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

|  |  |
| --- | --- |
| **\_\_getattr\_\_** | **\_\_getattribute\_\_** |
| \_\_getattr\_\_ gets called if there is no attribute in the instance. | \_\_getattribute\_\_ gets called all the times, whether there is the attribute or not. |
| It’s invoked “last”, if Python can’t find that attribute.(lowest priority) | It’s invoked “first”(highest priority) — it actually “intercepts” every lookup.  So, even if there is the attribute in the instance, Python calls \_\_getattribute\_\_ first, with the attribute as an argument. |
|  |  |

Q2. What is the difference between properties and descriptors?

**Properties:** With Properties we can bind getter, setter and delete functions together with an attribute name, using the built-in property function or @property decorator. When we do this, each reference to an attribute looks like simple, direct access, but involes the appropriate function of the object.

**Descriptor:** With Descriptor we can bind getter, setter and delete functions into a seperate class. we then assign an object of this class to the attribute name in our main class. When we do this, each reference to an attribute looks like simple, direct access but invokes an appropriate function of descriptor object.

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

The Key Differences between \_\_getattr\_\_, \_\_getattribute\_\_, Properties and Descriptors are:

**\_\_getattr\_\_**: Python will call this method whenever you request an attribute that hasn't already been defined

**\_\_getattribute\_\_** : This method will invoked before looking at the actual attributes on the object. Means,if we have \_\_getattribute\_\_ method in our class, python invokes this method for every attribute regardless whether it exists or not.

**Properties:** With Properties we can bind getter, setter and delete functions together with an attribute name, using the built-in property function or @property decorator. When we do this, each reference to an attribute looks like simple, direct access, but involes the appropriate function of the object.

**Descriptor:** With Descriptor we can bind getter, setter and delete functions into a seperate class. we then assign an object of this class to the attribute name in our main class. When we do this, each reference to an attribute looks like simple, direct access but invokes an appropriate function of descriptor object.