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

The main difference between \_\_getattr\_\_ and \_\_getattribute\_\_ lies in when they are invoked. \_\_getattr\_\_ is only called when an attribute is not found through normal lookup, while \_\_getattribute\_\_ is called for every attribute access, even if the attribute exists. This means that \_\_getattribute\_\_ can potentially interfere with accessing existing attributes.

Q2. What is the difference between properties and descriptors?

Properties and descriptors are both ways to manage attribute access, but they differ in their level of control. Properties are a simpler mechanism that allows you to define getter, setter, and deleter methods for an attribute, offering a convenient interface for attribute access. Descriptors provide more granular control over attribute access by defining the \_\_get\_\_, \_\_set\_\_, and \_\_delete\_\_ methods in a separate descriptor class.

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

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

* \_\_getattr\_\_ is called when an attribute is not found; \_\_getattribute\_\_ is called for every attribute access.
* Properties provide a high-level interface for attribute access; descriptors offer a more customizable and lower-level approach.
* \_\_getattribute\_\_ can potentially lead to infinite recursion if not used carefully; properties and descriptors provide more controlled ways to handle attribute access.
* \_\_getattr\_\_ and \_\_getattribute\_\_ are often used for different purposes, such as handling missing attributes and intercepting all attribute access, respectively.
* Properties and descriptors are often used to encapsulate logic and validation around attribute access, promoting better code organization and maintainability.