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

Exception Chaining: In Python 3.x, you can use the from keyword to chain exceptions, allowing you to associate an original exception with a new exception. This helps in preserving the original exception information while raising a new exception.

Exception Context: Python 3.x introduced the \_\_context\_\_ attribute for exceptions, which allows access to the original exception context. The \_\_context\_\_ attribute refers to the exception that caused the current exception to be raised

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

class-based exceptions that have been raised are matched to handlers based on their inheritance hierarchy. When an exception is raised, Python traverses the inheritance chain of the exception class to find the first matching exception handler.

The raised exception is checked against each except clause in the try-except statement, from top to bottom, in the order they are defined.

Python compares the raised exception with the exception types specified in each except clause. If the raised exception is an instance of or a subclass of any of the specified exception types, the corresponding except clause is considered a match

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

Adding Custom Attributes: You can attach additional context information to an exception by adding custom attributes to the exception object. This allows you to provide more specific details about the exception or the circumstances in which it occurred.

Using Exception Arguments: Python exceptions allow you to pass arguments during exception instantiation, which can be used to provide context information. You can pass these arguments when raising an exception, and they can be accessed through the args attribute of the exception obje

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

Using the Exception Class Constructor: You can specify the error message of an exception object by providing a string argument to the constructor of the exception class. This allows you to customize the error message according to the specific situation

Formatting Exception Messages: Another method is to use string formatting techniques to dynamically construct the error message at the point of raising the exception. You can use formatting placeholders and pass the necessary values to generate a more informative error message

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

In Python, it is generally recommended to avoid using string-based exceptions and instead use class-based exceptions. Here are a few reasons why string-based exceptions are no longer commonly used.

Lack of Specificity

Limited Functionality

Better Exception Handling

Code Readability and Maintainability