OOPs- PYTHON

Dr Sivabalan,

Technical Training Advisor

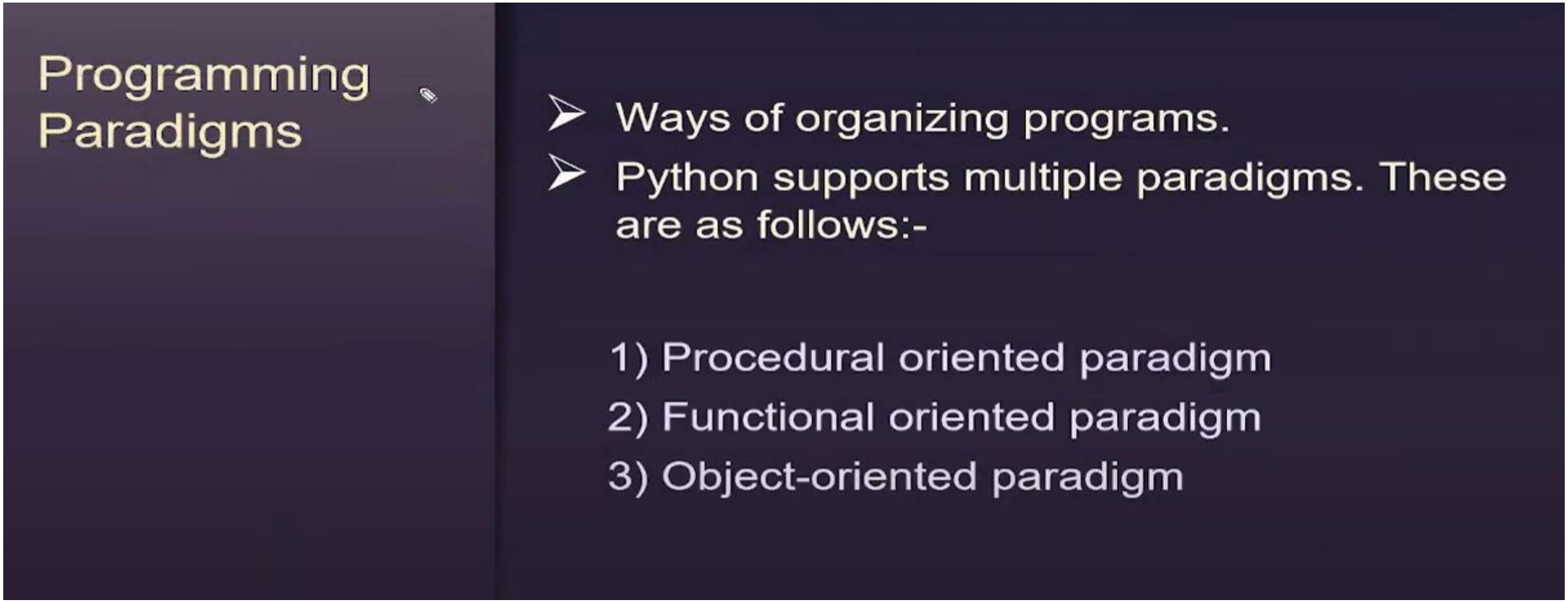
Sivabalan.n@nttdata.com

Agenda

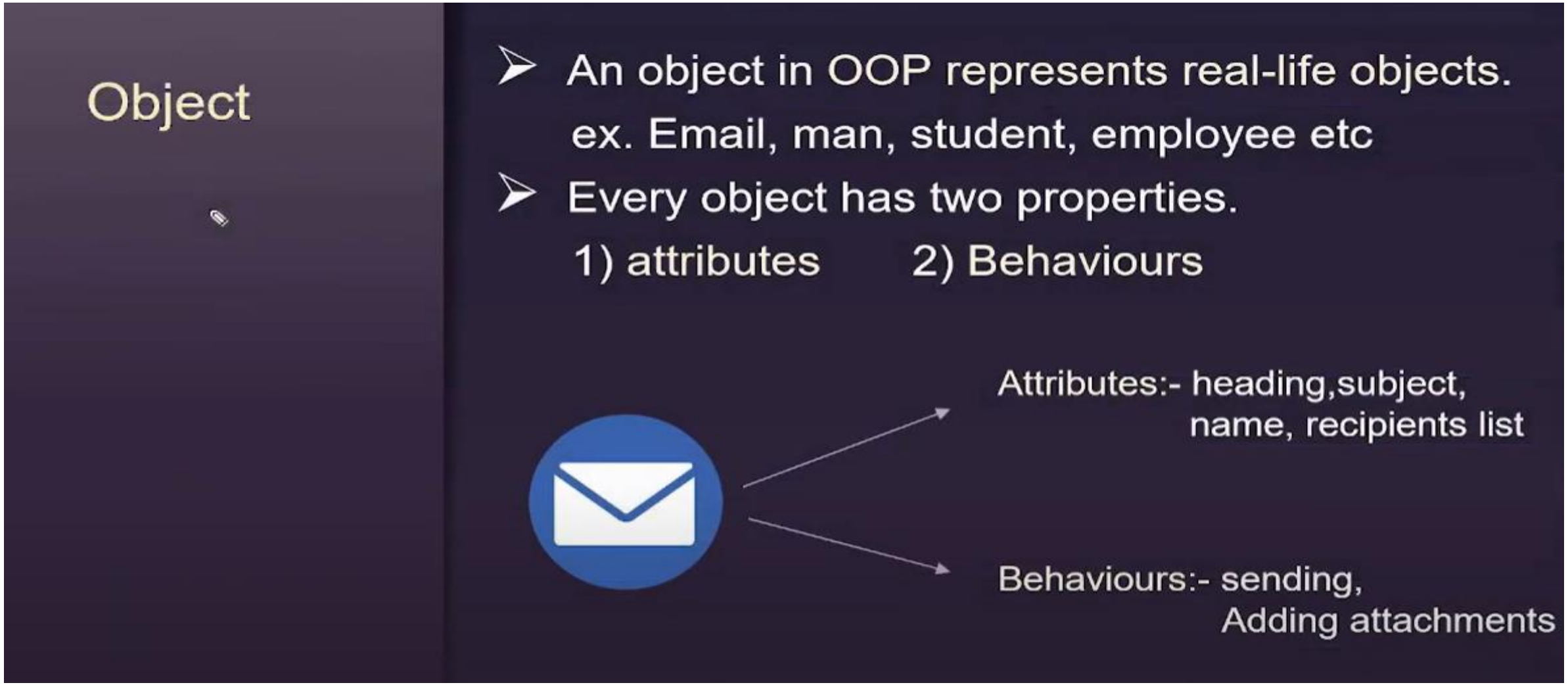
1. Basics of OOP
2. Types of variables & methods
3. Inheritance
4. Polymorphism
5. Encapsulation
6. Abstraction
7. Interface

Requirements

Requirements



Requirements

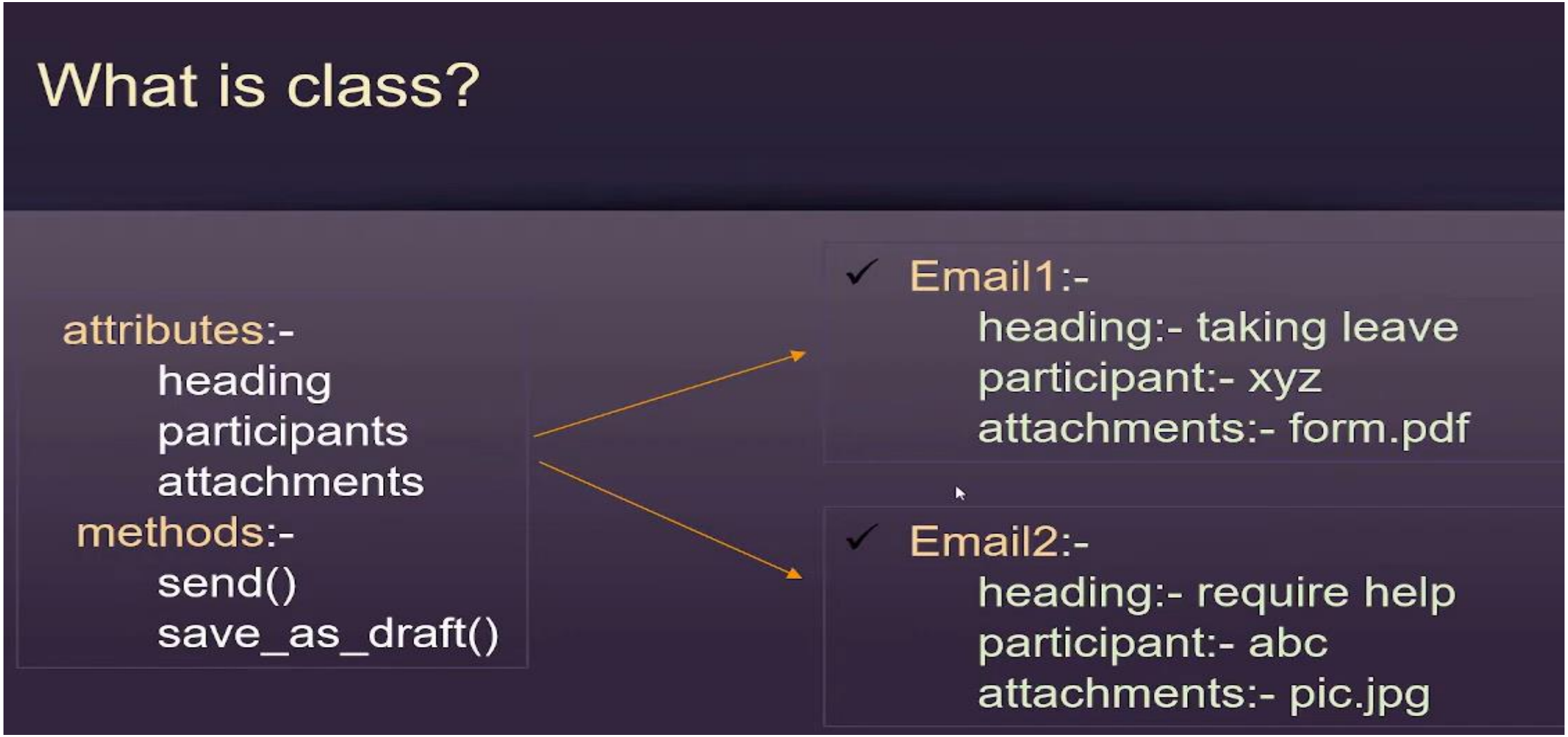


OOPs

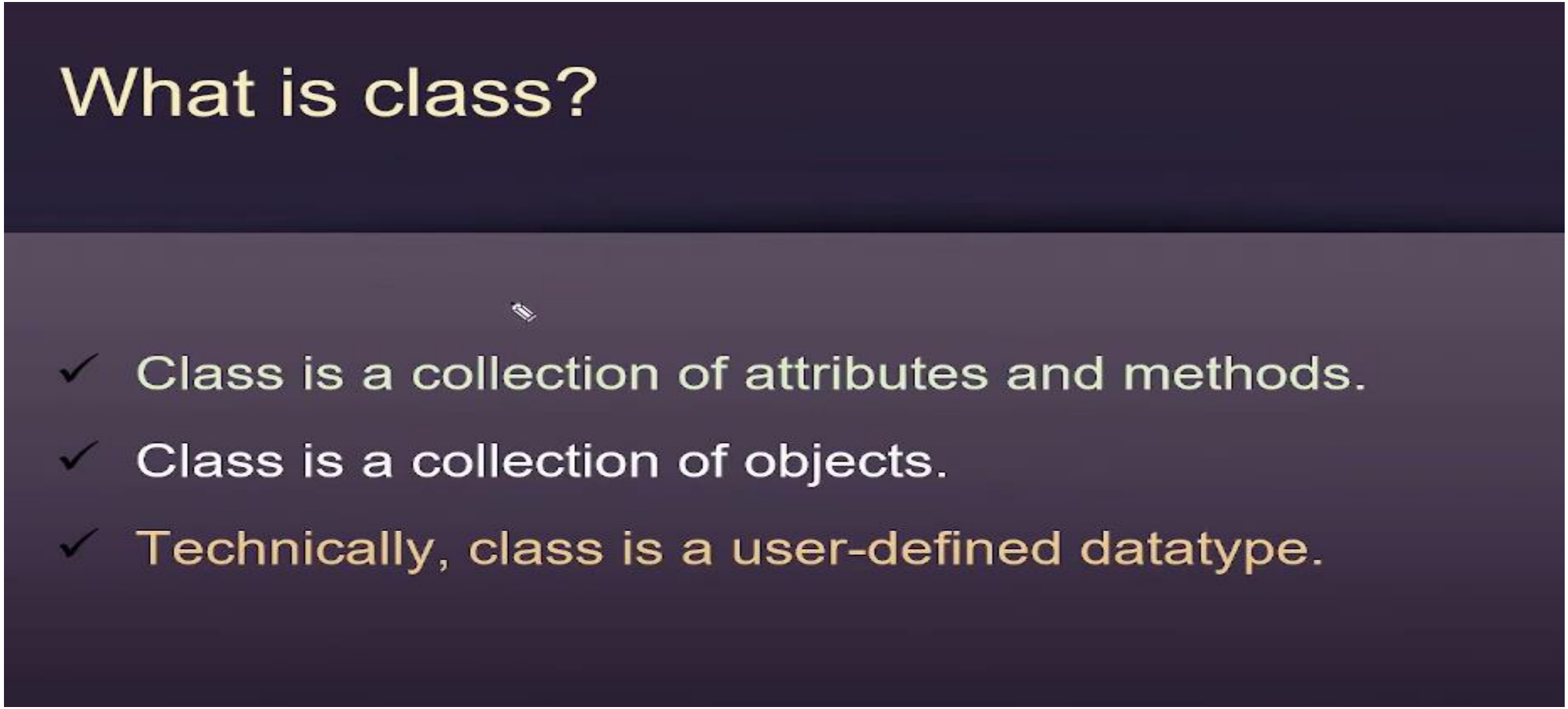
What is class?

