Q1. What is the concept of a metaclass?

Answer: 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. In order to understand metaclasses well, one needs to have prior experience working with Python classes.

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

Answer: The key concept of python is objects. Almost everything in python is an object, which includes functions and as well as classes. As a result, functions and classes can be passed as arguments, can exist as an instance, and so on. Above all, the concept of objects let the classes in generating other classes.

The classes that generate other classes are defined as metaclasses. In this section, we will discuss the concept of metaclasses and the specific ways to use them. In this section, we will cover the following topics:

type Writing Metaclasses Metaclass Usecases

In Python, we can customize the class creation process by passing the metaclass keyword in the class definition. This can also be done by inheriting a class that has already passed in this keyword. We can see below that the type of MyMeta class is type and that the type of MyClass and MySubClass is MyMeta .

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!-)