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

The main difference between `\_\_getattr\_\_` and `\_\_getattribute\_\_` is how they handle attribute access in Python. `\_\_getattr\_\_` is called when an attribute is not found in the usual places, such as the instance or class dictionary. `\_\_getattribute\_\_` is called for every attribute access and allows customization, but requires caution to avoid infinite recursion when accessing existing attributes.

Q2. What is the difference between properties and descriptors?

Properties and descriptors are both mechanisms in Python for managing attribute access, but they differ in their level of control. Properties are a simple way to define computed or derived attributes with getter, setter, and deleter methods. Descriptors, on the other hand, provide more fine-grained control by allowing customization of attribute access, including defining custom getters, setters, and deleters at the class level.

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

- `\_\_getattr\_\_` is called when an attribute is not found in the usual places, while `\_\_getattribute\_\_` is called for every attribute access.

- Properties provide a way to define computed or derived attributes with getter, setter, and deleter methods.

- Descriptors offer more control by allowing customization of attribute access at the class level, including defining custom getters, setters, and deleters.