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

Ans1- The two latest user-defined exception constraints introduced in Python 3.X are the requirement that custom exception classes must inherit from the built-in BaseException class, and they should generally include a docstring explaining the exception.

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

Ans2- Class-based exceptions are matched to handlers using the inheritance hierarchy. Python searches for an appropriate exception handler from the innermost try block to the outermost, looking for a class that matches the exception or one of its base classes.

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

Ans3- Two methods for attaching context information to exception artifacts are:

Using the raise ... from ... syntax to create chained exceptions, preserving the context of the original exception.

Adding custom attributes to the exception instance to store additional context information.

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

Ans4- Two methods for specifying the text of an exception object's error message are:

Providing a string argument when raising the exception, which becomes the error message.

Defining a custom \_\_str\_\_ method in the exception class to generate a custom error message.

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

Ans5- String-based exceptions are no longer used because they lack the rich features and extensibility of class-based exceptions. Class-based exceptions allow for better organization, customization, and handling of exceptions and provide more information about the exception, including its type, attributes, and traceback information.