Q1. What is the concept of a metaclass?

**A Class is also an object**, and just like any other object, it’s an instance of something called Metaclass. A **special class type** creates these Class objects. The type class is default metaclass which is responsible for making classes. In the above example, if we try to find out the type of Student class, it comes out to be a type.

**class** Student:

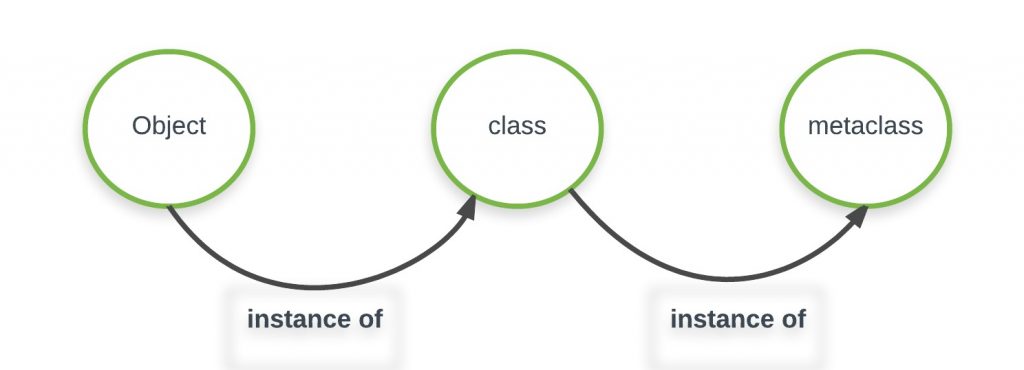
**pass**

# Print type of Student class

print("Type of Student class is:", type(Student))

Type of Student class is: <class 'type'>

This whole meta thing can be summarized as – **Metaclass create Classes and Classes creates objects**



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

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

class meta(type):

pass

class class\_meta(metaclass=meta):

pass

print(type(meta))

print(type(class\_meta))

op:

<class 'type'>

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

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

 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!).

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

 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!).