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. Not all object-oriented programming languages support metaclasses.

A metaclass in Python is **a class of a class that defines how a class behaves**. A class is itself an instance of a metaclass. A class in Python defines how the instance of the class will behave.

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

Ans:- In order to set metaclass of a class, we use the **\_\_metaclass\_\_ attribute**. Metaclasses are used at the time the class is defined, so setting it explicitly after the class definition has no effect.

 A metaclass in Python is a class of a class that **defines how a class behaves**. A class is itself an instance of a metaclass. A class in Python defines how the instance of the class will behave.

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

Ans:-Both are triggered at the end of class statement,class decorator and metaclases can both be used to manage classes. Decorator rewind a class name to a callable’s result and metaclasses route class creation through a callable, but both hooks can be used for similar purposes. To manage classes decorator simply argument and return the original class objects.Metaclasses argument a class after they create it

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

Ans:- Both are triggered at the end of class statement,class decorator and metaclases can both be used to manage class instances by inserting a wrapper object to catch instance creation calls.Decorator may rewind the class name to a callable run on instance creation that retains the original class object.Metaclasses can do the same but they must also create the class object, so their usage is somewhat more complex in this role.