Q1. What is the concept of a metaclass?

ANS :- In object-oriented programming, a metaclass is a class whose instances are classes. Just as an ordinary class defines the behavior of certain objects, a metaclass defines the behavior of certain classes and their instances

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

ANS class MyMeta(type): pass

class MyClass(metaclass=MyMeta): pass

class MySubclass(MyClass): pass.

print(type(MyMeta))

print(type(MyClass))

print(type(MySubclass))

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

Ans

Class decorated  
Class decorators are functions or callable objects that are applied to a class definition using the @ syntax. They allow you to modify or augment the behavior of a class by wrapping it with additional functionality. Decorators are applied at the time of class creation and can alter the class attributes, methods, or add new functionality to the class. However, decorators operate at a higher level than metaclasses and do not directly manipulate the class creation process.

Meta classes:

Meta classes, on the other hand, are mechanisms that define the creation and behavior of classes themselves. They operate at a lower level than decorators and are responsible for defining how classes are created, what attributes and methods they have, and how they behave. Metaclasses allow you to intervene in the class creation process by overriding methods such as \_\_new\_\_ or \_\_init\_\_ in the metaclass definition. They provide a way to modify the behaviour of classes globally, affecting all instances of the class.

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

Ans

class decorators and metaclasses offer flexibility and customization, they primarily focus on class objects and their creation rather than directly handling instances.