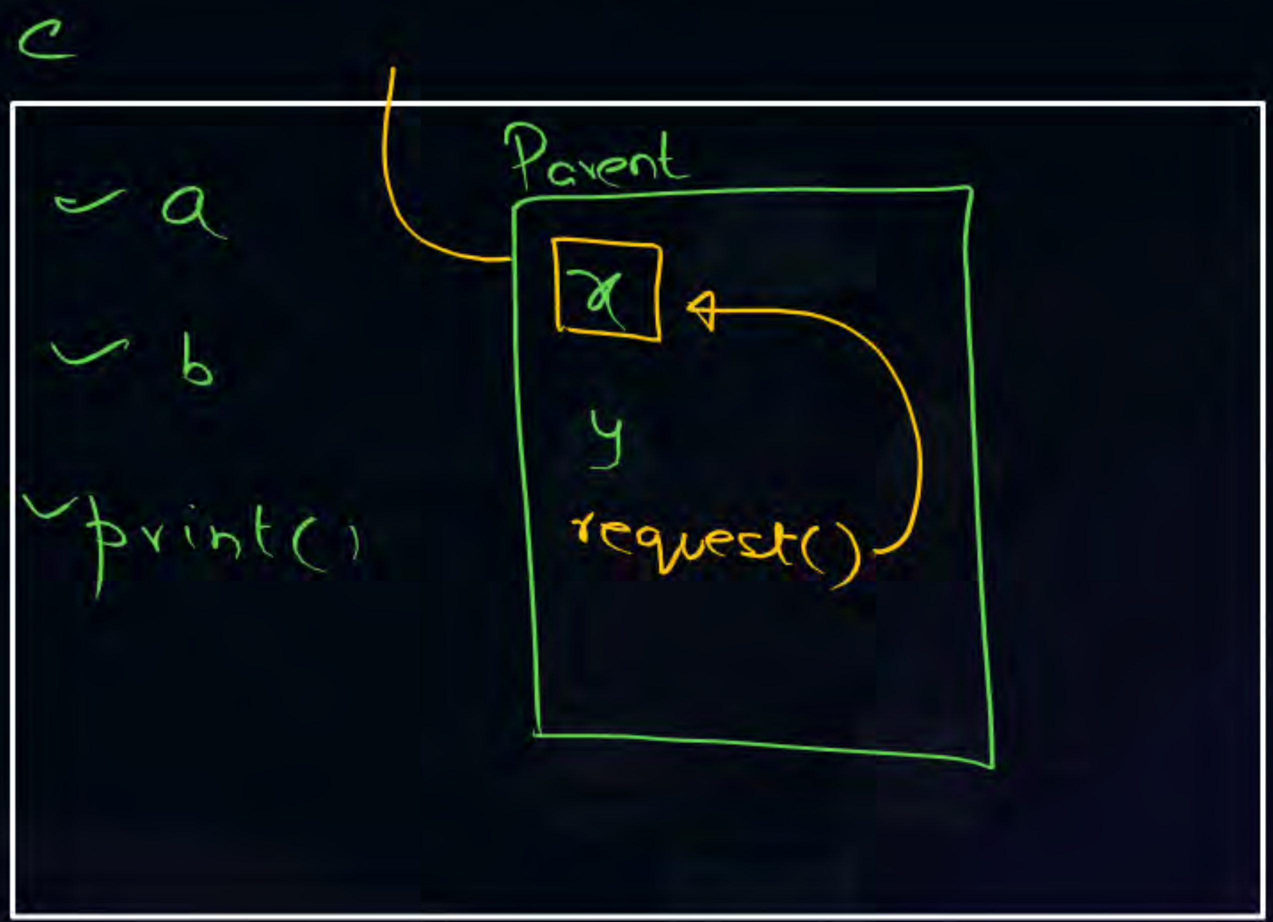


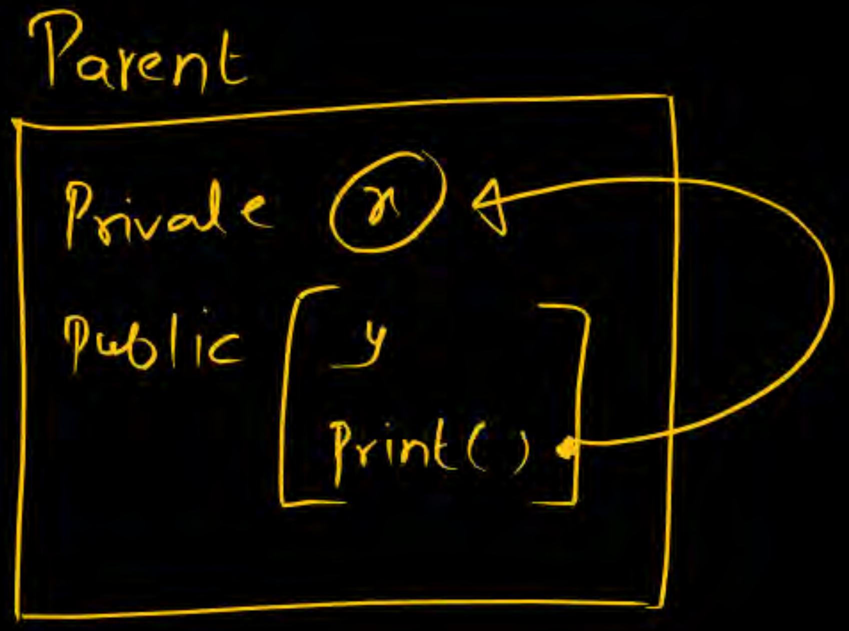
