Q1. What is the concept of a metaclass?

🡪 Metaclass is a class that defines the behaviour of other classes, also known as its instances. It is essentially a class that creates and controls other classes.

classes are objects themselves, and they are instances of metaclasses. By default, the metaclass of a class is the built-in type metaclass. However, you can define your own metaclasses by creating a class that subclasses the type metaclass or any other metaclass.

Q2. What is the best way to declare a class's metaclass?

🡪 A way to declare a class’ metaclass is by using metaclass keyword in class definition.

Q3. How do class decorators overlap with metaclasses for handling classes?

🡪 Class decorators are applied directly to class definitions using the @decorator syntax. They allow you to modify or wrap the class itself, typically by adding or modifying attributes, methods, or other behaviours. Class decorators are executed immediately after the class is defined, providing a way to modify the class object directly.

Metaclasses, are used to control the creation of classes themselves. They allow you to customize the process of class creation, including modifying class attributes, methods, inheritance, and other class-level behaviour. Metaclasses operate at a higher level than decorators, as they are responsible for creating the class object itself.

Q4. How do class decorators overlap with metaclasses for handling instances?

🡪 Class decorators primarily focus on modifying the class itself, and their impact on instances is limited to the attributes, methods, or behaviours added or modified on the class. When a class decorator is applied, it affects all instances created from that class.

Metaclasses allow you to customize the creation and behaviour of instances at a deeper level. By defining a metaclass, you can control how instances are created and customize their attributes, methods, or behaviours.