- ✓ Class is a template/blueprint/prototype for creating objects.
- ✓ Every object belong to some class
- ✓ Email class:- email1 + email2 + email3 +email4

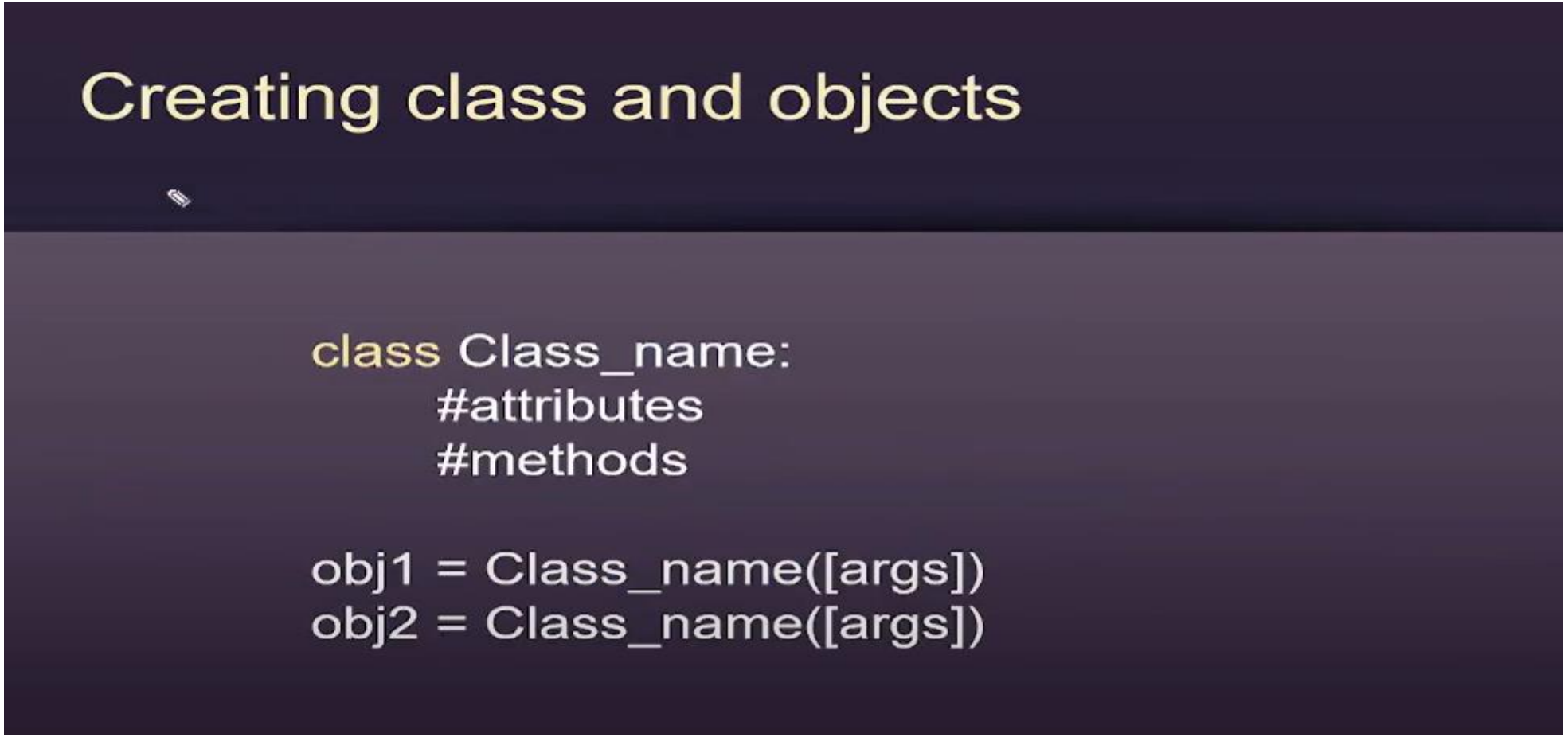
OOPs



OOPs



OOPs

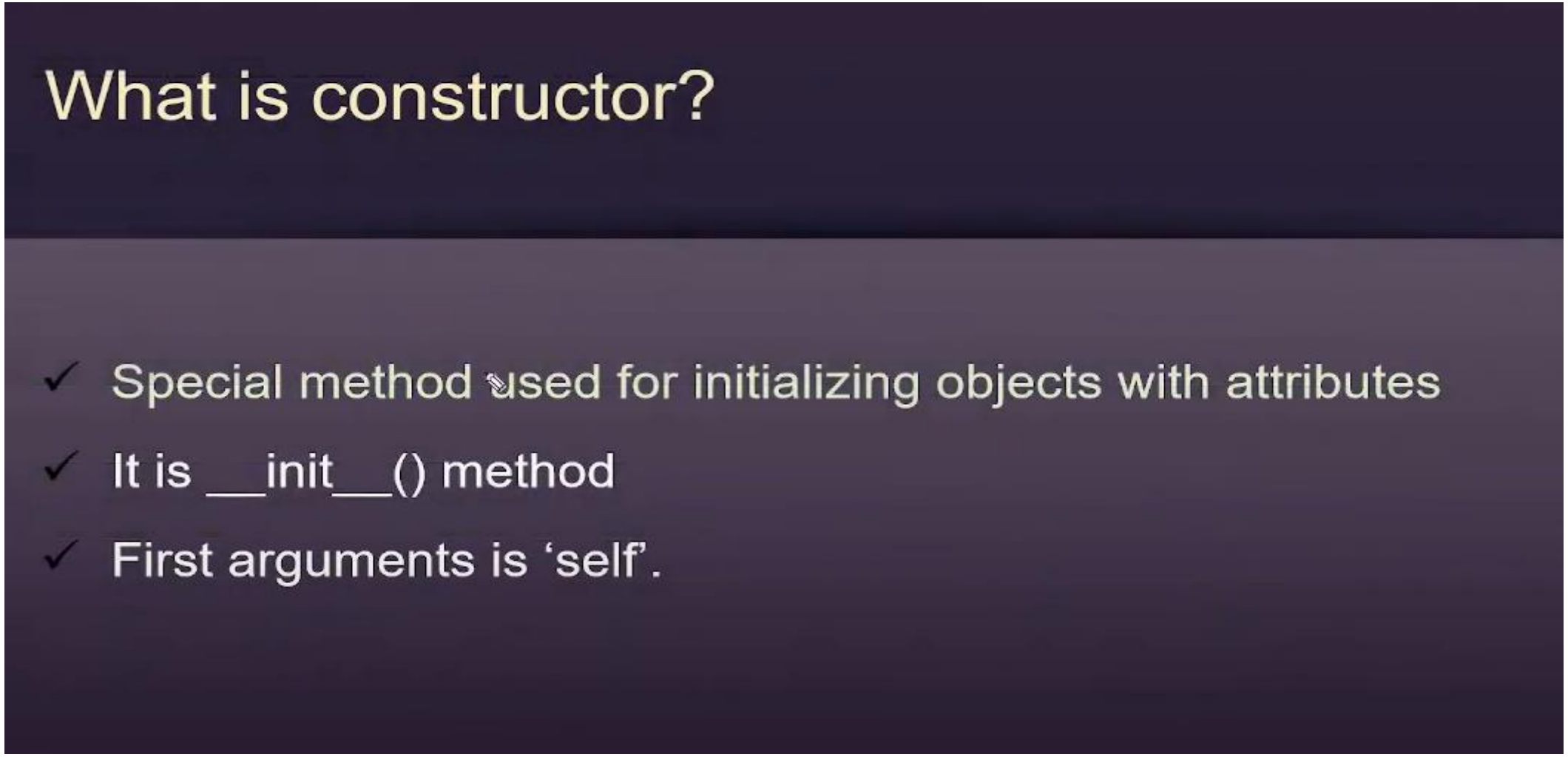


OOPs

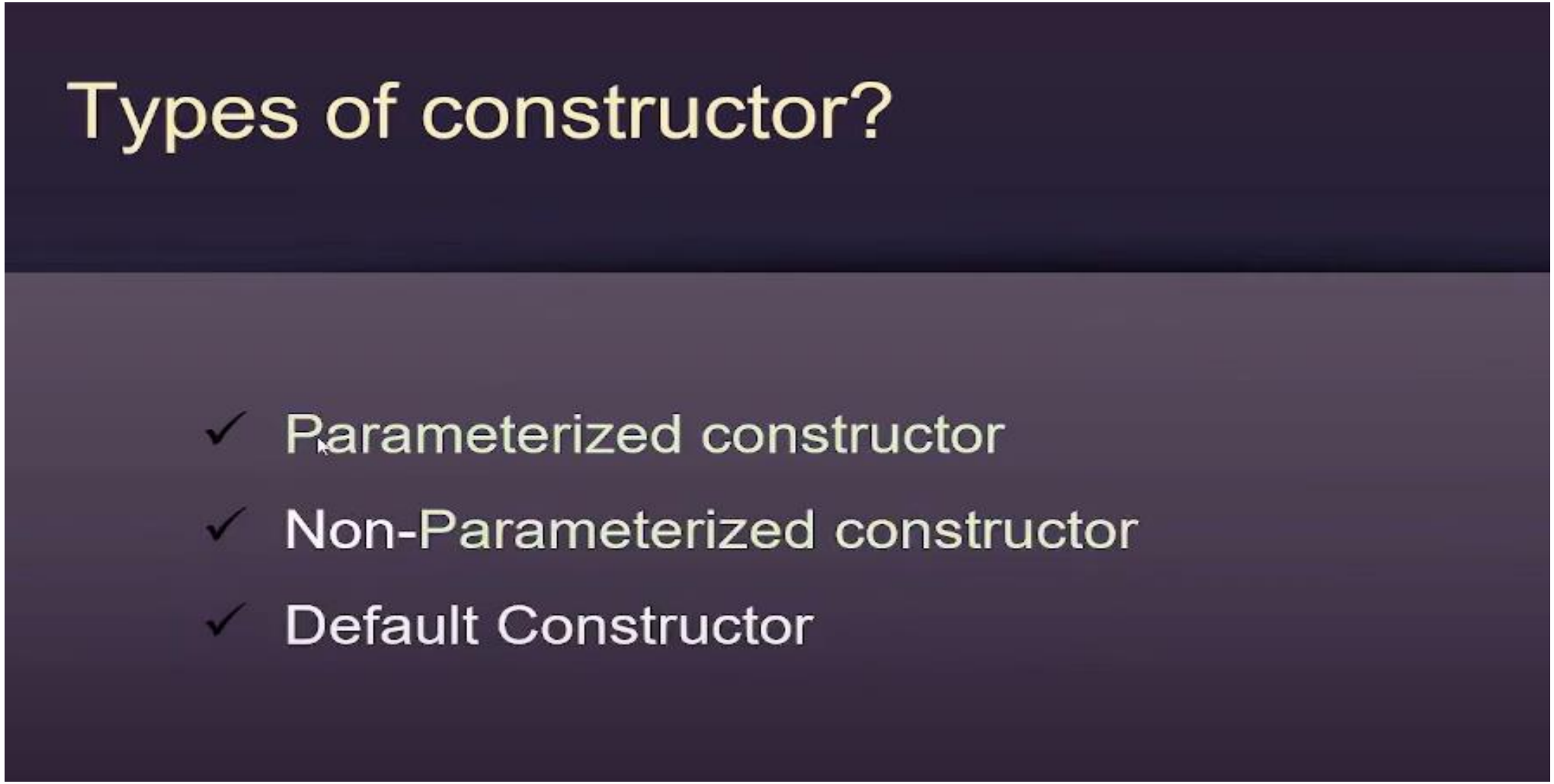
example.py

```
1  class Employee:
2      def __init__(self,nm,ag):
3          self.name=nm
4          self.age=ag
5      def display(self):
6          print(self.name)
7
8  e1=Employee('raj',21)
9  e2=Employee('jay',22)
```