Topic : Object-Oriented Programming

inheritance

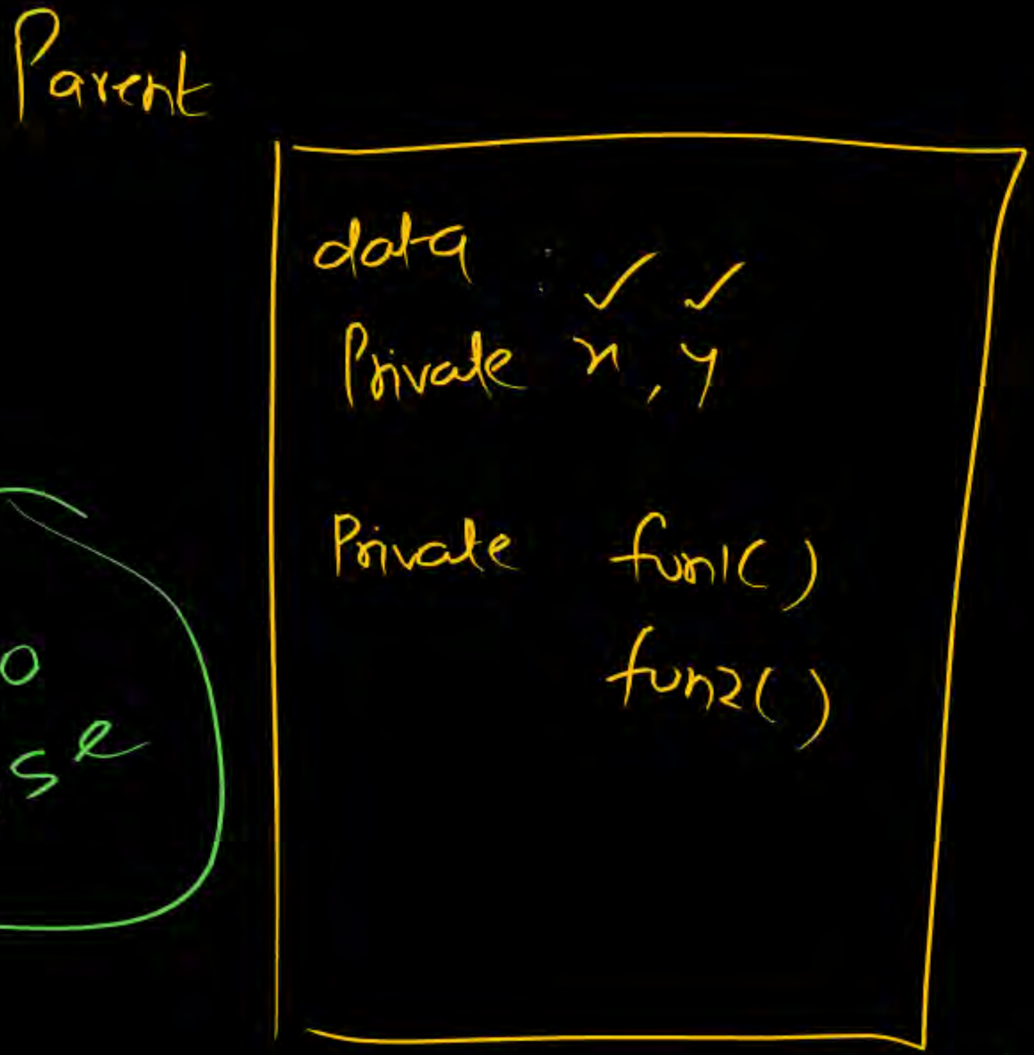


c = child( )



