ASSIGNMENT 11

# Q1

A metaclass is the class of a class. It is responsible for defining the behaviour and structure of a class. In other words, a metaclass defines how a class should be created and what attributes and methods it should have. It is a powerful feature in Python that allows us to customise and control the creation and behaviour of classes.

# Q2

The best way to declare a class's metaclass is by using the metaclass argument in the class definition. we can assign a metaclass to a class by specifying the metaclass as the value of the metaclass argument.

# Q3

Class decorators and metaclasses can both be used to handle classes. However, they have different strengths and weaknesses.

Class decorators are typically used to add functionality to existing classes. For example, a class decorator could be used to add logging to all classes.

Metaclasses are typically used to control the behaviour of classes. For example, a metaclass could be used to ensure that all classes have a certain set of attributes, or to implement a custom inheritance scheme.

# Q4

Class decorators do not directly overlap with metaclasses for handling instances. Class decorators primarily operate on the class object itself, whereas metaclasses handle the creation and behaviour of classes. However, class decorators can indirectly affect instances by modifying the class's behaviour or adding functionality to its methods.

When a class decorator is applied to a class, it can modify the behaviour of the class's methods, attributes, or constructor. These modifications can indirectly affect instances created from the class by altering how methods are called or how attributes are accessed. In this way, class decorators can influence the behaviour of instances but do not have the same level of control as metaclasses.