

# Python Advanced Assignment 11

## 1. What is the concept of a metaclass?

Ans: Metaclass in Python is a class of a class that defines how a class behaves. A class is itself a 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. What is the best way to declare a class's metaclass?

Ans: A way to declare a class' **metaclass** is by using metaclass keyword in class definition.

```
In [1]: class meta(type):  
        pass  
        class class_meta(metaclass=meta):  
            pass  
        print(type(meta))  
        print(type(class_meta))  
  
<class 'type'>  
<class '__main__.meta'>
```

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

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

Ans: 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!).