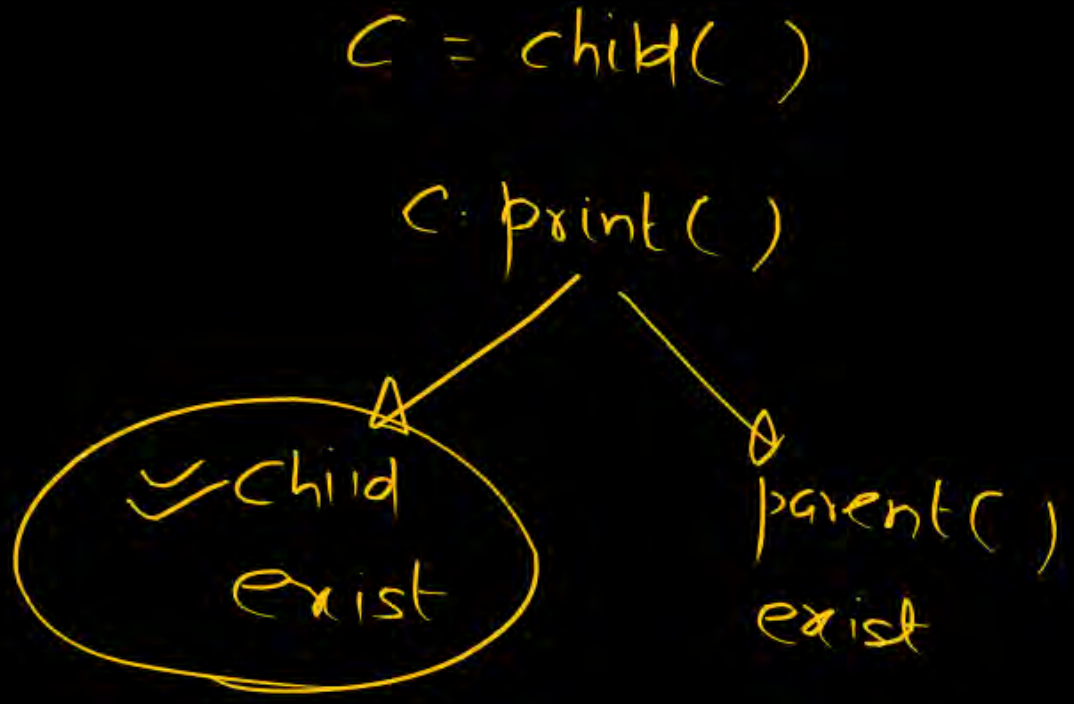


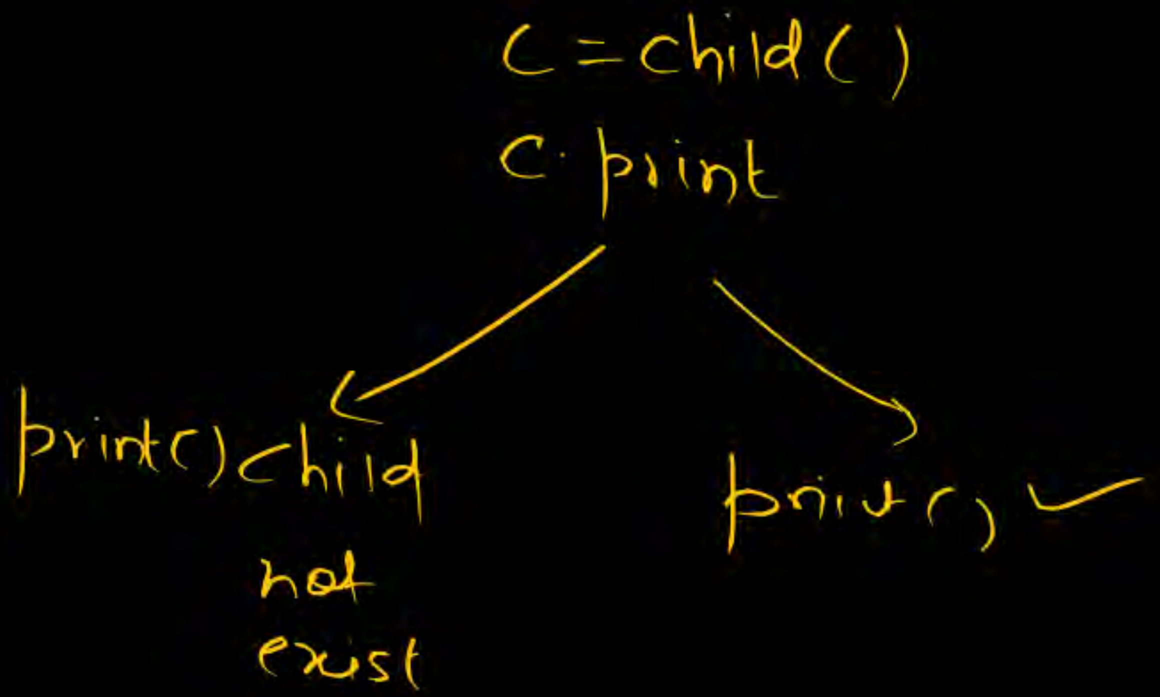
ObjPar . x =