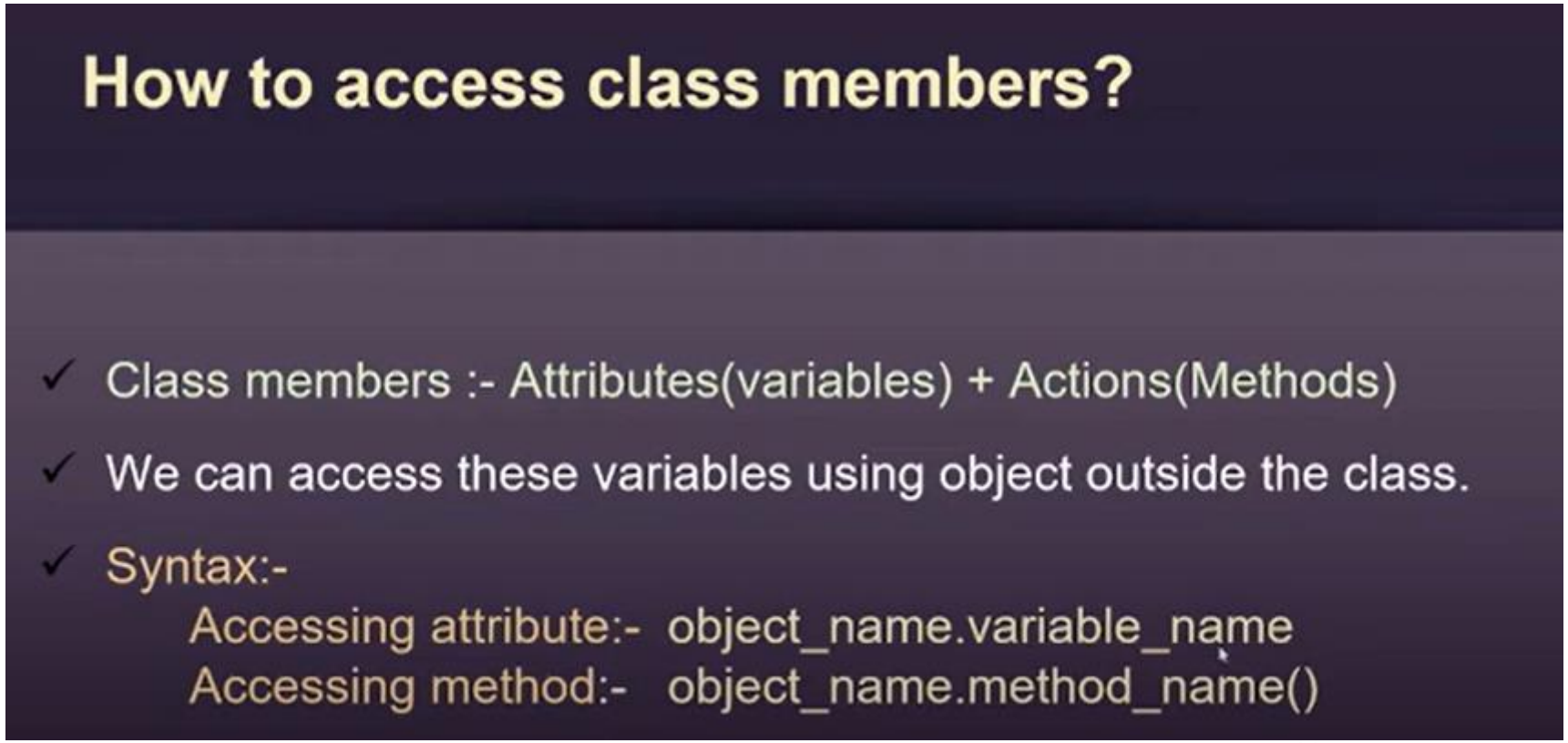
OOPs



OOPs

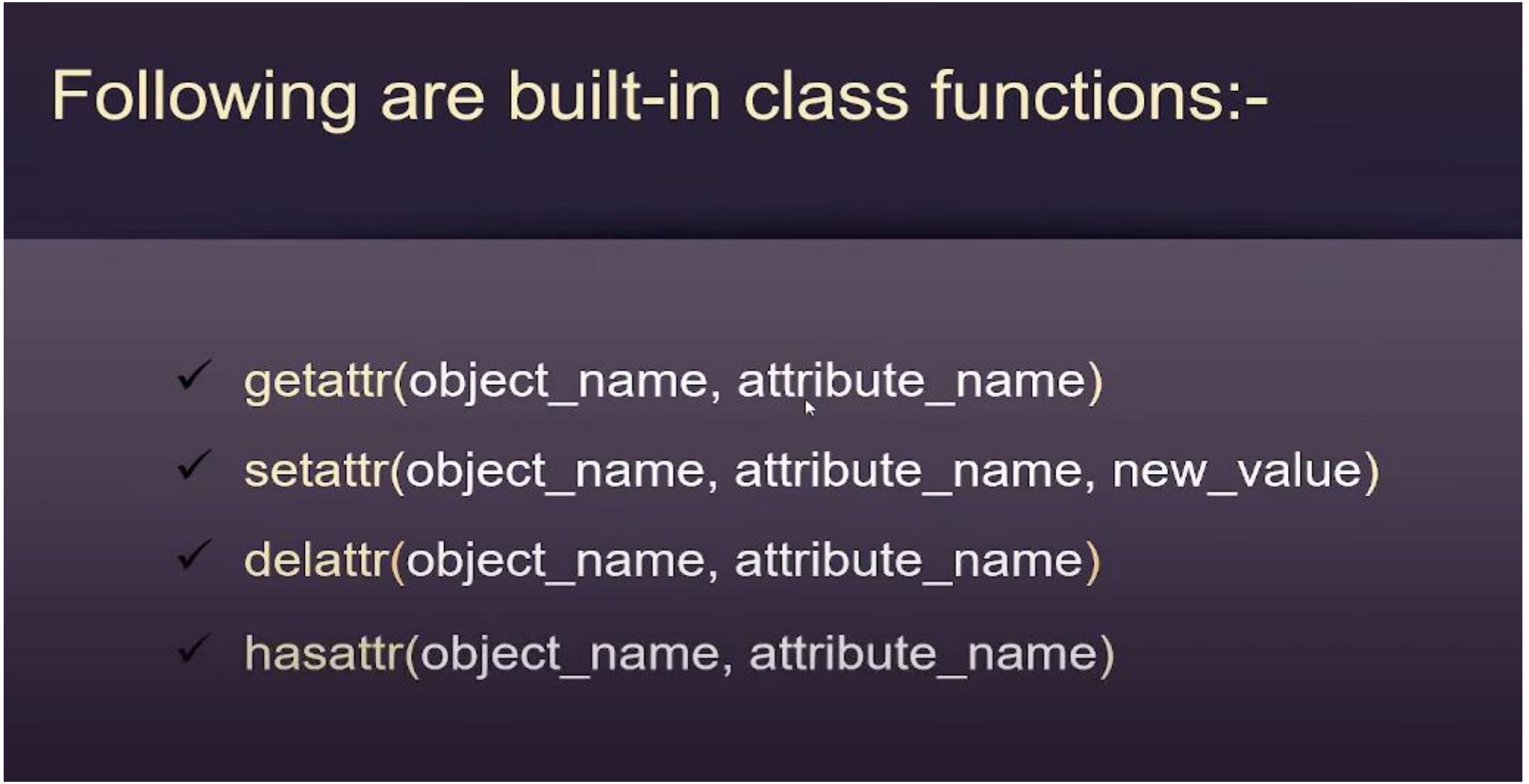


Accessing Class Members



OOPs

OOPs

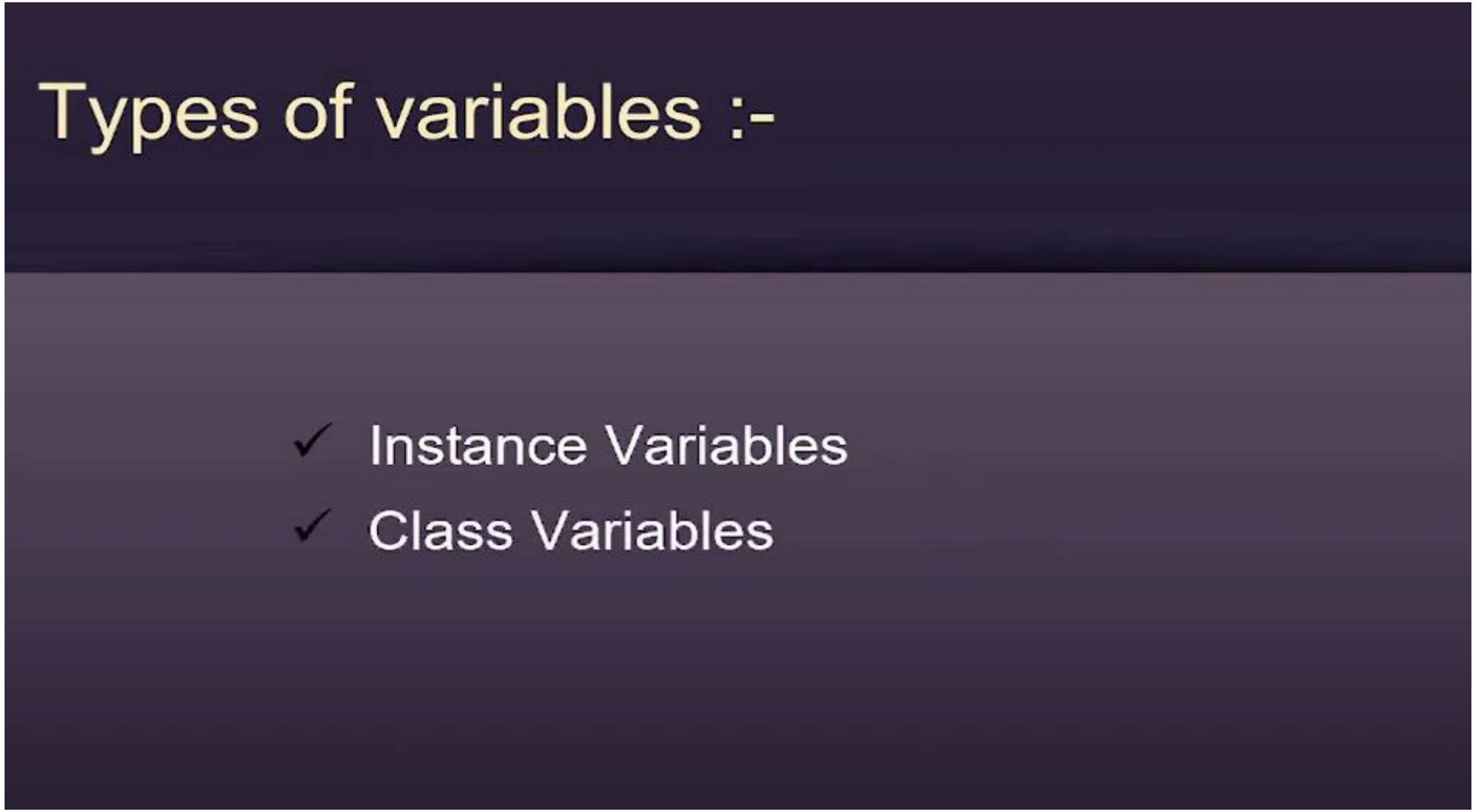


OOPs

Following are built-in class attributes:-

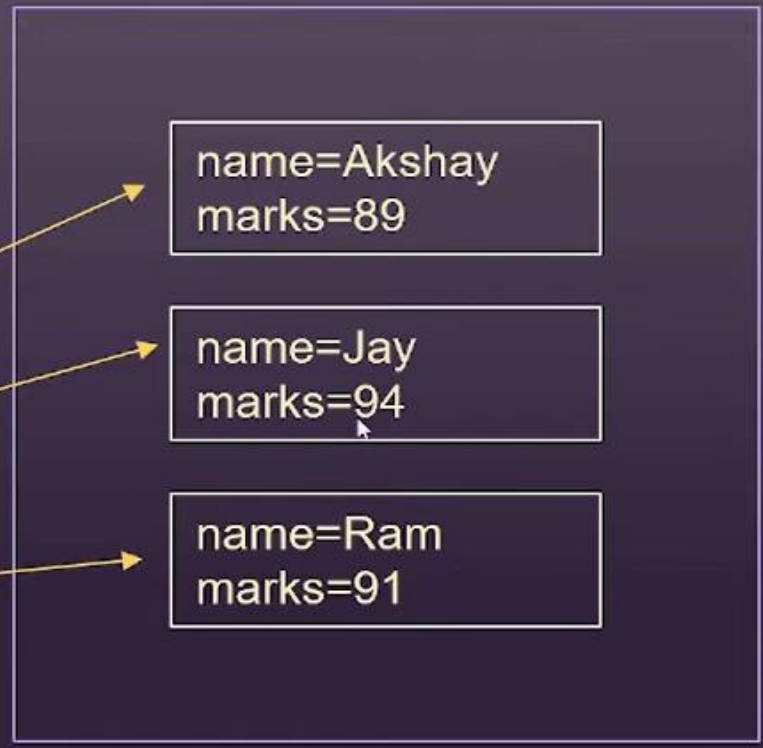
- ✓ __dict__ :- Dictionary containing class's namespace
- ✓ __doc__ :- Class documentation string.
- ✓ __name__ :- Class Name
- ✓ __module__ :- Module name in which class is defined
- ✓ __bases__ :- List of base classes

