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

Answer 1: The main difference between \_\_getattr\_\_ and \_\_getattribute\_\_ in Python is when they are called. \_\_getattr\_\_ is called only when the requested attribute is not found through the usual means, while \_\_getattribute\_\_ is called every time an attribute is accessed.

**Q2. What is the difference between properties and descriptors?**

Answer 2: Properties and descriptors are both ways to manage attribute access in Python.

- Properties are a simpler way to define read-only or read-write attributes on a class. They use the @property, @attribute\_name.setter, and @attribute\_name.deleter decorators.

- Descriptors are more general and allow you to define how attribute access is handled for both getting and setting values. Descriptors are defined as separate classes with \_\_get\_\_, \_\_set\_\_, and \_\_delete\_\_ methods.

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

Answer 3:

The key differences in functionality between \_\_getattr\_\_ and \_\_getattribute\_\_, as well as properties and descriptors:

* \_\_getattr\_\_ is called only when the requested attribute is not found, while \_\_getattribute\_\_ is called for every attribute access.
* Properties provide a simple way to manage attribute access with predefined methods for getting, setting, and deleting.
* Descriptors offer more flexibility and control over attribute access by allowing custom methods for getting, setting, and deleting. They are defined in a separate class and can be reused across multiple attributes.