Q1. What are the two latest user-defined exception constraints in Python 3.X?

- raise statement - This is used to raise an inbuilt exception explicitly.
- assert statement - when condition in assert statement is true, program is executed normally. When condition is false, program gives an AssertionError

Q2. How are class-based exceptions that have been raised matched to handlers?

- When exceptions are raised, then the exception is handled by the same class mentioned in except statement or its base classes. "Exception" is the base class(Superclass) for all other exception classes

Q3. Describe two methods for attaching context information to exception artefacts.

- GetHibernateTemplate().update(obj) This works if and only if an object doesn't already exist in the hibernate session. Exceptions are thrown stating an object with the given identifier already exists in the session when I need it later.
-GetHibernateTemplate().merge(obj) This works if and only if an object exists in the hibernate session. Exceptions are thrown when I need the object to be in a session later if I use this.

Q4. Describe two methods for specifying the text of an exception object's error message.

- Errors happen all the time in the software world. It might be an invalid user input or an external system that is not responding, or it is a simple programming error. In all these situations, the errors occur at runtime and the application needs to handle them. Otherwise, it crashes and cannot process further requests. Java provides a powerful mechanism which allows you to handle the exceptional event where it occurred or in one of the higher methods in the call stack.

Q5. Why do you no longer use string-based exceptions?

- In Python versions 1.5 and earlier, Exceptions were strings and not classes. Now all exceptions are classes, derived from the superclass BaseException. With classes it is more efficiently handled as classes have many useful features.