OOPs

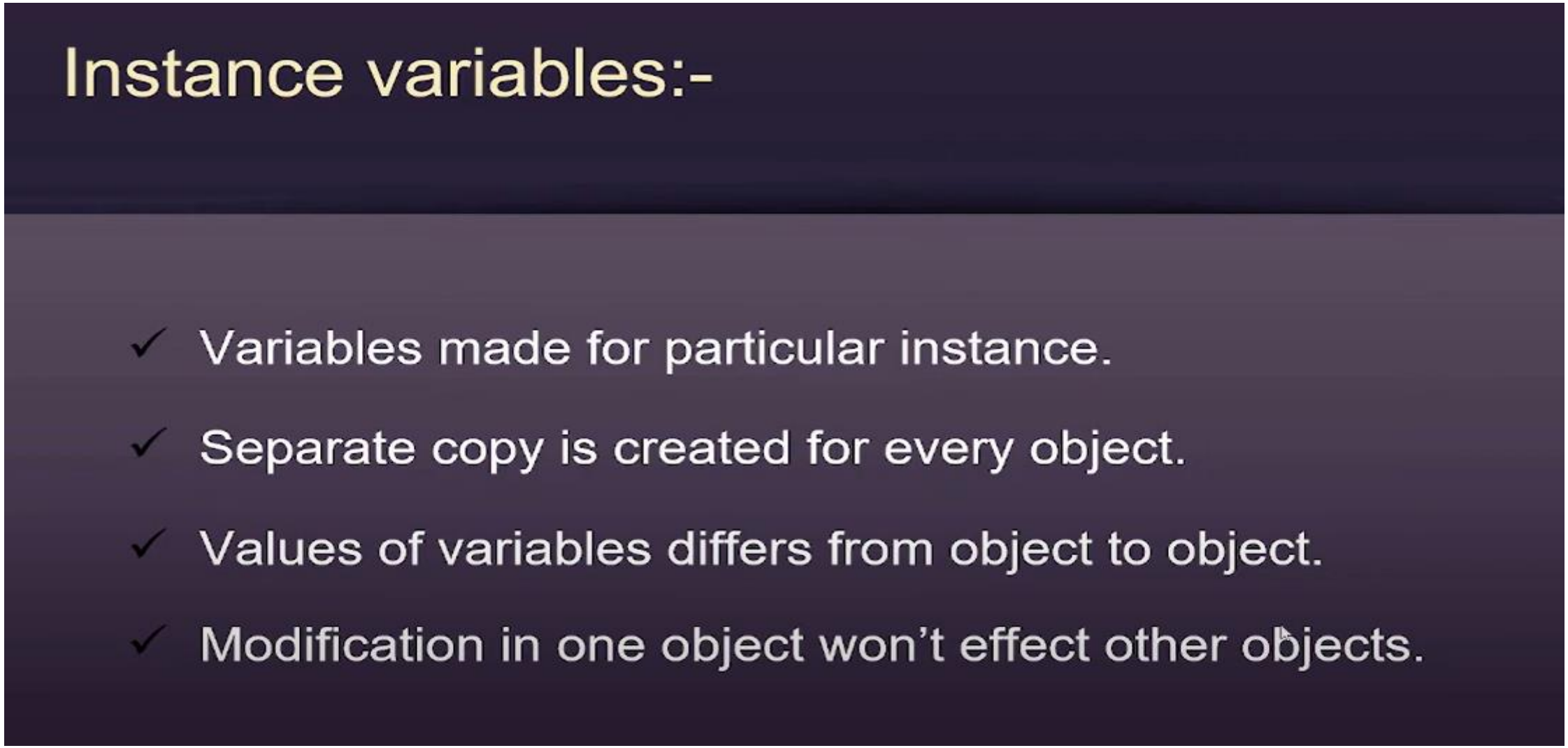


Instance variables

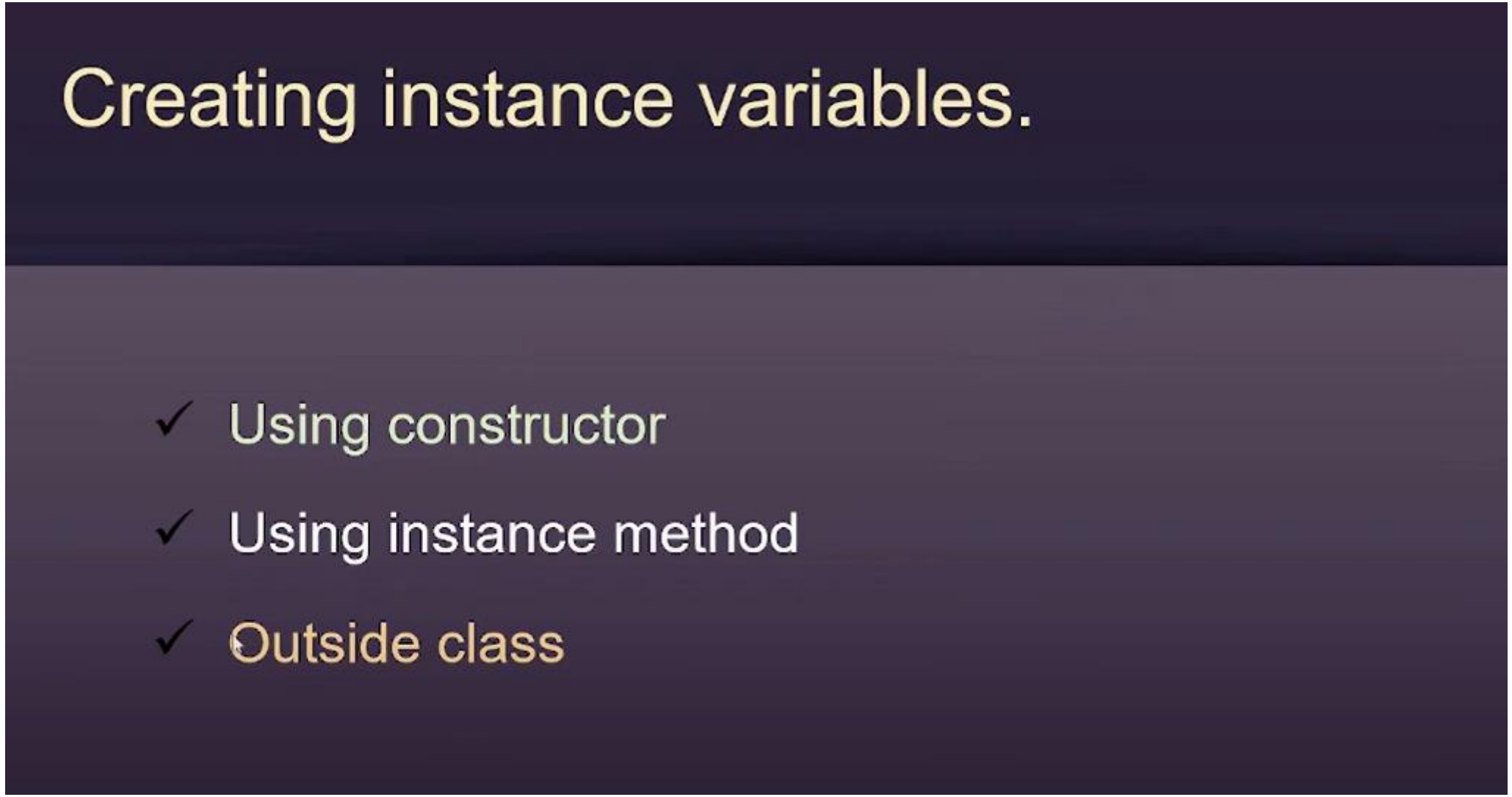
```
class Student:
        def __init__( self,nm,m ) :
                self.name=nm
                self.marks=m

std1=Student('Akshay',89)
std2=Student('Jay',94)
std3=Student('Ram',91)
```



OOPs

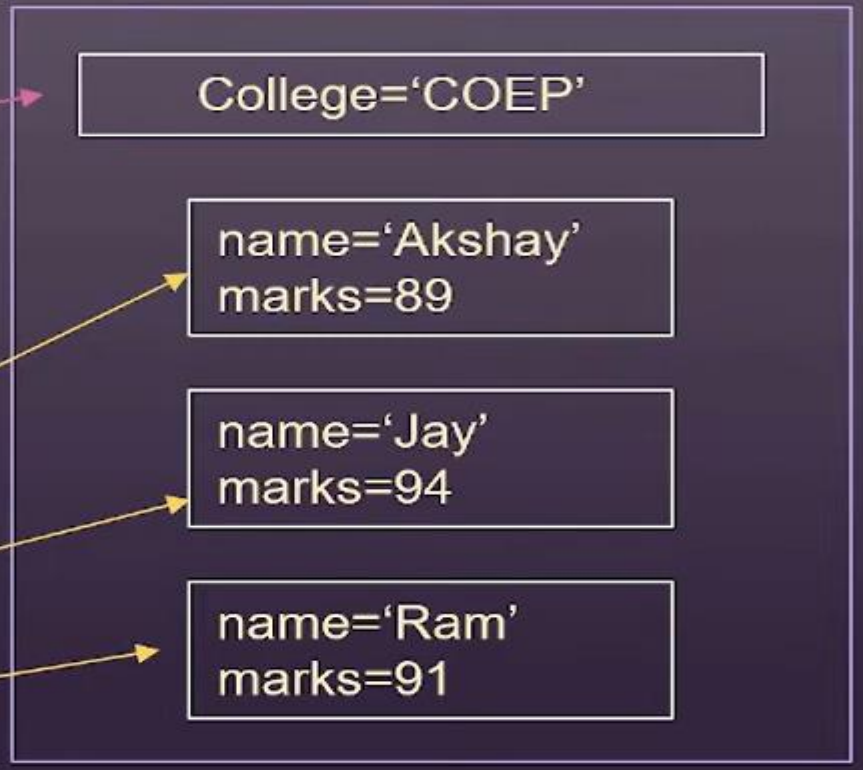


OOPs



Class variables

```
class Student:
        college='COEP'
        def __init__( self,nm,m ) :
                self.name=nm
                self.marks=m

std1=Student('Akshay',89)
std2=Student('Jay',94)
std3=Student('Ram',91)
```



Class variables:-

- ✓ Variables made for entire class (All objects)
- ✓ Only one copy is created and distributed to all objects
- ✓ Modification in class variable impact on all objects.

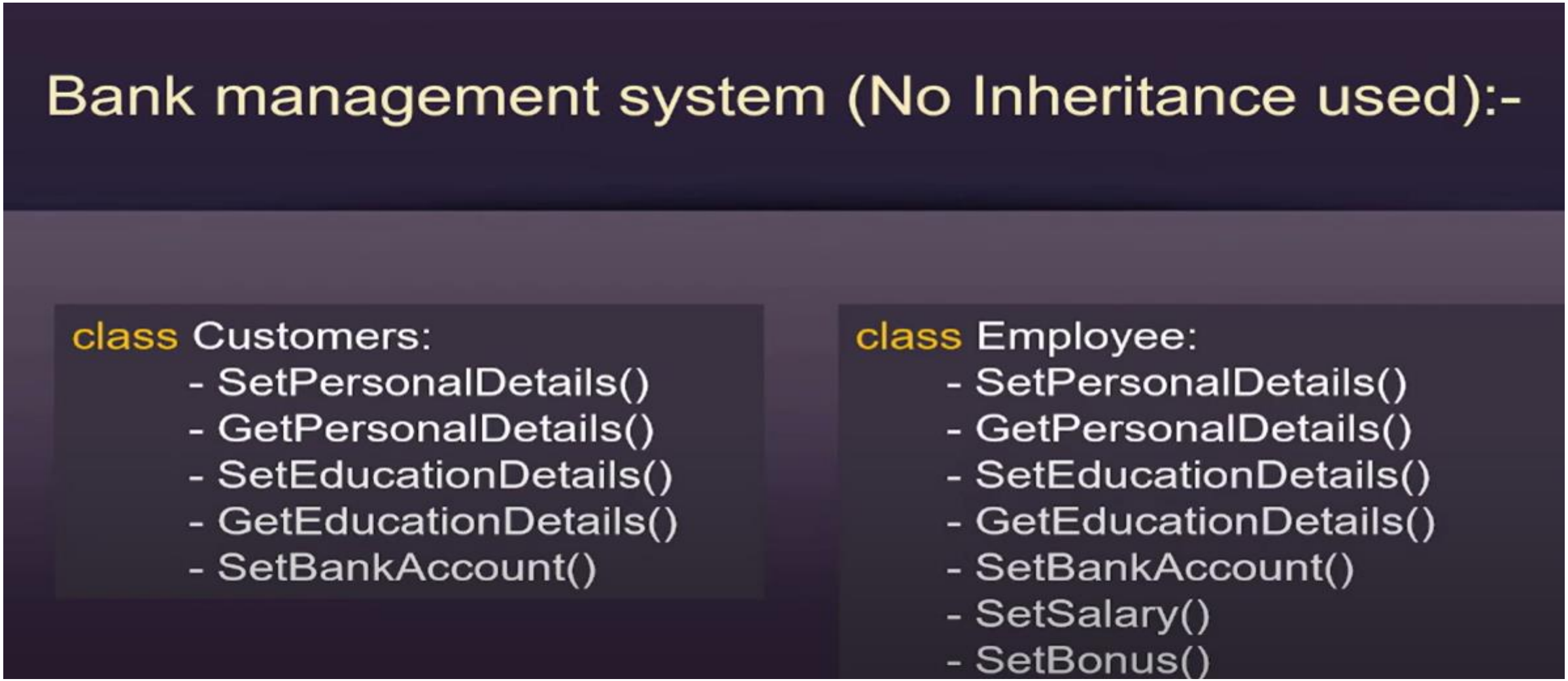
OOPs

What is inheritance ?

- ✓ Deriving a new class from an existing class so that new class inherits all members (attributes + methods) of existing class is called as inheritance.
- ✓ Old class :- Parent class, Base class, Existing class, Super class
- ✓ New class :- Child class, sub class, derived class

OOPs

OOPs



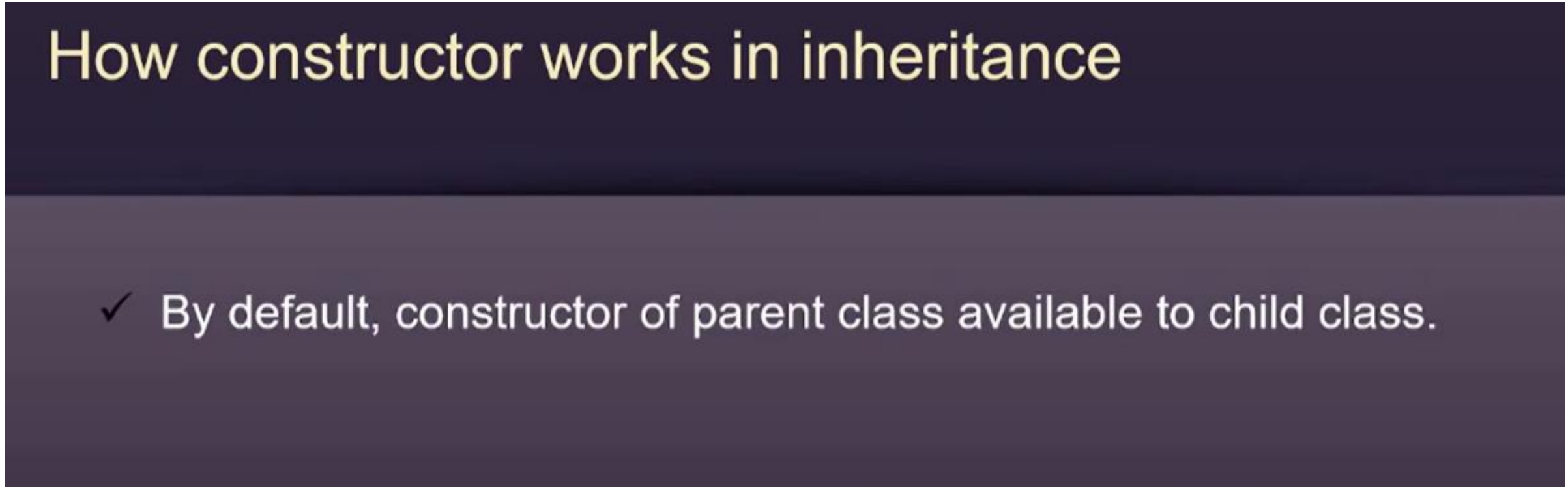
OOPs

Bank management system (Inheritance used):-

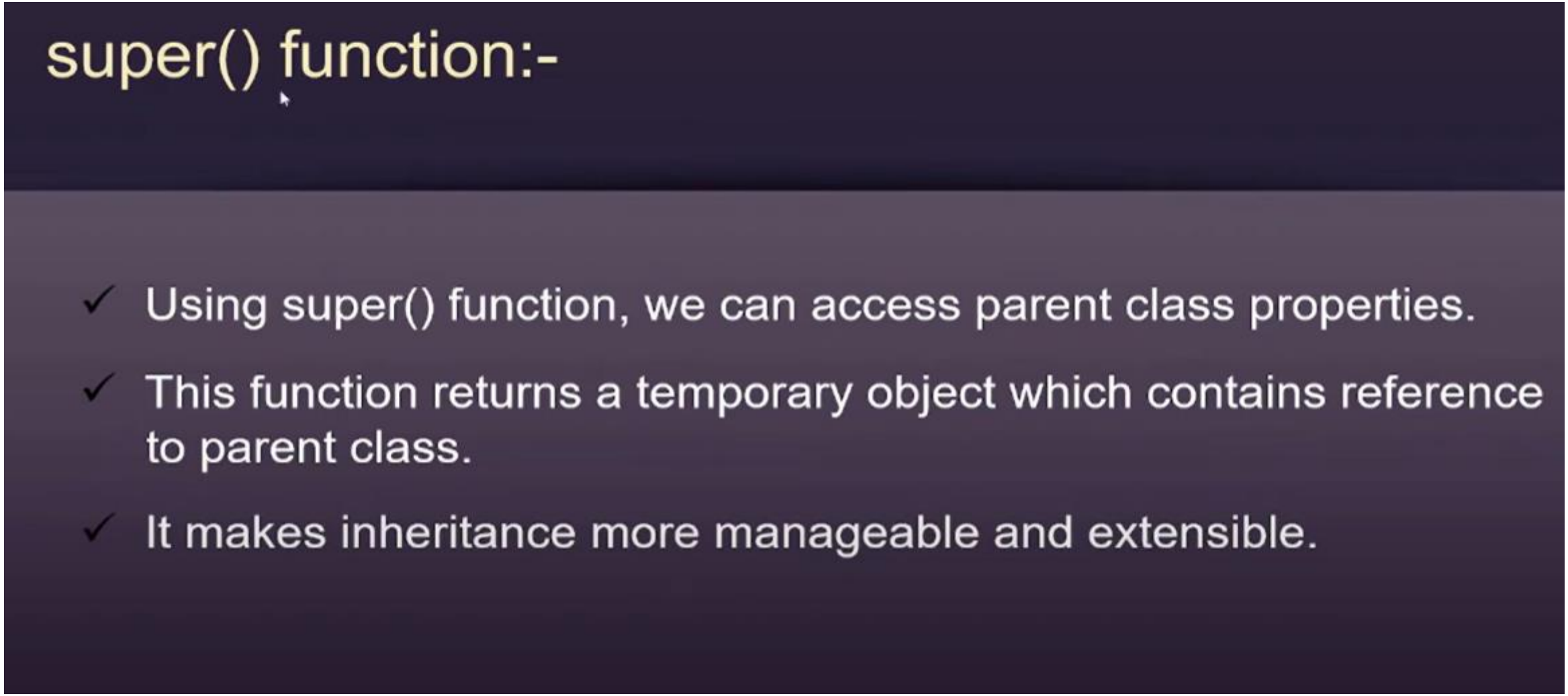
```
class Customers:
        - SetPersonalDetails()
        - GetPersonalDetails()
        - SetEducationDetails()
        - GetEducationDetails()
        - SetBankAccount()
```

```
class Employee(Customers):
        - SetBankAccount()
        - SetSalary()
        - SetBonus()
```

