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

Q2. What is the difference between properties and descriptors?

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

Answers:

1. The main difference between **\_\_getattr\_\_** and **\_\_getattribute\_\_** is that **\_\_getattr\_\_** is only called if the attribute is not found by searching the instance or class, while **\_\_getattribute\_\_** is called for every attribute access. In other words, **\_\_getattribute\_\_** is called first and is always called, while **\_\_getattr\_\_** is only called if **\_\_getattribute\_\_** cannot find the attribute.
2. Properties and descriptors are two different ways to manage attribute access in Python. Properties allow you to define a method that looks like an attribute, but is actually a method that gets called every time the attribute is accessed. Descriptors, on the other hand, are a more general way to manage attribute access. A descriptor is an object that defines one or more of the special methods **\_\_get\_\_**, **\_\_set\_\_**, or **\_\_delete\_\_**. When an attribute of an object is accessed, the descriptor's **\_\_get\_\_**, **\_\_set\_\_**, or **\_\_delete\_\_** method is called instead of the object's own attribute access methods.
3. The key differences in functionality between **\_\_getattr\_\_** and **\_\_getattribute\_\_** are that **\_\_getattr\_\_** is only called for attributes that are not found by the normal attribute lookup process, while **\_\_getattribute\_\_** is called for every attribute access. Properties and descriptors are both ways to define special behavior for attribute access, but properties are simpler and more limited in scope than descriptors. Properties allow you to define a method that looks like an attribute, but is actually a method that gets called every time the attribute is accessed. Descriptors, on the other hand, are a more general way to manage attribute access and can define complex behavior for attribute access.