Python OOP Questions & Answers (New Set)

Q1. What is the relationship between classes and modules?

A module is a file (.py) containing Python definitions (functions, variables, classes).
A class is a code structure inside a module that defines objects.
Relationship: Classes live inside modules. Modules group related classes, functions, and data into reusable code units.

Q2. How do you make instances and classes?

Making a class: Use the 'class' keyword.
class MyClass: pass

Making an instance: Call the class like a function.
obj = MyClass()

Q3. Where and how should class attributes be created?

Class attributes belong to the class itself, shared across all instances.
Create them inside the class body but outside any method.
Example:
class Car:
wheels = 4

Q4. Where and how are instance attributes created?

Instance attributes are unique to each object.
Usually created in the __init__ method with self.
Example:
class Car:
def __init__(self, color):
self.color = color

Q5. What does the term 'self' in a Python class mean?

self is the instance reference passed automatically to methods.
It lets methods access and modify instance attributes.

Q6. How does a Python class handle operator overloading?

Python supports special methods (dunder methods) to redefine operator behavior.
Example:
class Vector:
def __init__(self, x, y):
self.x, self.y = x, y
def __add__(self, other):

return Vector(self.x + other.x, self.y + other.y)

Q7. When do you consider allowing operator overloading of your classes?

When the class represents entities where natural operators make sense (vectors, complex numbers, matrices, comparisons).
Avoid if it makes code confusing or less readable.

Q8. What is the most popular form of operator overloading?

The most popular forms are:
- __str__ and __repr__ (string representation)
- Arithmetic operators (__add__, __sub__, etc.)
These make objects print nicely or support math-like operations.

Q9. What are the two most important concepts to grasp in order to comprehend Python OOP code?

1. Attribute Lookup & MRO (Method Resolution Order): how Python searches attributes in instances, classes, and superclasses.
2. Binding of Methods: how functions inside classes become bound methods with self (instance) or cls (class).