OOPs



OOPs

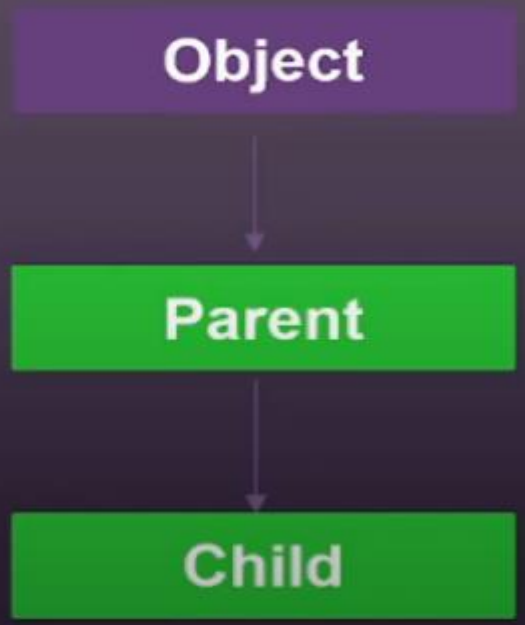


OOPs

OOPs

Single Inheritance:-

- ✓ One Parent and one child class



OOPs

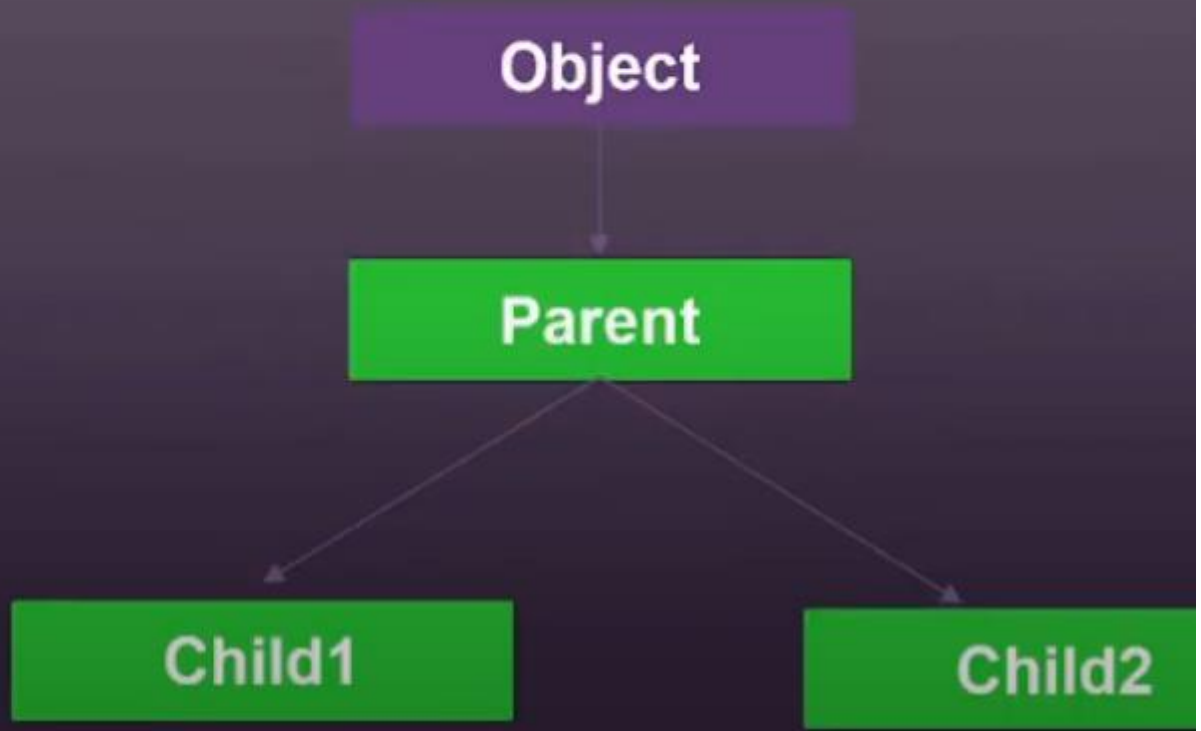
Multi-level Inheritance:-

- ✓ Parent class and child class further inherited into new class forming multiple levels.

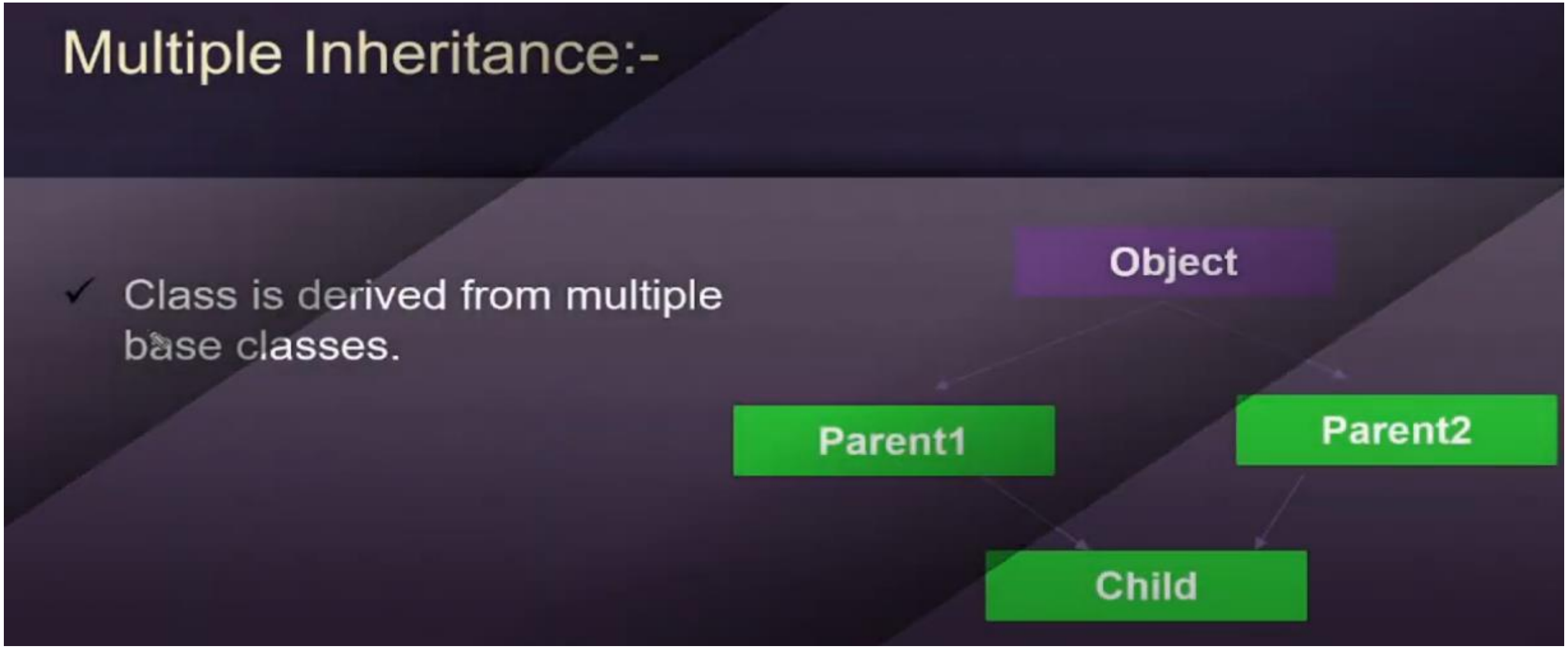
OOPs

Hierarchical Inheritance:-

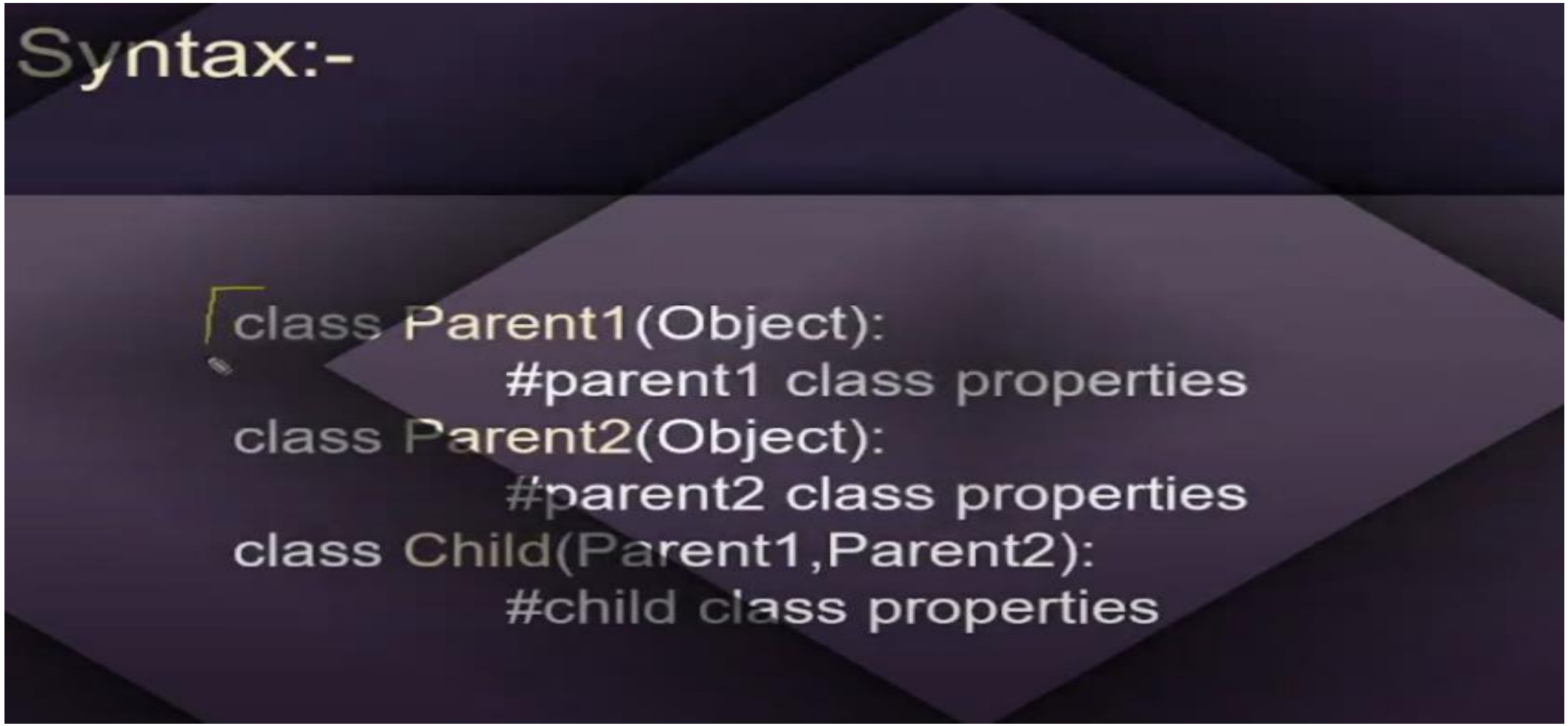
- ✓ One Parent and multiple child classes



