Q1. What is the concept of a metaclass?
Answer: Metaclass in Python is a class of a class that defines how a class behaves. A class is itself a instance of metaclass, any instance of class in python is an instance of type metaclass. E.g. type of int,str,float, list, tuple and many more is metaclass type.

Q2. What is the best way to declare a class's metaclass?
Answer: A way to declare a class' metaclass is by using metaclass keyword in class definition.
E.g.

```
class Meta1(type):
        pass

class                                                    Class1(metclass=meta1):
        pass

print(type(Meta1)) #output will be <class 'type'>

print(type(Class1)) #output will be <class '__main__.Meta1>
```

Q3. How do class decorators overlap with metaclasses for handling classes?
Answer : Anything you can do with a class decorator, you can of course do with a custom metaclass (just apply the functionality of the "decorator function", i.e., the one that takes a class object and modifies it, in the course of the metaclass's __new__ or __init__ that make the class object!-).

Q4. How do class decorators overlap with metaclasses for handling instances?
Answer : Anything you can do with a class decorator, you can of course do with a custom metaclass (just apply the functionality of the "decorator function", i.e., the one that takes a class object and modifies it, in the course of the metaclass's __new__ or __init__ that make the class object!-).