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

\_\_init\_\_,\_\_str\_\_ are the two latest user-defined exception constraints in python3.X

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

The exceptions that can be handled are class-based and can be handled between TRY and catch. The associated exception classes are predefined in the system and begin with the prefix CX\_SY\_, such as CX\_SY\_ZERODIVIDE. In the ABAP keyword documentation, the exception classes whose exceptions may occur when a corresponding ABAP statement is executed are listed for each keyword.

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.

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

User-defined **Python exceptions can** be either **strings** or **Python** classes. ... Because it **is** possible (although unlikely) that this change broke existing code, the **Python** interpreter **can** be invoked the command line option -X to disable this feature, and **use string exceptions** like before.