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

Ans:

The main difference between \_\_getattr\_\_ and \_\_getattribute\_\_ is that if the attribute was not found by the usual way then \_\_getattr\_\_ is used.

Whereas the \_\_getattribute\_\_ is used before looking at the actual attributes on the object. You will have to use it more consciously otherwise very easily you can end up in infinite recursions.

Q2. What is the difference between properties and descriptors?

Ans:

Descriptors are a low-level mechanism that lets you hook into an object's attributes being accessed. Properties are a high-level application of this; that is, properties are implemented using descriptors

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

A key difference between \_\_getattr\_\_ and \_\_getattribute\_\_ is that \_\_getattr\_\_ is only invoked if the attribute wasn't found the usual ways. It's good for implementing a fallback for missing attributes, and is probably the one of two you want.

\_\_getattribute\_\_ is invoked before looking at the actual attributes on the object, and so can be tricky to implement correctly. You can end up in infinite recursions very easily.

New-style classes derive from object, old-style classes are those in Python 2.x with no explicit base class. But the distinction between old-style and new-style classes is not the important one when choosing between \_\_getattr\_\_ and \_\_getattribute\_\_.

Properties and Descriptor: properties *are* descriptors that are already provided for you in the standard library.

If you need a simple way to return a computed value from an attribute read, or to call a function on an attribute write, use the @property decorator. The descriptor API is more flexible, but less convenient, and arguably "overkill" and non-idiomatic in this situation. It's useful for more advanced use cases, like implementing bound methods, or static and class methods; when you need to know, for example, if the attribute was accessed through the type object, or an instance of the type.