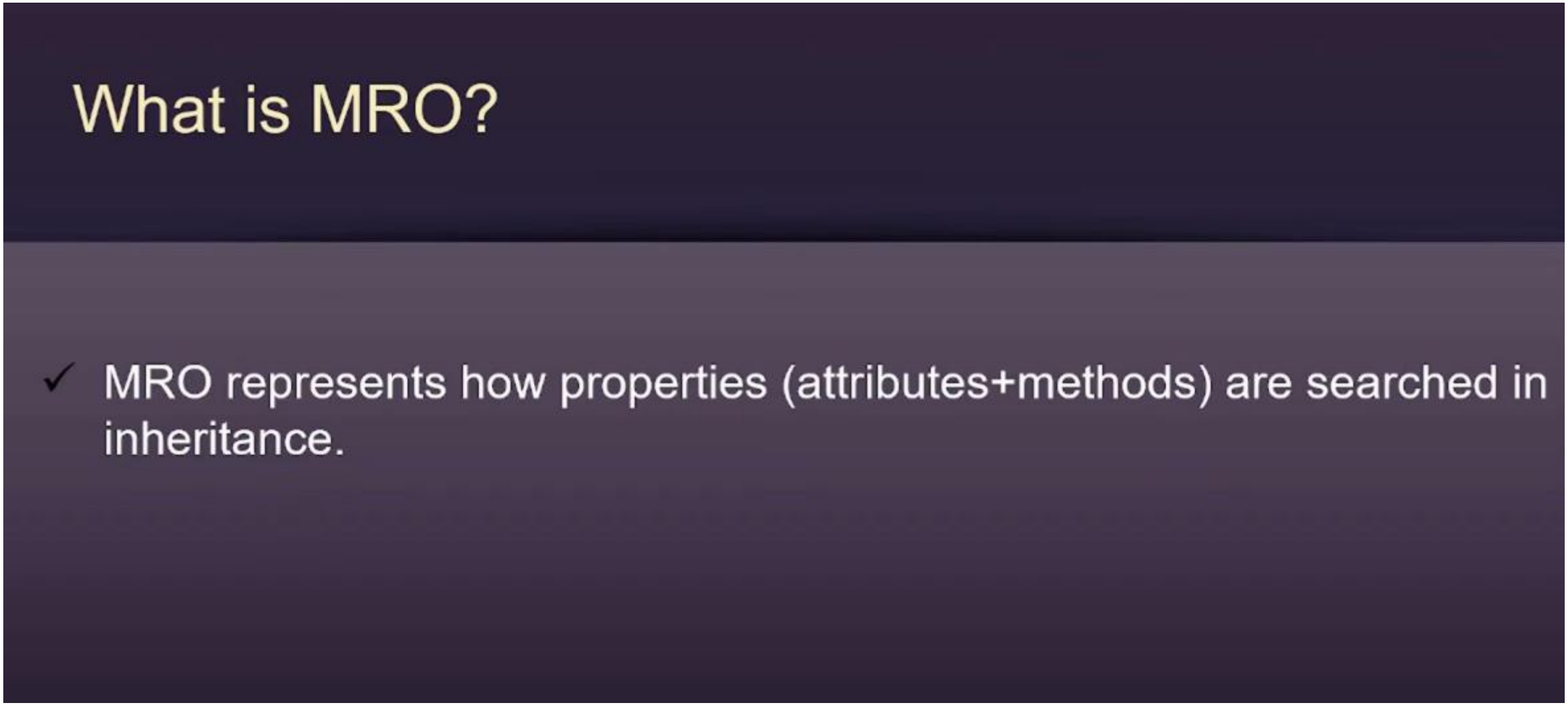
OOPs



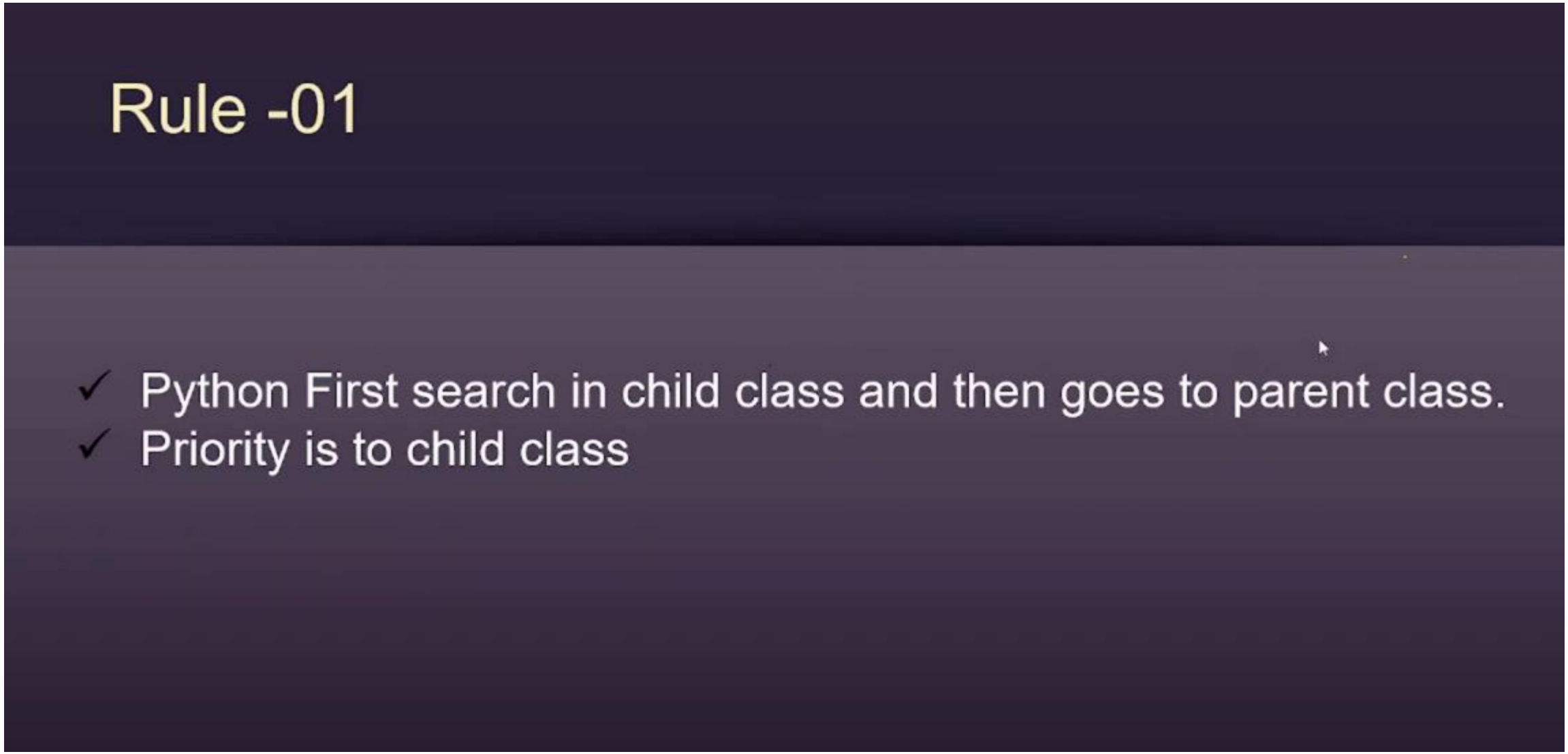
OOPs



OOPs

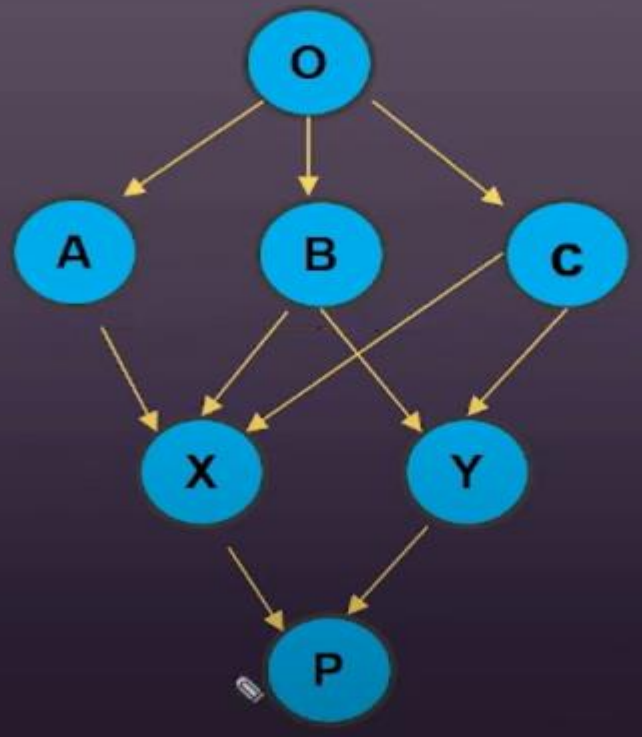


OOPs



OOPs

Rule -02 MRO Follows 'Depth First Left to Right approach'

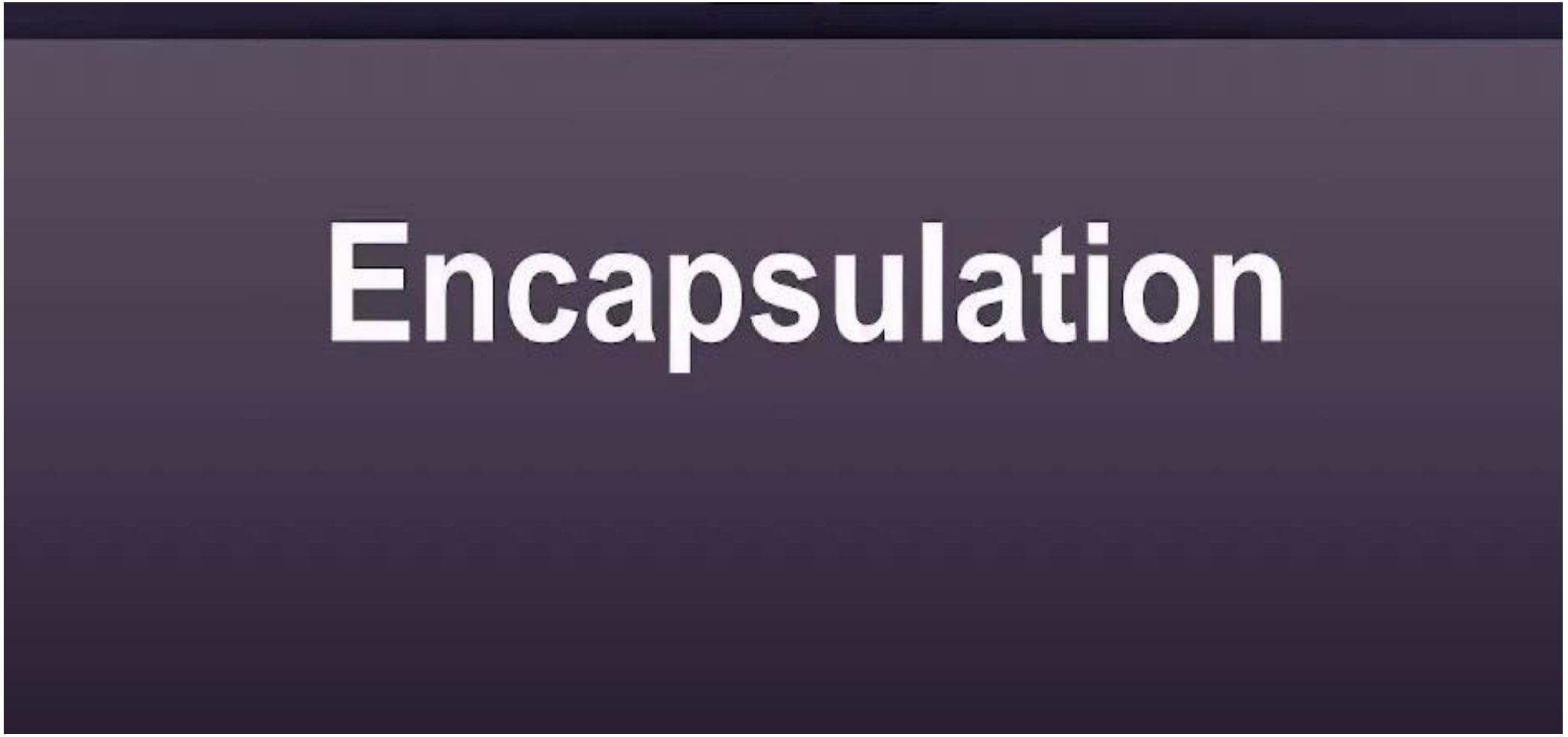


- ✓ mro(o):- Object
- ✓ mro(A):- A,O
- ✓ mro(B):- B,O
- ✓ mro(C):- C,O
- ✓ mro(X):- X,A,B,C,O
- ✓ mro(Y):- Y,B,C,O

