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

In Python, \_\_getattr\_\_ and \_\_getattribute\_\_ are both special methods used for attribute access, but there are some key differences:

* \_\_getattr\_\_ is only called when an attribute is not found through the usual lookup process (i.e., it is not in the instance's dict and not in its class's dict). It is meant to provide a fallback mechanism for handling attribute access.
* \_\_getattribute\_\_ is called for every attribute access, regardless of whether the attribute is found or not. It is used to customize the behavior of attribute access for all attributes, not just those that are missing.

One important thing to note is that overriding \_\_getattribute\_\_ can be risky, as it can cause infinite recursion if not implemented carefully. If you need to customize attribute access, it is often safer to use \_\_getattr\_\_ instead.

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

In Python, properties and descriptors are both mechanisms that allow you to define custom behavior for attribute access, but there are some key differences:

* Properties are a simple way to add computed attributes to a class. They allow you to define a method that can be accessed like an attribute. When the property is accessed, the method is called and its return value is returned instead of the property itself. Properties are defined using the @property decorator.
* Descriptors are a more powerful mechanism for customizing attribute access. They allow you to define the behavior for getting, setting, and deleting an attribute. Descriptors are defined as classes that implement at least one of the \_\_get\_\_, \_\_set\_\_, or \_\_delete\_\_ methods. When an attribute with a descriptor is accessed, its \_\_get\_\_ method is called to retrieve the attribute's value. Similarly, when the attribute is set or deleted, its \_\_set\_\_ or \_\_delete\_\_ method is called.

One key difference between properties and descriptors is that properties are defined on the class itself, while descriptors are defined on separate objects that are then attached to the class. This means that descriptors can be used to customize attribute access for multiple attributes across different classes, while properties are typically used for a single attribute within a single class.

Another difference is that properties are read-only by default, meaning that they can't be set directly. However, you can define a setter method for a property using the @<property>.setter decorator. Descriptors, on the other hand, can be used to define read-write or write-only attributes, depending on how their \_\_get\_\_ and \_\_set\_\_ methods are defined.

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

Here are the key differences in functionality between \_\_getattr\_\_, \_\_getattribute\_\_, properties, and descriptors in Python:

* \_\_getattr\_\_ is called only when an attribute is not found through the usual lookup process, and is intended to provide a fallback mechanism for handling attribute access. In contrast, \_\_getattribute\_\_ is called for every attribute access, and is used to customize the behavior of attribute access for all attributes, not just those that are missing.
* Properties are a simple way to add computed attributes to a class, and are defined using the @property decorator. They allow you to define a method that can be accessed like an attribute, and when the property is accessed, the method is called and its return value is returned instead of the property itself. Properties are read-only by default, but you can define a setter method using the @<property>.setter decorator.
* Descriptors are a more powerful mechanism for customizing attribute access, and allow you to define the behavior for getting, setting, and deleting an attribute. They are defined as classes that implement at least one of the \_\_get\_\_, \_\_set\_\_, or \_\_delete\_\_ methods, and can be used to customize attribute access for multiple attributes across different classes. Descriptors can be used to define read-write or write-only attributes, depending on how their \_\_get\_\_ and \_\_set\_\_ methods are defined.

In summary, \_\_getattr\_\_ and \_\_getattribute\_\_ are used to customize attribute access for specific cases, while properties and descriptors are used to add custom behavior to attributes in a more general way. Properties are simpler and read-only by default, while descriptors are more powerful and can be used to define read-write or write-only attributes.