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

A1. The two latest user-defined exception constraints in Python 3.x are the "exception chaining" constraint and the "suppress context" constraint. Exception chaining allows one exception to reference another exception that caused it, while the suppress context constraint allows an exception handler to selectively suppress parts of the error message or traceback.

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

A2. When a class-based exception is raised, Python searches for the most specific handler that matches the exception type. The search starts with the current function or method, then moves up the call stack until a matching handler is found. If no matching handler is found, the exception propagates up to the top level of the program and causes the program to terminate.

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

A3. Two methods for attaching context information to exception artifacts are the "args" attribute and the "with\_traceback" method. The "args" attribute is a tuple of arguments that were passed to the exception when it was raised. These can include additional information that might help diagnose the cause of the exception. The "with\_traceback" method allows us to associate a traceback object with the exception, which can provide additional information about the context in which the exception occurred.

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

A4. Two methods for specifying the text of an exception object's error message are to pass a string message to the exception constructor or to define a str or repr method in the exception class. If a string message is passed to the constructor, it will be stored in the args attribute of the exception object. If a str or repr method is defined, it will be called when the exception object is printed or converted to a string, respectively.

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

A5. String-based exceptions have been deprecated in favor of class-based exceptions because they are less flexible and less expressive. With class-based exceptions, we can define custom exception types that can carry additional data or context information, making it easier to diagnose and fix errors. Class-based exceptions also allow us to create more specific handlers that can handle only certain types of exceptions, rather than relying on string matching to determine the appropriate handler.