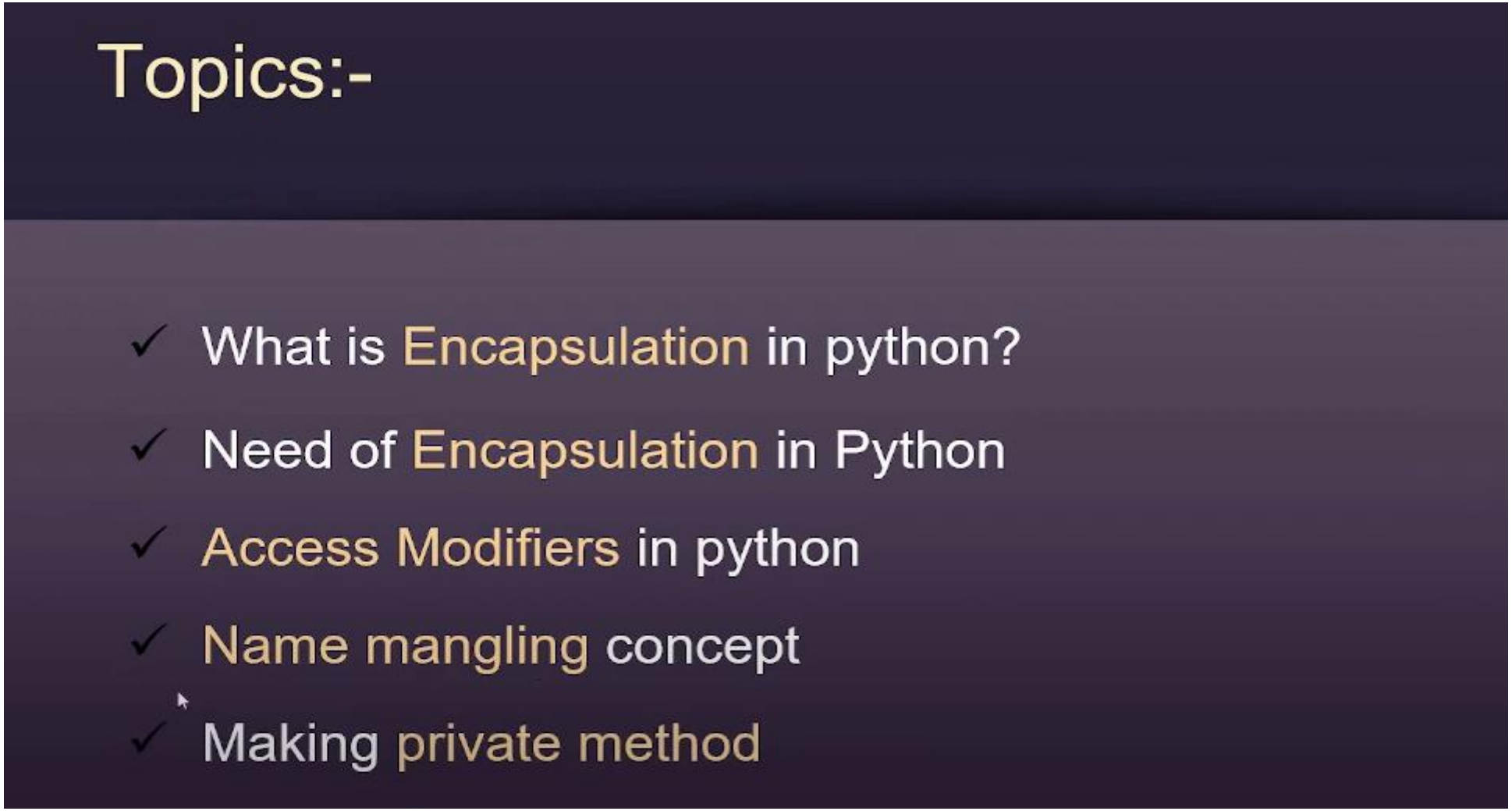
OOPs

Manual Way will not give correct order always

OOPs



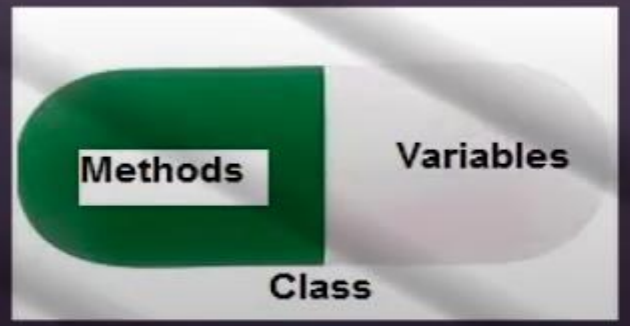
OOPs



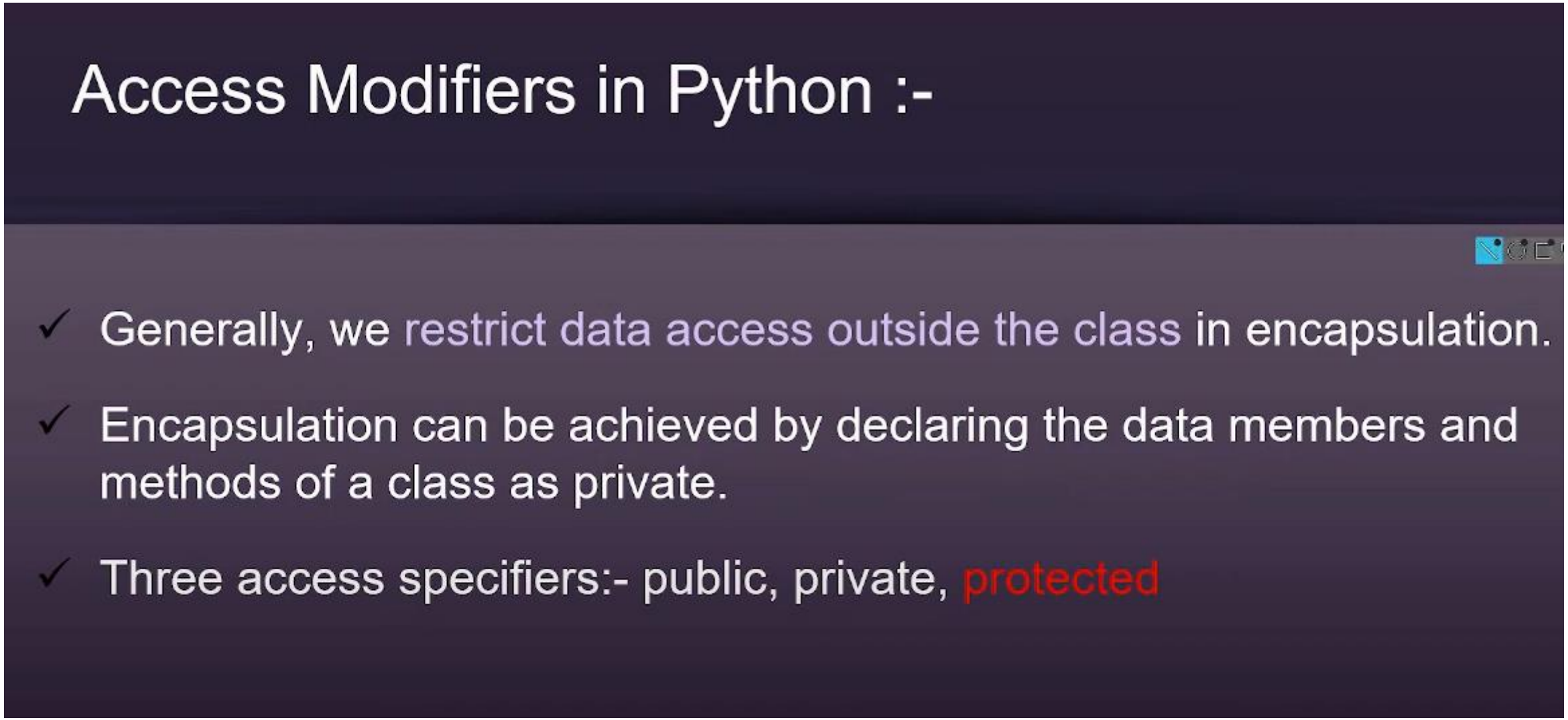
OOPs

What is Encapsulation in python?

- ✓ Wrapping up data and methods working on data together in a single unit (i.e class) is called as encapsulation.



OOPs



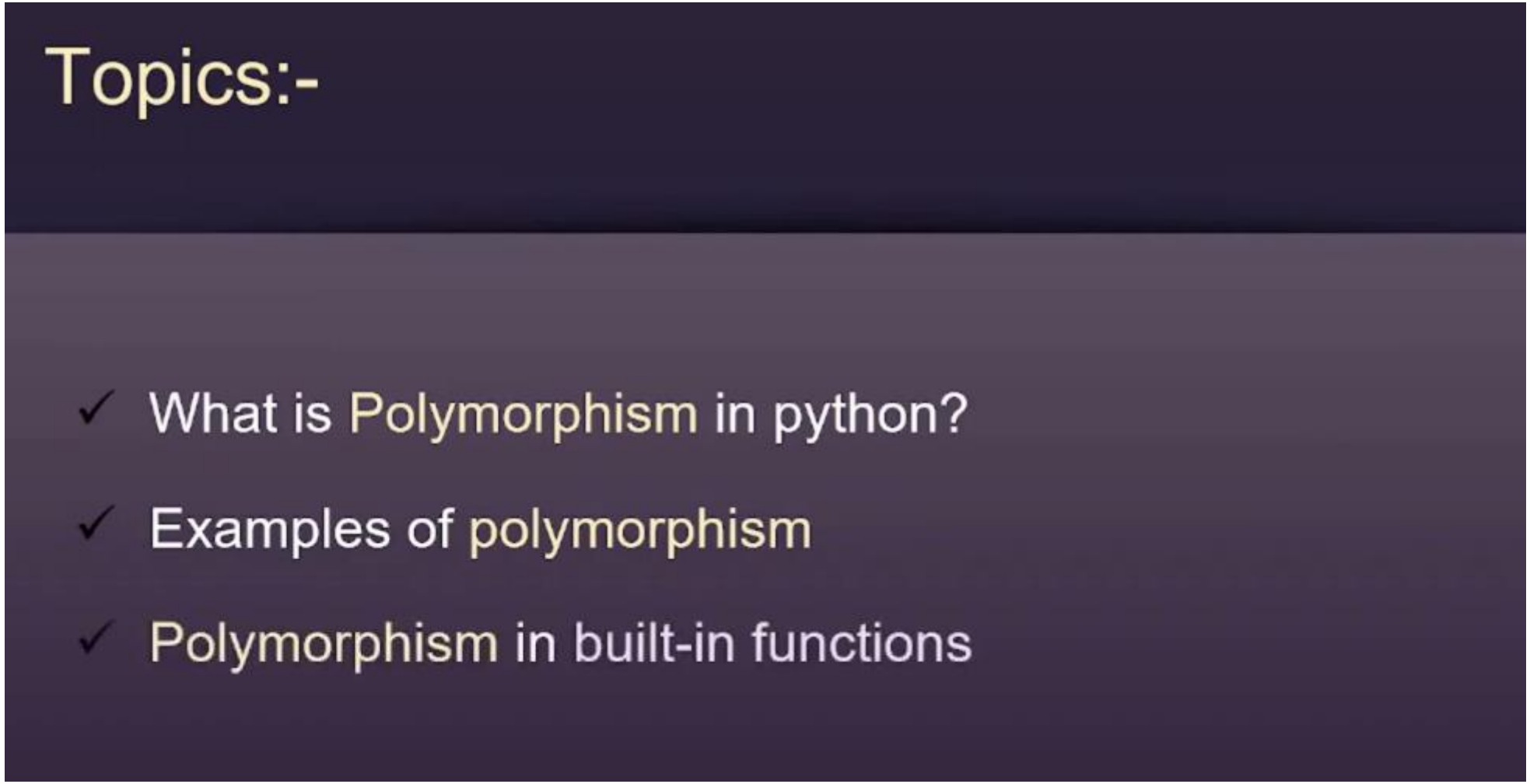
OOPs

Access Modifiers in Python :-

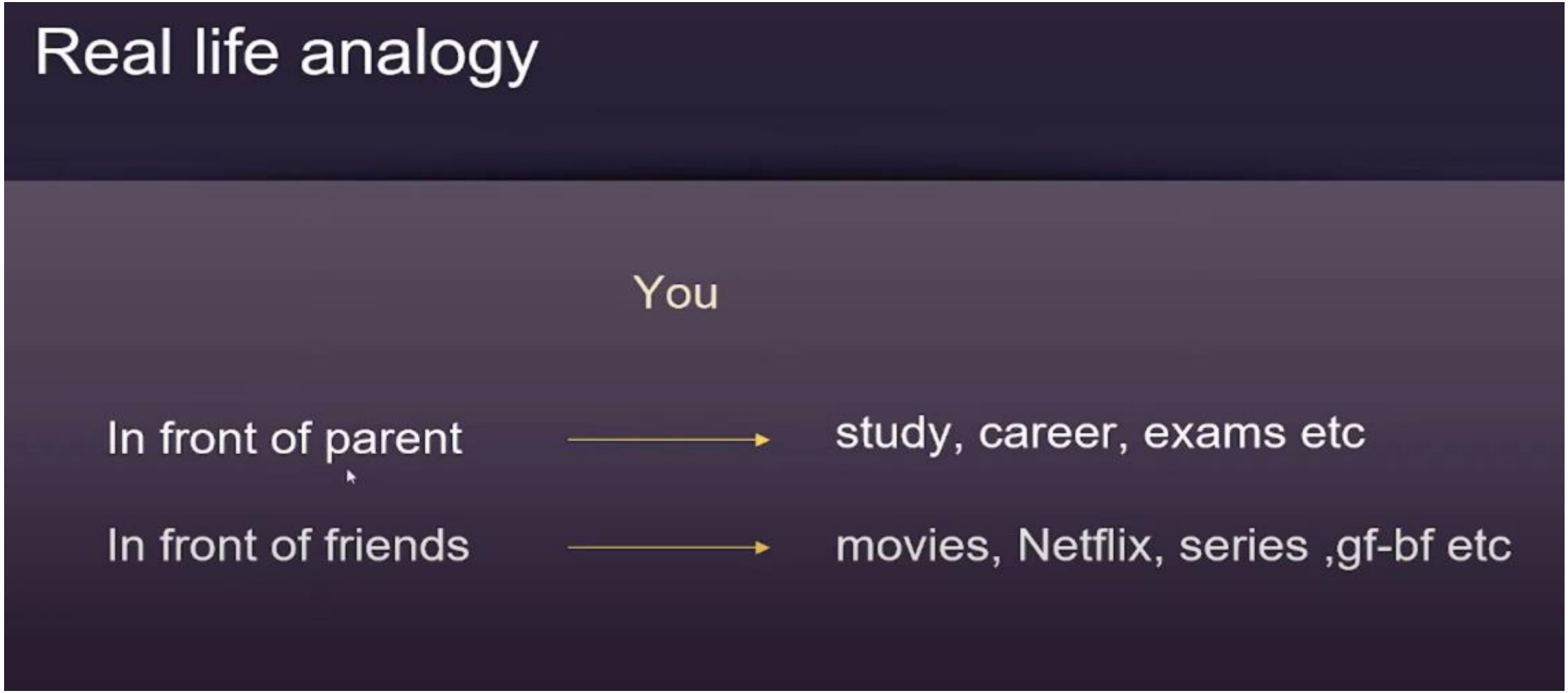
- ✓ Public member:- Accessible anywhere by using object reference.
- ✓ Private member:- Accessible within the class. Accessible via methods only.
- ✓ Protected member:- Accessible within class and it's subclasses