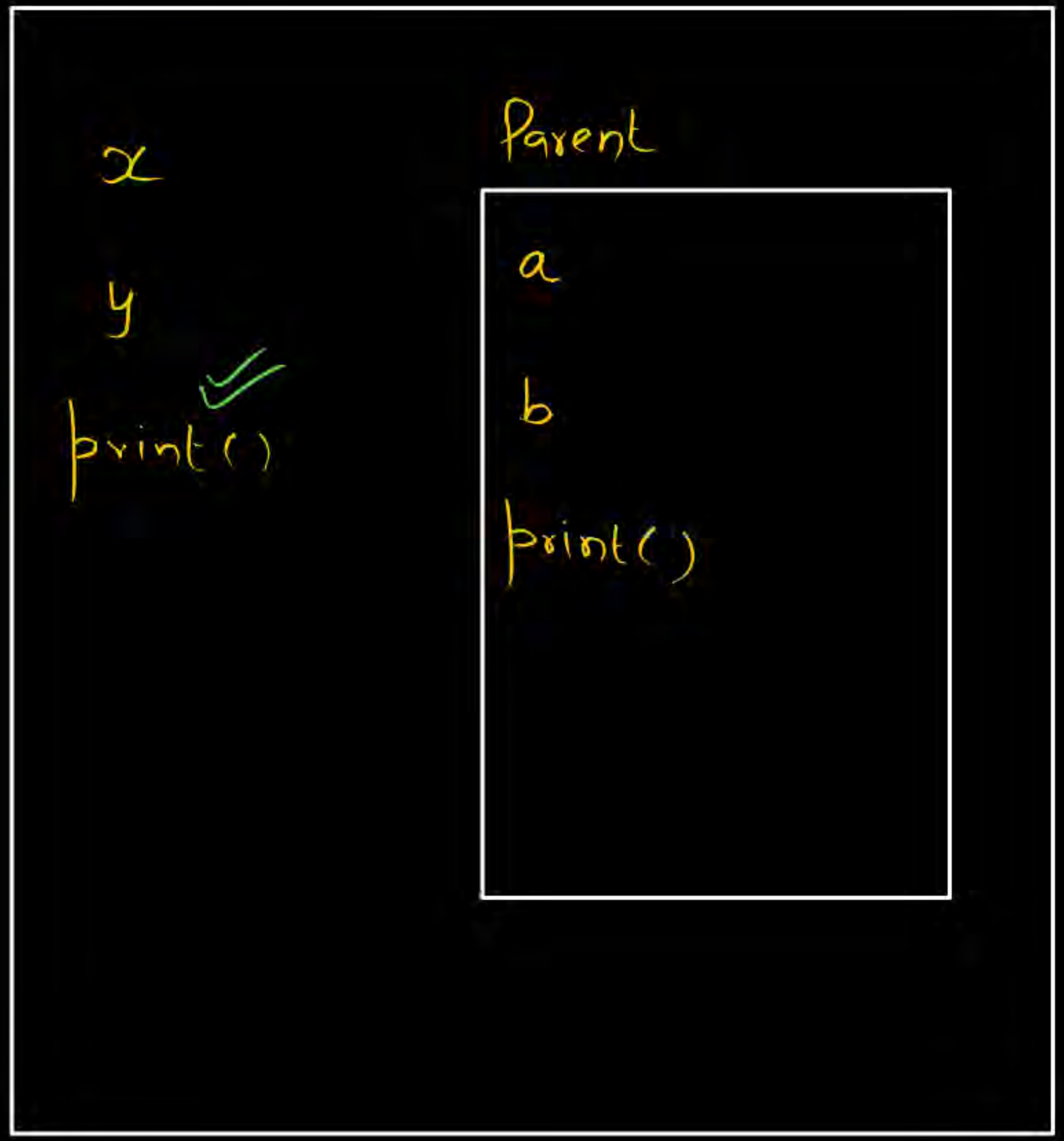
public

```
def getx()
    return self.__x
```

