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

**Ans-** Both ‘\_\_getattr\_\_’ and ‘\_\_getattribute\_\_’ are special methods in Python classes that are called when an attribute is accessed on an object. However, they have some key differences in their behaviour.

‘\_\_getattr\_\_’ is called only when an attribute is not found through the usual means, i.e., when the attribute is not an instance attribute or a class attribute. It takes one argument, the name of the attribute being accessed, and must return the value of the attribute or raise an ‘AttributeError’ if the attribute is not found

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

**Ans-** Both properties and descriptors are mechanisms for controlling access to attributes in Python, but they have different purposes and mechanisms. Properties are a simpler mechanism for controlling access to attributes. They allow you to define special methods that are called when an attribute is accessed, set, or deleted. Properties are essentially a syntactic sugar that lets you define these special methods using a decorator-like syntax, instead of having to manually define the \_\_get\_\_, \_\_set\_\_, and \_\_delete\_\_ methods that would be required for a descriptor.

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

**Ans-**\_\_getattr\_\_ and \_\_getattribute\_\_ are both special methods in Python that are called when an attribute is accessed on an object, but they have different purposes and behavior:

‘\_\_getattr\_\_’ is called only when an attribute is not found through the usual means, i.e., when the attribute is not an instance attribute or a class attribute. It takes one argument, the name of the attribute being accessed, and must return the value of the attribute or raise an AttributeError if the attribute is not found.

‘\_\_getattribute\_\_’ is called for every attribute access on an object, whether the attribute is found or not. It takes one argument, the name of the attribute being accessed, and must return the value of the attribute. If \_\_getattribute\_\_ raises an exception, \_\_getattr\_\_ is called if it is defined.

Properties and descriptors are both mechanisms for controlling access to attributes in Python, but they have different purposes and behaviour.

Properties allow you to define special methods that are called when an attribute is accessed, set, or deleted. Properties are essentially a syntactic sugar that lets you define these special methods using a decorator-like syntax, instead of having to manually define the \_\_get\_\_, \_\_set\_\_, and \_\_delete\_\_ methods that would be required for a descriptor.

Descriptors allow you to define a class that implements the \_\_get\_\_, \_\_set\_\_, or \_\_delete\_\_ method, and then use an instance of that class as an attribute on another class. When the attribute is accessed, set, or deleted on an instance of the containing class, the corresponding method is called on the descriptor instance. Descriptors are a more powerful mechanism than properties, as they allow you to define custom behavior for attribute access beyond simple getter and setter methods.