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

**Answer:** SecondaryException() from Primary\_Exception & SystemExit or KeyboardInterrupt.

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

**Answer:** Class-based exceptions are either raised by the ABAP statement RAISE EXCEPTION or by the ABAP runtime environment. If a class-based exception occurs, the system interrupts the normal program flow and tries to find a suitable handler. If it does not find a handler, a runtime error occurs.

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

**Answer:** \_\_suppress\_context\_\_ & \_context\_ are used for attaching context information to exception artefacts.

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

**Answer:** str(ex) gives us the error message. You could use repr(ex) to get the full text, including the name of the exception raised.

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

**Answer:** The obsolete AccessError exception has been deleted. 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. This option is a temporary measure - eventually the string-based standard exceptions will be removed from the language altogether.