**Python Advanced Assignment 10**

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

Ans-) The \_\_getattr\_\_ method is called by Python when an attribute is accessed and does not exist on the object. On the other hand, the \_\_getattribute\_\_ method is called every time an attribute is accessed on an object, regardless of whether it exists or not. \_\_getattribute\_\_ is a lower-level method and should be used with caution as it can lead to infinite recursion if not implemented carefully.

Q2. What is the difference between properties and descriptors?

Ans-) Properties and descriptors are both used to define computed attributes in Python. Properties are a simpler mechanism for creating read-only computed attributes. They are defined using the @property decorator and allow for the use of the dot notation to access the computed attribute. Descriptors are more powerful and allow for more fine-grained control over attribute access. They are defined by creating a class with one or more of the methods \_\_get\_\_, \_\_set\_\_, or \_\_delete\_\_.

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

Ans-) The key difference in functionality between \_\_getattr\_\_ and \_\_getattribute\_\_ is when they are called. \_\_getattr\_\_ is called only when an attribute is not found in the usual way, whereas \_\_getattribute\_\_ is called every time an attribute is accessed. Properties are a simpler way to define computed attributes, and only support read-only access. Descriptors are more powerful and allow for more control over attribute access, supporting read, write, and delete access. Descriptors can also be used to define attributes that are shared by all instances of a class, while properties cannot.