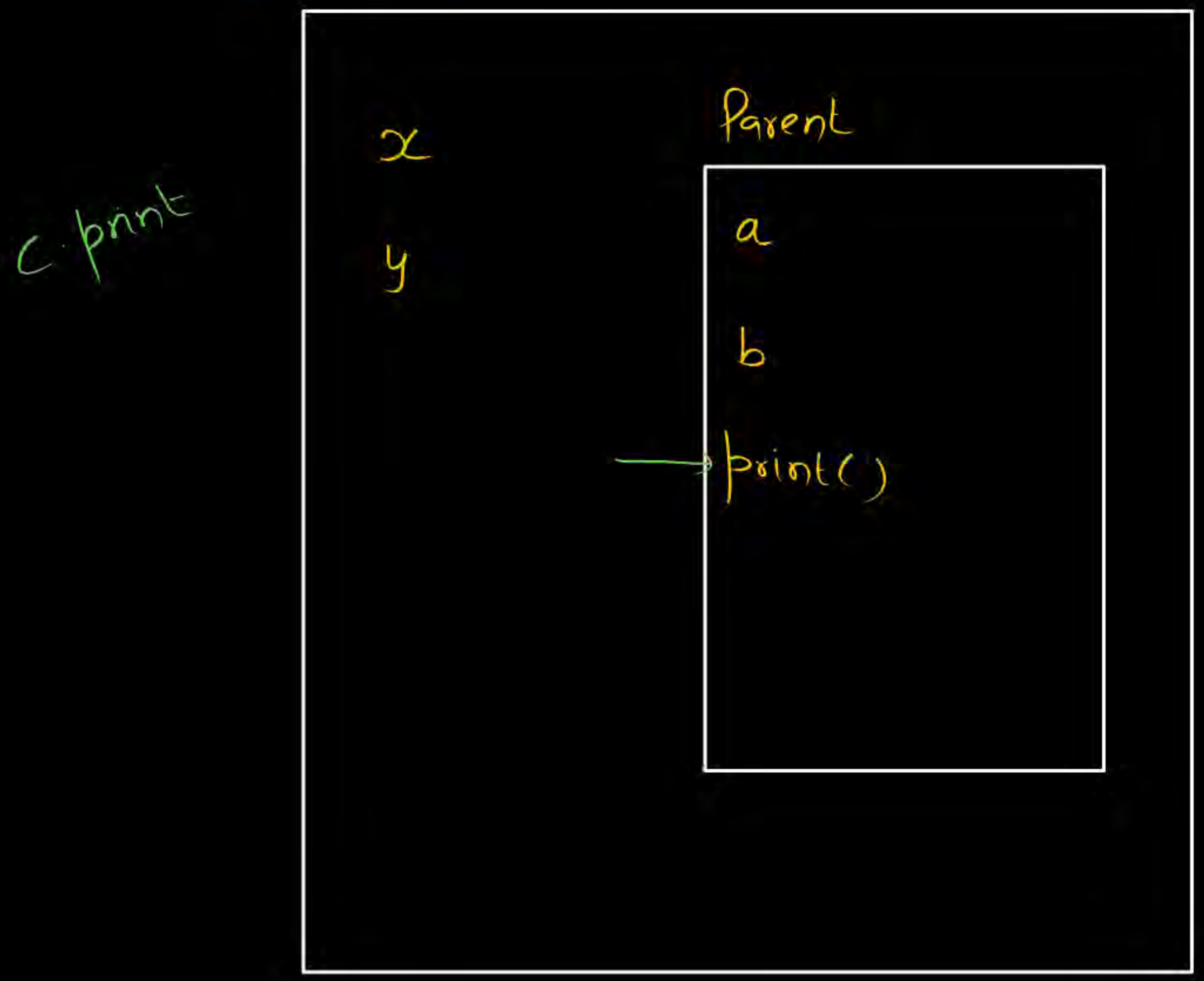

c = child( )
c. print( )
✓ child
exist
parent( )
exist


c = child( )
c. print
print( ) child
not
exist
print( ) ✓

c = child( )



c = child( )



