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

Sol: `\_\_getattr\_\_` and `\_\_getattribute\_\_` are both special methods in Python classes that allow you to define custom behavior when accessing an attribute of an object. However, they differ in some important ways:

1. `\_\_getattr\_\_(self, name)` is called when the requested attribute is not found by the usual means, i.e., when a non-existent attribute is accessed using the dot notation. It takes a single argument, `name`, which is the name of the attribute that was requested. If `\_\_getattr\_\_` is defined, it will be called instead of raising an `AttributeError`.

2. `\_\_getattribute\_\_(self, name)` is called for every attribute access, regardless of whether the attribute exists or not. It takes a single argument, `name`, which is the name of the attribute that was requested. If `\_\_getattribute\_\_` is defined, it will always be called before any other attribute access method, including `\_\_getattr\_\_`. Therefore, if `\_\_getattribute\_\_` is defined, it is responsible for handling all attribute accesses.

Q2. What is the difference between properties and descriptors?

Sol: Both properties and descriptors are Python language features that allow you to define custom behavior for accessing and setting attributes of an object. However, they differ in some important ways:

1.Properties are a simpler and more limited mechanism than descriptors. They are defined using the `@property` decorator, and allow you to define custom behavior for getting and setting an attribute value. When a property is accessed, the getter method is called, and when it is set, the setter method is called. Properties are bound to a specific instance of a class, and are accessed using the dot notation.

2.Descriptors are a more powerful mechanism than properties, and allow you to define custom behavior for attribute access that can be shared across multiple classes and instances. They are defined using a descriptor class, which must implement at least one of the special methods `\_\_get\_\_()`, `\_\_set\_\_()`, or `\_\_delete\_\_()`. Descriptors are accessed using the dot notation, and are not bound to a specific instance of a class.

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

Sol: differences between `\_\_getattr\_\_` and `\_\_getattribute\_\_`, as well as properties and descriptors:

1.`\_\_getattr\_\_` is called only when an attribute cannot be found through the usual lookup mechanism, while `\_\_getattribute\_\_` is called for every attribute access, even if the attribute exists.

2. `\_\_getattribute\_\_` can be used to intercept all attribute access and customize the behavior of the object, while `\_\_getattr\_\_` is used to customize the behavior only for non-existent attributes.

3. Properties allow you to define custom behavior for getting and setting an attribute value, but are bound to a specific instance of a class, and are accessed using the dot notation.

4. Descriptors are more powerful than properties and allow you to define custom behavior for attribute access that can be shared across multiple classes and instances. Descriptors are accessed using the dot notation and are not bound to a specific instance of a class. Descriptors can define custom behavior for getting, setting, and deleting attributes.