Q1. What is the difference between \_\_getattr\_\_ and \_\_getattribute\_\_?

Ans1- \_\_getattr\_\_ is called when an attribute is not found via the usual attribute lookup. It is only invoked when the attribute is not already present in the instance's namespace.

\_\_getattribute\_\_ is called for all attribute access, even if the attribute exists in the instance's namespace. It allows you to intercept and customize all attribute access.

Q2. What is the difference between properties and descriptors?

Ans2- Properties are a simpler way to manage attribute access by allowing you to define getter and setter methods using decorators (@property, @<attr>.setter). They are usually bound to a single attribute.

Descriptors are more versatile and provide greater control over attribute access. They are defined as separate classes and can be reused across multiple attributes or objects.

Q3. What are the key differences in functionality between \_\_getattr\_\_ and \_\_getattribute\_\_, as well as properties and descriptors?

Ans3- \_\_getattr\_\_ vs. \_\_getattribute\_\_:

\_\_getattr\_\_ is called only when an attribute is not found, providing a fallback mechanism.

\_\_getattribute\_\_ is called for all attribute access, including existing attributes, giving full control over attribute access.

Properties vs. Descriptors:

Properties are typically used for simple attribute access and provide a convenient way to add getter and setter methods.

Descriptors are more powerful and flexible, allowing you to customize attribute access, reuse descriptor classes, and control access at the class level.