Papa

property()

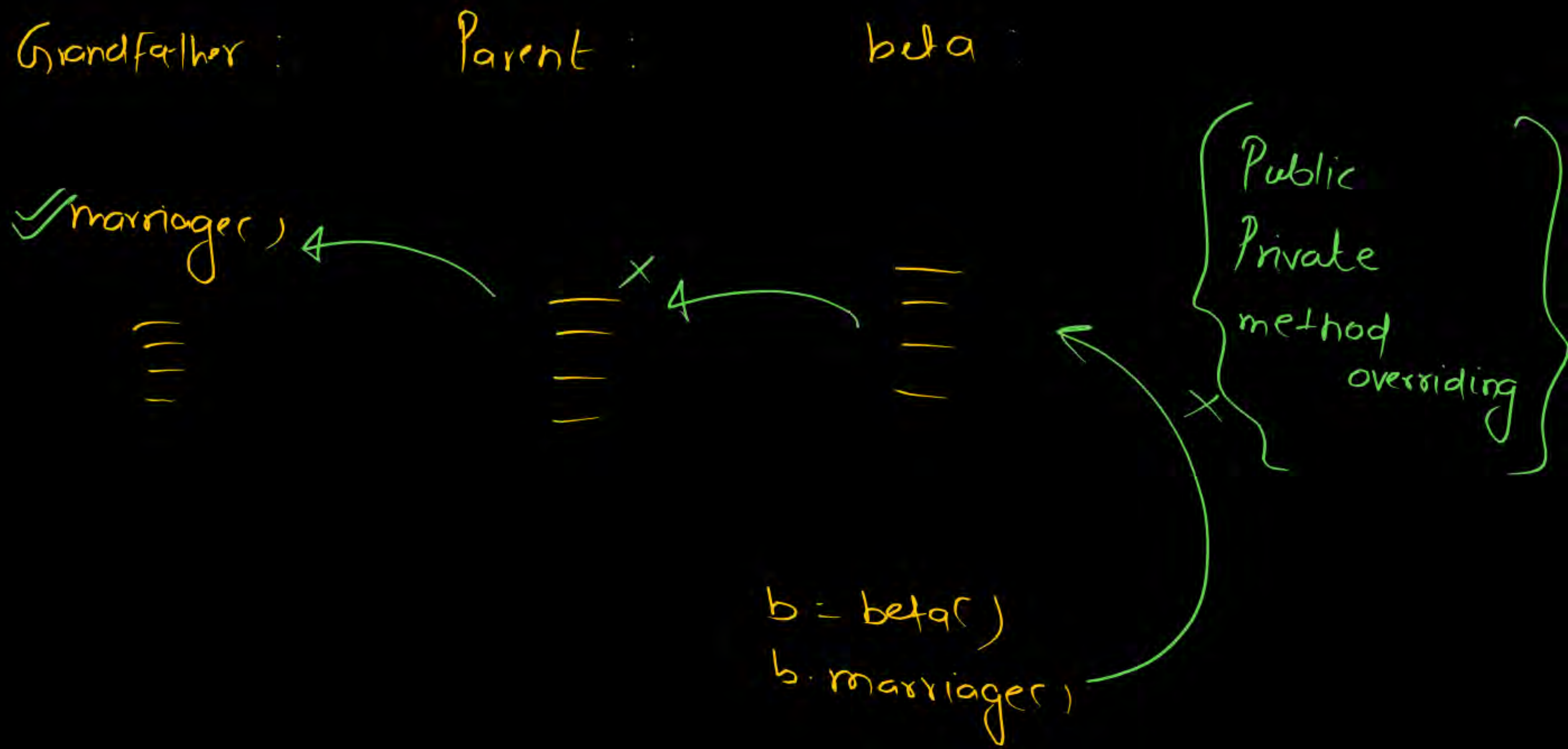
marriage()

beta

marriage()

```
b = beta()
b.marriage()
```

① method overriding


Grandfather :
Parent :
beta :
✓ marriage( )
×
Public
Private
method
overriding
×
b = beta( )
b. marriage( )

