Q1. What is the concept of a metaclass?

Answer:

In Python, a metaclass is a class of a class that defines how a class behaves. A class is itself an instance of a metaclass. Everything is an object in Python which includes classes as well. Thus, if classes are an object then they must be created by another class which also called a metaclass. Therefore, a metaclass is just another class that creates class objects. As for example, type, int, str, float, list, tuple, etc. are the built-in metaclass in Python.

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

Answer:

The keyword metaclass can be used in the class definition to declare a class' metaclass.

Example:

class A(type):

pass

class B(metaclass=A): # here, class A is a metaclass for class B

pass

class C(B):

pass

print(type(A))

print(type(B))

print(type(C))

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

Answer:

Python classes can be modified using both metaclasses and class decorators. Functions known as class decorators take a class object as an input and output a changed class object. The @decorator\_name syntax is used to apply them to classes before the class definition. Metaclasses, on the other hand, are classes that specify a class's behaviour. They can be modified to change the behaviour of the class they produce and are used to build new classes dynamically. While the decorator just affects the current class, the use of a metaclass impacts its children.

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

Answer:

A function that takes a class object and produces a new class object is called a class decorator. The behaviour of instances of the original class can be altered using this new class object by adding new methods, changing existing methods, or changing the class attributes. The behaviour of instances of a class can also be altered by altering the class itself using metaclasses. Metaclasses have the ability to change the class attributes, add new methods, or modify existing methods. When a class instance is formed, the metaclass has the ability to alter the instance's behaviour by adding new instance methods or changing instance attributes.