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

A key difference between \_\_getattr\_\_ and \_\_getattribute\_\_ is that \_\_getattr\_\_ is only invoked if the attribute wasn't found the usual ways. It's good for implementing a fallback for missing attributes.

\_\_getattribute\_\_ is invoked before looking at the actual attributes on the object, and so can be tricky to implement correctly. You can end up in infinite recursions very easily.

New-style classes derive from object, old-style classes are those in Python 2.x with no explicit base class. But the distinction between old-style and new-style classes is not the important one when choosing between \_\_getattr\_\_ and \_\_getattribute\_\_.

Q2. What is the difference between properties and descriptors?

descriptors are a low-level mechanism that lets you hook into an object's attributes being accessed. Properties are a high-level application of this; that is, properties are implemented using descriptors. Or, better yet, properties are descriptors that are already provided for you in the standard library.

There are several ways that we can tap into Python's internal mechanisms for getting and setting attribute values. The most accessible technique is to use the property function to define get, set and delete methods associated with an attribute name. The property function builds descriptors for you. A slightly less accessible, but more extensible and reusable technique is to define descriptor classes yourself. This allows you considerable flexibility. You do this by creating a class which defines get, set and delete methods, and you associate your descriptor class with an attribute name.

The property function gives us a handy way to implement a simple descriptor without defining a separate class. Rather than create a complete class definition, we can write getter and setter method functions, and then bind these functions to an attribute name.

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

* that \_\_getattr\_\_ is only invoked if the attribute wasn't found the usual ways
* \_\_getattribute\_\_ is invoked before looking at the actual attributes on the object
* Descriptors are invoked by the \_\_getattribute\_\_() method.
* Overriding \_\_getattribute\_\_() prevents automatic descriptor calls.
* object.\_\_getattribute\_\_() and type.\_\_getattribute\_\_() make different calls to \_\_get\_\_().
* Data descriptors always override instance dictionaries.
* Non-data descriptors may be overridden by instance dictionaries.
* [Properties](https://en.wikipedia.org/wiki/Property_(programming)) represent an intermediate functionality between a plain attribute (or field) and a method. In other words, they allow you to create methods that behave like attributes. With properties, you can change how you compute the target attribute whenever you need to do so