Q1. What is the concept of a metaclass?

Ans: A metaclass in Python is a class of classes that defines the behavior of the class. A class is itself an instance of a Metaclass, and all instances of Class in Python are instances of type metaclass. For example, types such as int, str, float, list, and tuple are metaclass types.

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

Ans: metaclass keyword in class definition

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

Ans: Decorators are much simpler and more specific. Therefore, decorators should be preferred whenever either metaclasses or class decorators can be used to achieve the desired effect.

Anything that can be done with a class decorator can of course also be done with a custom metaclass (just apply the functionality of the metaclass's "decorator function" d \_\_new\_\_ or \_\_init\_\_ that creates the class object!-).

There are many things you can do with custom metaclasses that decorators can't

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

Ans: Everything that can be done with a class decorator can also be done with a custom metaclass, of course (just apply the functionality of a "decorator function", i.e. the ability to get a class object in the metaclass history and modify it to create a class object with \_\_new\_\_ or \_\_init\_\_! ).