10 + 20

10.34 + 20

"Pankaj" + "sharma"

overloading

f1 = Fraction(2,3)

f2 = Fraction(3,4)

f1+f2 ×

f1.add(f2) ✓

+ { }

f1 + f2

object of Fraction class

Operator overloading

m1 + m2

```
class matrix:
    def __init__(self, row, col):
        ≡

    def add( - - - )
```

7 days

- L.L ✓
- Stack ✓
- Queue ✓
- Trees ✓

Graph ✓

Morning ✓

- OOPS

Evening

- L.L

7 days

$m_1 + m_2 \Rightarrow$

vector

$3 \times (2i + 3j + 4k)$

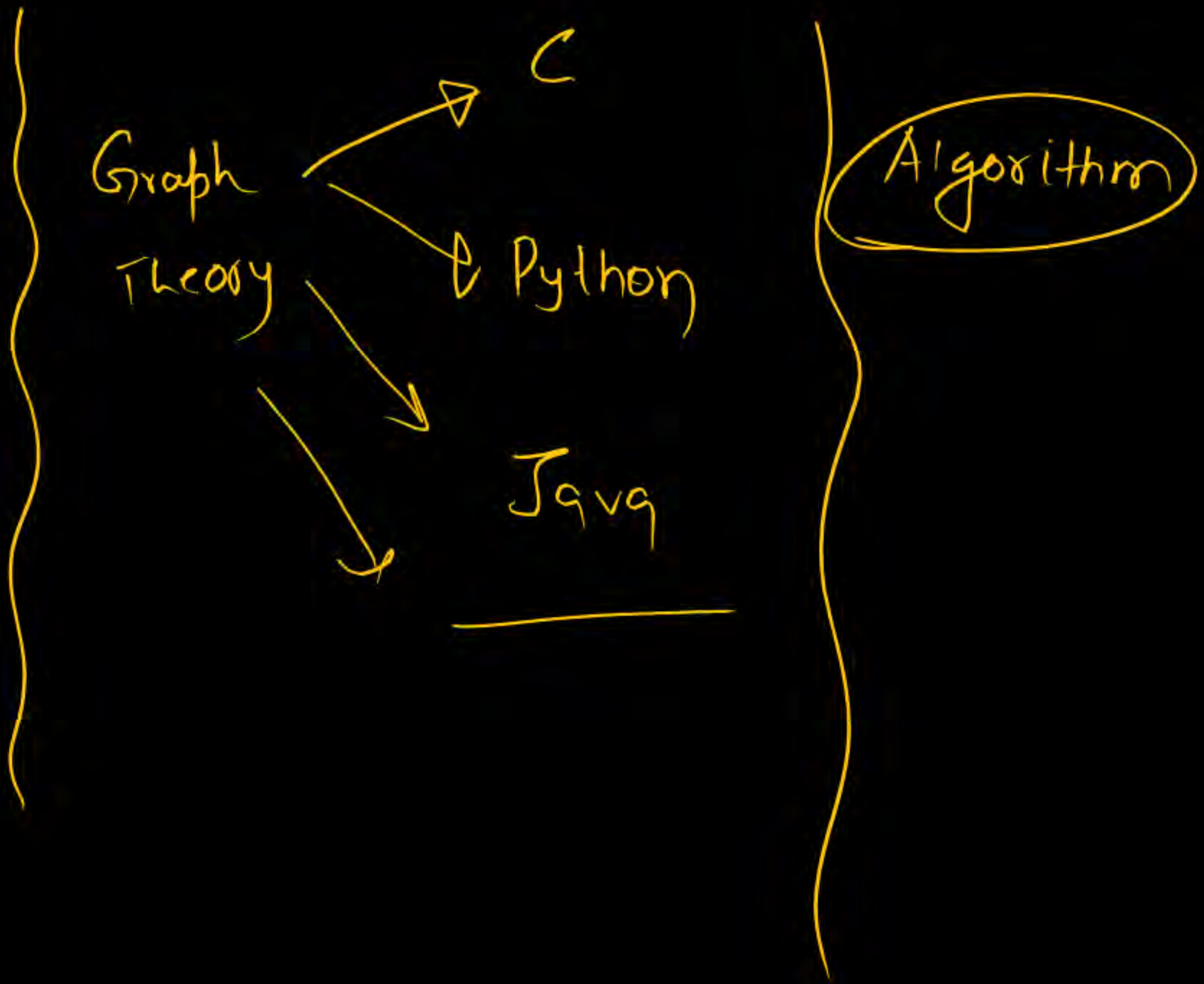
Class vector :