OOPs

OOPs



OOPs

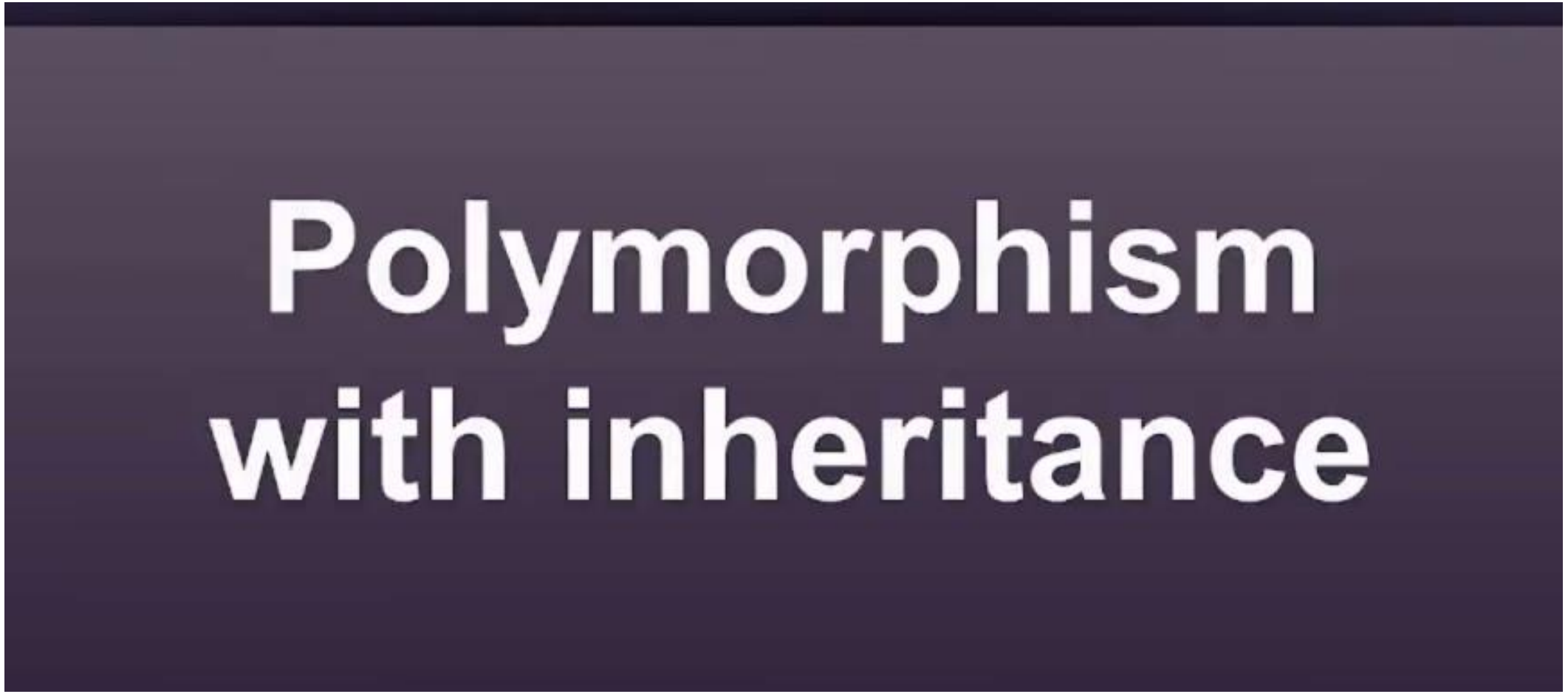


OOPs

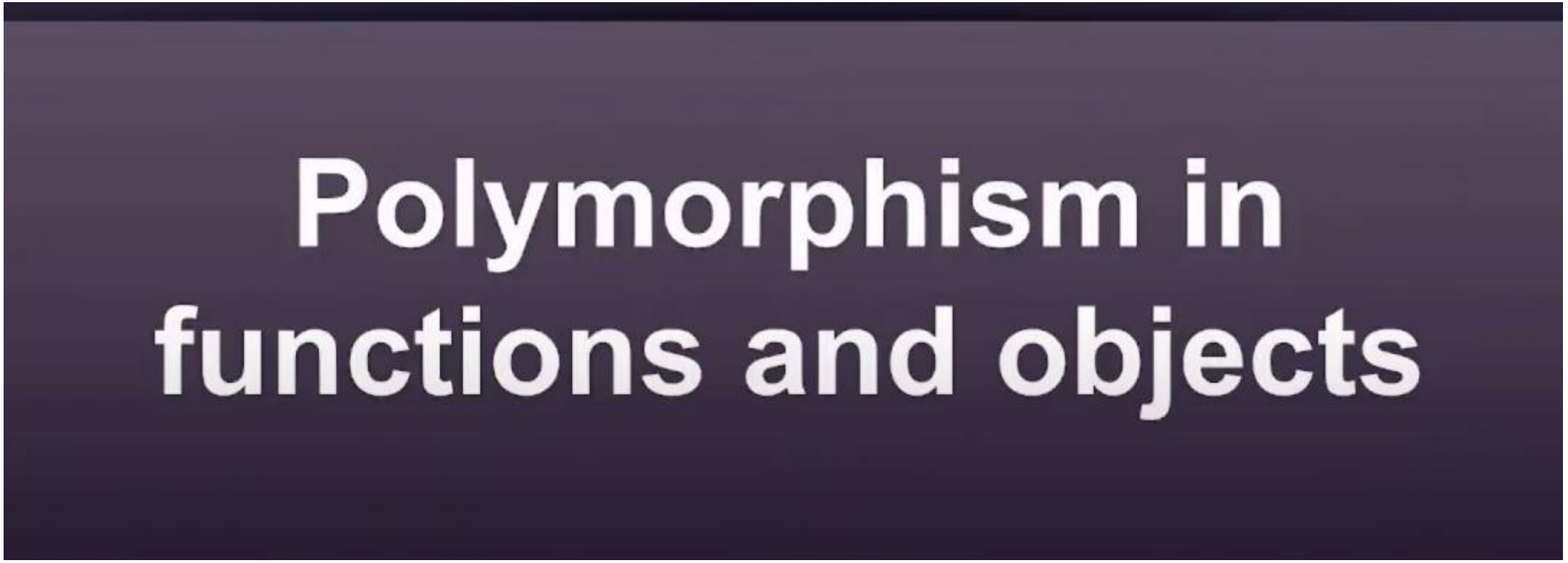
What is Polymorphism in python?

- ✓ Polymorphism in python is an ability of python object to take many forms.
- ✓ If a variable, object, method performs different behaviour according to situation is called as polymorphism.

OOPs

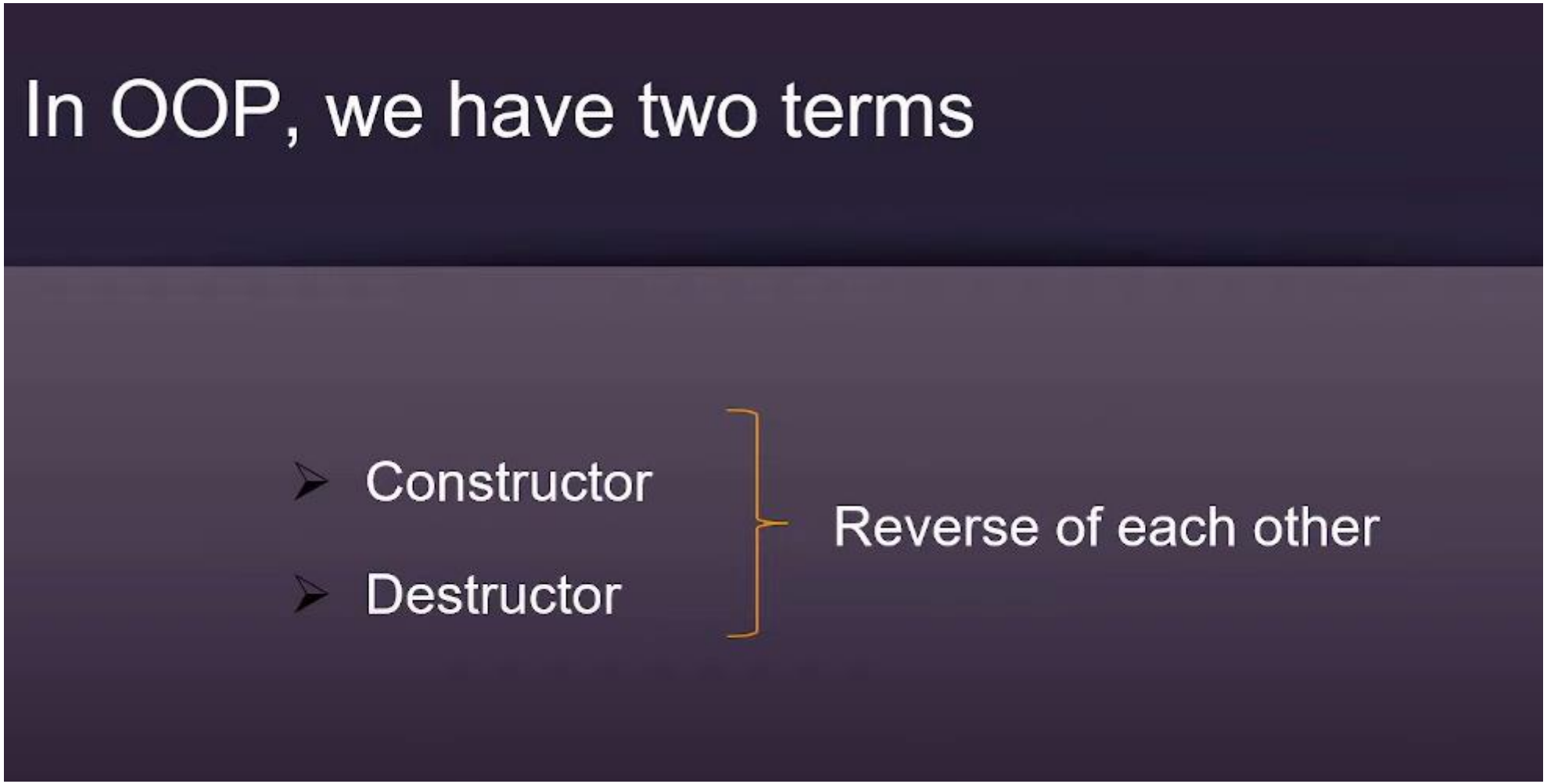


OOPs



OOPs

OOPs

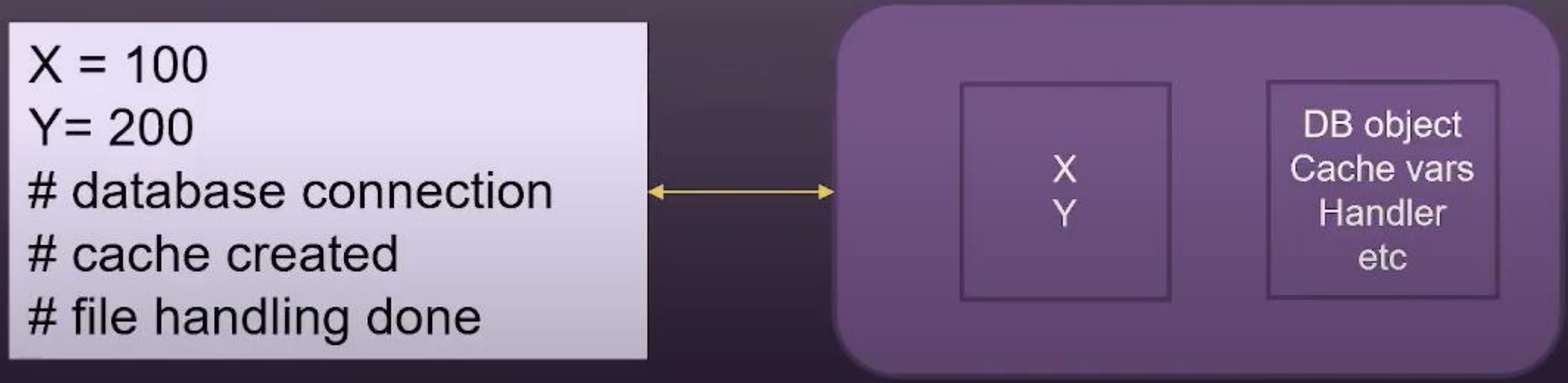


What is Destructor?

- A special method which destroys objects and releases resources tied to objects.
- Destructor is called automatically when object is destroyed.

What is purpose of Destructor ?

- Releasing objects tied to destroyed objects



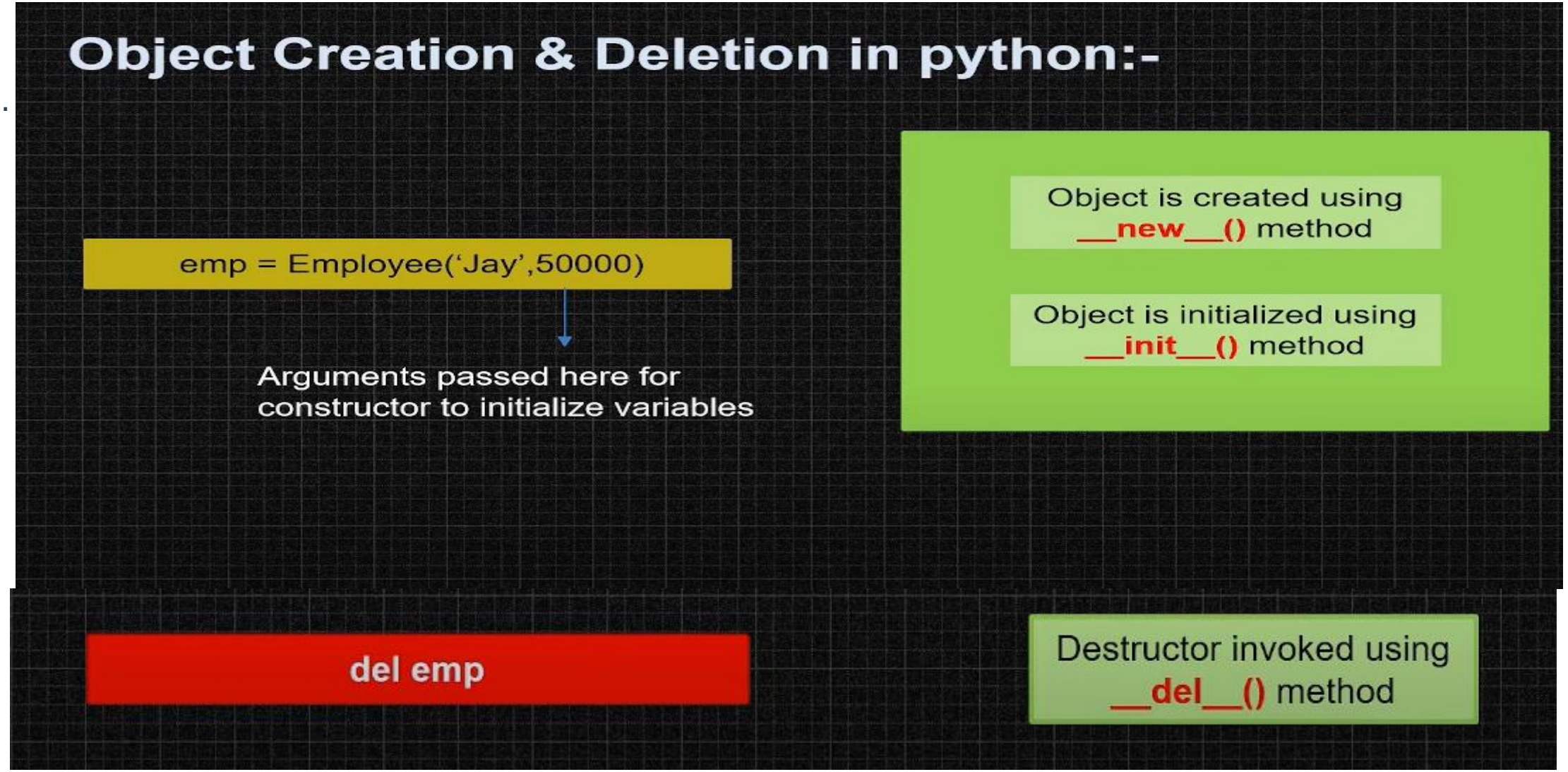
OOPs

Below are two conditions when destructor is called :-

- Reference counting reaches to 0.
- When variable goes out of scope

Note:- In Python, The special method __del__() is used to define a destructor.

OOPs



OOPs