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

Answer 1: The two common constraints for user-defined exceptions are:

- Inheriting from the built-in Exception class.

- Providing a docstring for the custom exception class.

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

Answer 2: Class-based exceptions that have been raised are matched to handlers based on the class hierarchy. The interpreter looks for the most specific exception type first and then moves up the hierarchy until it finds a matching exception handler.

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

Answer 3: Two methods for attaching context information to exception artifacts are:

- Adding custom attributes to the exception instance.

- Utilizing the \_\_context\_\_ attribute to link to another exception.

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

Answer 4: Two methods for specifying the text of an exception object's error message are:

- Defining a \_\_str\_\_ method in the exception class to customize the string representation.

- Passing a custom error message as an argument when raising the exception.

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

Answer 5: String-based exceptions are no longer recommended due to several reasons, including:

- Lack of consistency in error handling.

- Difficulty in programmatic handling since string matching is error-prone.

- Limited ability to attach additional information to exceptions.

- Class-based exceptions provide a more structured and reliable way to handle errors in Python.