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

As of my knowledge cutoff in September 2021, there were no predefined or built-in user-defined exception constraints in Python. Python allows you to create your own custom exceptions by defining new classes that inherit from the built-in Exception class or any of its subclasses.

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

In Python, when a class-based exception is raised, the process of matching the exception to an appropriate exception handler follows a hierarchical order. The exception hierarchy is based on the inheritance structure of the exception classes.

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

When dealing with exceptions in Python, it is often helpful to attach additional context information to the exception artifacts to provide more detailed information about the cause or context of the error. Here are two common methods for attaching context information to exception artifacts:

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

When raising an exception in Python, you can specify the text of the exception object's error message to provide meaningful information about the error. Here are two common methods for specifying the text of an exception object's error message:

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

The use of string-based exceptions in Python is generally discouraged and considered a deprecated practice. Instead, it is recommended to use exception classes that inherit from the built-in Exception class or its subclasses. Here are a few reasons why string-based exceptions are no longer preferred: