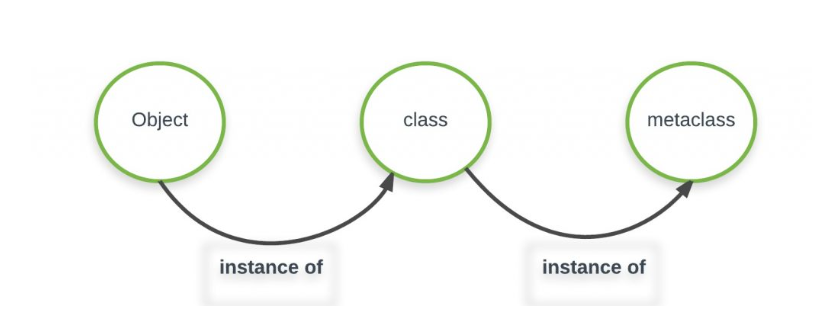
Q1. What is the concept of a metaclass?

Ans: **A metaclass in Python is a class of a class which defines how a class behaves. A class itself is an instance of a metaclass.**



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

Ans: W**e use \_\_metaclass\_\_ attribute to set metaclass of a class.**

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

Ans: **Metaclass** affects its children while the decorator affects only the current class.

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

Ans: **Metaclass** affects its children while the decorator affects only the current class.

The difference between metaclass and decorators can draw a line of distinction between them. Therefore answer the question of the repercussions of overlapping.

The decorator approach

* Syntax differs from ordinary class statement.
* Awkward if class decorators are not available.
* As is, the name is not picked up.
* Easier to construct Bunch classes dynamically.
* The Point class is an instance of type.

The \_\_metaclass\_\_ approach

* Syntax the same as ordinary class statement.
* ‘Magic’ takes place behind the scenes.
* Requires more knowledge to implement.
* Awkward to construct Bunch classes dynamically.
* The Point class is an instance of MetaBunch.