# Python OOP Questions & Answers (Set 5)

## Q1. What is the meaning of multiple inheritance?

Multiple inheritance means a class can inherit from more than one parent class.  
This allows a subclass to combine and reuse behaviors from multiple superclasses.  
Example:  
class A: ...  
class B: ...  
class C(A, B): pass # C inherits from both A and B  
  
Python resolves conflicts with the Method Resolution Order (MRO) using C3 linearization.

## Q2. What is the concept of delegation?

Delegation means an object hands over (delegates) responsibility for some tasks to another object.  
Instead of inheriting, it stores a reference to another object and forwards calls.  
Example:  
class Printer:  
 def print\_text(self, text): print(text)  
  
class Manager:  
 def \_\_init\_\_(self):  
 self.printer = Printer() # delegate  
 def print\_text(self, text):  
 return self.printer.print\_text(text)  
  
Delegation is useful for loose coupling and reusing behaviors without full inheritance.

## Q3. What is the concept of composition?

Composition is a design principle where a class is built from other classes by containing their objects.  
It models 'has-a' relationships rather than 'is-a'.  
Example:  
class Engine: ...  
class Car:  
 def \_\_init\_\_(self):  
 self.engine = Engine() # Car has an Engine  
  
Composition encourages modularity and reuse while avoiding deep inheritance trees.

## Q4. What are bound methods and how do we use them?

A bound method is a function that is tied (bound) to an instance.  
When you access a method through an object, Python automatically passes the instance (self) as the first argument.  
Example:  
class A:  
 def greet(self, msg): print(msg)  
  
obj = A()  
f = obj.greet # bound method  
f('Hello') # obj is automatically passed  
  
Bound methods carry both the function and the instance context.

## Q5. What is the purpose of pseudoprivate attributes?

Pseudoprivate attributes use name mangling: names with \_\_ (double underscore prefix) are rewritten by Python to include the class name.  
Example:  
class A:  
 def \_\_init\_\_(self):  
 self.\_\_x = 10  
  
print(dir(A())) # shows \_A\_\_x  
  
Purpose:  
- Avoid accidental name clashes in subclasses.  
- Provide a weak form of encapsulation (not true private, but harder to access).  
They are still accessible using \_ClassName\_\_attr.