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

Two Latest User-Defined Exception Constraints in Python 3.X:

In Python 3.X, the two latest user-defined exception constraints are:

Exception Chaining: It allows one exception to be raised on top of another, preserving the original traceback. It helps in providing more detailed information about the context of the exception.

Exception Context: It allows attaching additional context information to exceptions using the \_\_context\_\_ attribute.

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

Class-based exceptions are matched to handlers based on their inheritance hierarchy. When an exception is raised, Python searches for the first matching except block that can handle that specific exception type or any of its base classes.

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

Using the raise ... from ... Syntax: This syntax allows raising an exception with additional context, linking it to the context of another exception.

try:

# Some code

except SomeException as e:

raise CustomException("Additional information") from e

Attaching Context Information Manually: You can manually attach context information by setting attributes in the exception object.

try:

# Some code

except SomeException as e:

e.context\_info = "Additional information"

raise

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

Override \_\_str\_\_ Method: You can override the \_\_str\_\_ method in your custom exception class to provide a custom error message.

Passing Message as an Argument: When raising an exception, you can pass the error message as an argument.

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

String-based exceptions were used in older versions of Python, but they lacked the structure and extensibility of class-based exceptions.

Class-based exceptions provide a more organized and object-oriented approach, allowing developers to define custom exception hierarchies with meaningful attributes and methods.

Class-based exceptions also support inheritance, making it easier to catch and handle different types of exceptions in a more structured manner.