1. The main difference between `\_\_getattr\_\_` and `\_\_getattribute\_\_` is that `\_\_getattr\_\_` is only called when an attribute is not found by normal means, while `\_\_getattribute\_\_` is called for every attribute access, whether the attribute exists or not. `\_\_getattribute\_\_` is usually only used for low-level attribute access, whereas `\_\_getattr\_\_` is used for attribute access when the attribute is not present in the object's namespace.

2. Properties and descriptors are both ways to add custom behavior to attribute access in Python. Properties are a simple way to define a computed attribute that looks like a regular attribute, but whose value is computed on the fly. A descriptor, on the other hand, is a more general mechanism for customizing attribute access. Descriptors can define custom behavior for getting, setting, and deleting attributes, and can be used to implement things like non-data descriptors, which compute the value of an attribute on the fly, or data descriptors, which store the value of an attribute in a separate attribute.

3. The key difference in functionality between `\_\_getattr\_\_` and `\_\_getattribute\_\_` is that `\_\_getattr\_\_` is only called when an attribute is not found by normal means, while `\_\_getattribute\_\_` is called for every attribute access, whether the attribute exists or not. This means that `\_\_getattribute\_\_` is usually only used for low-level attribute access, whereas `\_\_getattr\_\_` is used for attribute access when the attribute is not present in the object's namespace.