1. Metaclass in Python is a class of a class that defines how a class behaves. A class is itself an instance of Metaclass, and 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 of metaclass type.

2.  A way to declare a class’ metaclass is by using **metaclass** keyword in class definition.

Example:

**class** meta(type):

**pass**

**class** class\_meta(metaclass**=**meta):

**pass**

print(type(meta))

print(type(class\_meta))

Output: <class 'type'>

<class '\_\_main\_\_.meta'>

3. Anything you can do with a class decorator, you can of course do with a custom metaclasses (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!).

4. 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!).