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 was not found the usual ways. __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.

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 i.e properties are implemented using descriptors.

Q3. What are the key differences in functionality between __getattr__ and __getattribute__, as well as properties and descriptors?

- A key difference between __getattr__ and __getattribute__ is that __getattr__ is only invoked if the attribute was not found the usual ways. It's good for implementing a fallback for missing attributes and is probably the one of two you want.
- __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__.
- 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 i.e properties are implemented using descriptors.