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

* `\_\_getattr\_\_` is used for handling attribute access when the attribute is not found, while `\_\_getattribute\_\_` intercepts all attribute accesses, both existing and non-existing, providing more control but requiring careful implementation to avoid unintended side effects.

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

1. Properties:

- Properties are a simpler and more common way to control attribute access.

- They are defined using the `@property` decorator and a method with the same name.

- Properties allow you to customize the behaviour of attribute access (getting) without changing the way the attribute is accessed.

2. Descriptors:

- Descriptors are a more advanced and powerful way to control attribute access.

- They are defined using special methods like `\_\_get\_\_`, `\_\_set\_\_`, and `\_\_delete\_\_` within a descriptor class.

- Descriptors can customize attribute access (getting and setting) and can be applied to multiple attributes of a class, providing more control and flexibility.

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

1. `\_\_getattr\_\_` vs. `\_\_getattribute\_\_`:

* - `\_\_getattr\_\_` is called when accessing an attribute that doesn't exist, allowing you to handle attribute access errors.
* - `\_\_getattribute\_\_` is called for all attribute access (existing and non-existing) and provides more control but should be used with caution to avoid infinite recursion.

2. Properties vs. Descriptors:

* - Properties are a simpler way to customize attribute access, mainly for getting.
* - Descriptors provide more comprehensive control over attribute access, including getting, setting, and deleting.
* - Descriptors can be applied to multiple attributes and offer more flexibility for complex attribute behavior customizations.