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

***Ans***:

1. \_\_getattr\_\_ (self, name) is called when an attempt is made to access an attribute that does not exist in the object's namespace. It takes a single argument name, which is the name of the attribute being accessed. If this method is defined, it is only called when the attribute is not found by looking up in the object's namespace.
2. \_\_getattribute\_\_ (self, name) is called every time an attribute is accessed on the object, regardless of whether it exists or not. It takes a single argument name, which is the name of the attribute being accessed. If this method is defined, it is always called when an attribute is accessed.

Q2. What is the difference between properties and descriptors?

***Ans***:

The key difference between properties and descriptors is that properties are more limited in their capabilities, while descriptors provide much more flexibility and control. Properties are typically used for simple cases where a computed attribute is needed, while descriptors are used for more complex cases where attribute access needs to be customized.

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

***Ans***:

\_\_getattr\_\_ and \_\_getattribute\_\_ are used to intercept attribute access, while properties and descriptors are used to customize attribute behaviour. \_\_getattr\_\_ is called when an attribute lookup fails, while \_\_getattribute\_\_ is called for every attribute access. Properties provide a simple interface to computed attributes, while descriptors provide a more low-level mechanism for customizing attribute access.