## Q1. What is the purpose of Python's OOP?

Organize code around objects (state+behavior) to improve reuse, readability, and maintainability via encapsulation, abstraction, inheritance, and polymorphism. Models real-world entities and enables extensible designs.

## Q2. Where does an inheritance search look for an attribute?

Lookup order (descriptor-based): Instance \_\_dict\_\_ → Class \_\_dict\_\_ → Superclasses following the class's MRO (C3 linearization: left-to-right, depth-first, with monotonicity).

## Q3. How do you distinguish between a class object and an instance object?

- A class is a blueprint (created by a class statement); calling it creates instances.  
- An instance is a concrete object produced by calling the class.  
- type(instance) returns the class; type(Class) is type (the metaclass).  
- isinstance(obj, Class) vs. issubclass(Class, Base).  
- Instances typically have per-object state in obj.\_\_dict\_\_; classes define shared attributes/methods.

## Q4. What makes the first argument in a class’s method function special?

By convention it's self for instance methods and cls for class methods. When accessed via the class/instance, Python's descriptor protocol binds the method and automatically passes the instance (or class) as the first argument.

## Q5. What is the purpose of the \_\_init\_\_ method?

Initializer called after instance creation to set up state (attributes, validation). It's not the constructor (\_\_new\_\_ constructs the instance). Must return None.

## Q6. What is the process for creating a class instance?

obj = Class(\*args, \*\*kwargs) triggers:  
1) Class.\_\_new\_\_ to create the instance (usually from object).  
2) Class.\_\_init\_\_ to initialize it with the given arguments.

## Q7. What is the process for creating a class?

The class statement executes the body to build a namespace dict, then calls the metaclass (default type) to create the class object:  
class Name(Base1, Base2, metaclass=Meta): ...  
Include docstrings, attributes, methods, decorators, and class variables.

## Q8. How would you define the superclasses of a class?

List base classes in parentheses in the class header:  
class Sub(Base1, Base2): pass  
At runtime, inspect via Sub.\_\_bases\_\_ and navigate with Sub.\_\_mro\_\_/mro(). Use super() to delegate to the next class in the MRO.