H·W

[CC
  ↳ recursion
  ↳ uploaded]

OOPS

Linked - list


Graph Theory
C
Python
Java
Algorithm

day 24

In [7]:
```
class Parent :
    def __init__(self,x,y):
        self.x=x
        self.y=y
class child(Parent):
    def __init__(self,x,y,a,b):
        #to initailize x,y ====>Parent ka constructir chaiye
        super().__init__(x,y)
        self.a=a
        self.b=b
    def print(self):
        print("x==",self.x)
        print("y==",self.y)
        print("a==",self.a)
        print("b==",self.b)
```

In [8]:
```
c=child(10,20,30,40)#constructor called implicitly
```

In [9]:
```
c.print()
```

```
x== 10
y== 20
a== 30
b== 40
```

In [12]:
```
class Parent :
    def __init__(self,x,y):
        self.__x=x #private
        self.y=y
class child(Parent):
    def __init__(self,x,y,a,b):
        #to initailize x,y ====>Parent ka constructir chaiye
        super().__init__(x,y)
        self.a=a
        self.b=b
    def print(self):
        print("x==",self.x)
        print("y==",self.y)
        print("a==",self.a)
        print("b==",self.b)
```

In [13]:
```
c=child(10,20,30,40)
```

In [14]:
```
c.print()
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In[14], line 1
----> 1 c.print()

Cell In[12], line 12, in child.print(self)
     11 def print(self):
---> 12     print("x==",self.x)
     13     print("y==",self.y)
     14     print("a==",self.a)

AttributeError: 'child' object has no attribute 'x'
```

In [17]:
```python
class Parent :
    def __init__(self,x,y):
        self.__x=x #private
        self.y=y
    def getx(self):
        return self.__x
    def setx(x):
        self.__x=x
    def print(self):
        print("x==",self.__x)
        print("y==",self.y)
class child(Parent):
    def __init__(self,x,y,a,b):
        #to initailize x,y ====>Parent ka constructir chaiye
        super().__init__(x,y)
        self.a=a
        self.b=b
    def print(self):
        super().print()
        print("a==",self.a)
        print("b==",self.b)
```

In [18]:
```python
c=child(10,20,30,40)
```

In [19]:
```python
c.print()
```

```
x== 10
y== 20
a== 30
b== 40
```

In [20]:
```python
class Parent :
    def __init__(self,x,y):
        self.__x=x #private
        self.y=y
    def getx(self):
        return self.__x
    def setx(x):
        self.__x=x
    def print(self):
        print("x==",self.__x)
        print("y==",self.y)
class child(Parent):
    def __init__(self,x,y,a,b):
        #to initailize x,y ====>Parent ka constructir chaiye
        super().__init__(x,y)
        self.a=a
        self.b=b
    def print(self):
        #self.print()
        print("a==",self.a)
        print("b==",self.b)
```

In [21]:
```
c=child(10,20,30,40)
```

In [22]:
```
c.print()  #child wala
```

```
a== 30
b== 40
```

In [23]:
```
class Parent :
    def __init__(self,x,y):
        self.__x=x #private
        self.y=y
    def getx(self):
        return self.__x
    def setx(x):
        self.__x=x
    def print(self):
        print("x==",self.__x)
        print("y==",self.y)
class child(Parent):
    def __init__(self,x,y,a,b):
        #to initailize x,y ====>Parent ka constructir chaiye
        super().__init__(x,y)
        self.a=a
        self.b=b
```

In [24]:
```
c=child(10,20,30,40)
```

In [25]:
```
c.print()
```

```
x== 10
y== 20
```

In [26]:
```
class Parent :
    def __init__(self,x,y):
        self.__x=x #private
        self.y=y
    def getx(self):
        return self.__x
    def setx(x):
        self.__x=x

class child(Parent):
    def __init__(self,x,y,a,b):
        #to initailize x,y ====>Parent ka constructir chaiye
        super().__init__(x,y)
        self.a=a
        self.b=b
```

```
In [27]: c=child(10,2,30,40)
```

```
In [28]: c.print()
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In[28], line 1
----> 1 c.print()

AttributeError: 'child' object has no attribute 'print'
```

```
In [ ]:
```

THANK - YOU