**ADITYA AMIN**

**ASSIGN : 10**

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

The \_\_getattr\_\_ and \_\_getattribute\_\_ are two special methods in Python classes that are used for attribute access. However, there are important differences between them:

\_\_getattr\_\_(self, name): This method is called when an attribute lookup fails in the usual way. It is only invoked if the requested attribute is not found through the normal lookup process. The \_\_getattr\_\_ method is a fallback mechanism to dynamically handle attribute access that is not explicitly defined in the class.

\_\_getattribute\_\_(self, name): This method is called for every attribute access, regardless of whether the attribute exists or not. It is invoked first during attribute access, even before checking if the attribute exists. The \_\_getattribute\_\_ method allows you to intercept all attribute access and customize the behavior.

The \_\_getattribute\_\_ method takes two parameters: self (the instance of the class) and name (the name of the attribute being accessed).

It should return the value of the attribute or raise an AttributeError if the attribute cannot be accessed.

Q2. What is the difference between properties and descriptors?

properties provide a convenient way to define attribute access and assignment methods directly in the class, while descriptors offer a lower-level mechanism for creating reusable attribute access and assignment logic by defining separate descriptor classes. Properties are primarily used to provide a simple interface for accessing and modifying attributes, while descriptors are more flexible and can be shared among multiple instances or classes.

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

\_\_getattr\_\_ and \_\_getattribute\_\_ differ in when they are called and their purpose in handling attribute access. Properties provide a high-level interface for defining attribute access and assignment methods directly in the class, while descriptors offer a lower-level mechanism for creating reusable attribute access and assignment logic through separate descriptor classes. Both properties and descriptors provide customization and control over attribute access and assignment but differ in their implementation and usage patterns.