Q1. The main difference between `__getattr__` and `__getattribute__` is in their behavior and when they are invoked:
- `__getattr__` is invoked only when the requested attribute is not found through normal lookup in the instance and its class hierarchy. It is a fallback method that is called as a last resort when an attribute is not found.
- `__getattribute__` is called for every attribute access, regardless of whether the attribute exists or not. It is invoked before checking if the attribute exists, allowing you to intercept all attribute accesses and perform custom actions.

Q2. Properties and descriptors are both mechanisms in Python for defining and controlling attribute access, but they have some differences:
- Properties: Properties are a high-level way of defining computed attributes. They allow you to define methods that are accessed like attributes, providing custom behavior for getting, setting, and deleting values. Properties are defined using the `@property` decorator and associated getter, setter, and deleter methods.
- Descriptors: Descriptors are a lower-level mechanism for attribute access. They provide a way to define how attribute access is handled at the class level, allowing you to customize the behavior of getting, setting, and deleting attributes. Descriptors are implemented by defining special methods like `__get__`, `__set__`, and `__delete__`.

Q3. The key differences in functionality between `__getattr__` and `__getattribute__`, as well as properties and descriptors, are as follows:
- `__getattr__` vs. `__getattribute__`:
  - `__getattr__` is only invoked when the attribute is not found through normal lookup, providing a fallback mechanism for attribute access.
  - `__getattribute__` is called for every attribute access, allowing you to intercept and customize all attribute accesses, even for existing attributes.
- Properties vs. Descriptors:
  - Properties are used to define computed attributes that behave like normal attributes, providing custom behavior for getting, setting, and deleting values.
  - Descriptors provide a lower-level mechanism for attribute access at the class level, allowing you to customize the behavior of attribute access for all instances of a class.
  - Descriptors can be more powerful and flexible than properties but require more explicit implementation and can be more complex to use.

Overall, `__getattr__` and properties offer more convenience for defining computed attributes and customizing attribute access, while `__getattribute__` and descriptors provide more control and flexibility at the expense of increased complexity.