Q1. What is the concept of a metaclass?

In python everything is associated with type. In other programming languages, int, str, array are data type. But, in python, they are objects of the class int, str, list, dict. We can create user defined type by creating a class and then object of that class.

A Class is also an object, and just like any other object it’s a instance of something called Metaclass. A special class type creates these Class object. The type class is default metaclass which is responsible for making classes.

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

When defining a class and no metaclass is defined the default type metaclass will be used. If a metaclass is given and it is not an instance of type(), then it is used directly as the metaclass.

class A(type):  
 pass  
class B(metaclass=A):  
 pass  
class C(B):  
 pass  
print(type(A))  
print(type(B))  
print(type(C))

<class 'type'>

<class '\_\_main\_\_.A'>

<class '\_\_main\_\_.A'>

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

Both decorators and metaclass have some common things. When we want to aplly some common functionality to attributes of the class instances, both can be used. There are some problems which can be solved by decorators as well as by metaclasses. But there are a few problems that can only be solved by metaclasses. Use of metaclass affects its children while the decorator affects only the current class.

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

Decorators and metaclass have certain features in common. Both may be utilised when we wish to apply some common functionality to class instance properties. There are several issues that decorators and metaclasses can both help with. Yet, certain issues can only be resolved via metaclasses. While the decorator just affects the current class, the use of a metaclass impacts its offspring.