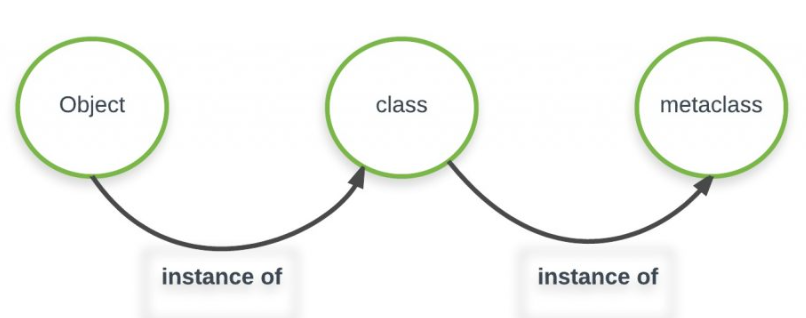
Q1. What is the concept of a metaclass?

A:

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&#39;s metaclass?

A:

num = 23

print("Type of num is:", type(num))

lst = [1, 2, 4]

print("Type of lst is:", type(lst))

name = "Atul"

print("Type of name is:", type(name))

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

A:

Decorators essentially automate the prior example's manual name rebinding here. Just like with metaclasses, because the decorator returns the original class, instances are made from it, not from a wrapper object. In fact, instance creation is not intercepted at all

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

A:

**\_\_new\_\_** allows subclasses of immutable types to customize instance creation. It can be overridden in custom metaclasses to customize class creation.