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

Ans:1 **The Exception class should be derived from the built-in Exception class or one of its subclasses.**

**2 The Exception class should provide a useful error message.**

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

Ans:

1. The interpreter looks for an appropriate except clause in the current try block. If the raised exception matches the exception type specified in the except clause, the corresponding block of code is executed.
2. If no matching except clause is found in the current try block, the interpreter checks the surrounding try statements (if any) in the order they were encountered during the execution.
3. If the interpreter does not find any matching except clause in the current try block or its surrounding try blocks, the exception is considered unhandled, and the program terminates, displaying the default error message and stack trace.

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

Ans:

**Using Exception Arguments:** One of the simplest ways to attach context information to exceptions is by passing relevant information as arguments when raising an exception. You can include additional information in the exception message to provide more context about the error.

**Creating Custom Exception Classes:** Another method is to create custom exception classes that contain attributes for storing specific context information related to the exception. By defining custom attributes within the exception class, you can attach detailed information that is accessible when handling the exception.

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

Ans:

**Passing the Message as an Argument:** One approach is to pass the error message as an argument when creating or raising the exception. This can be achieved by providing a string as an argument to the exception constructor.

**Customizing the Exception Class:** Another method is to customize the exception class by overriding the \_\_init\_\_ method and setting the error message as an attribute. By defining a custom exception class, you can have more control over the error message format and content.

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

Ans: Using string-based exceptions in Python is discouraged because they lack specificity, limited functionality, reduce code maintainability, and compromise readability and clarity. Class-based exceptions offer better organization, detailed information, and improved maintainability, making them the preferred approach in modern Python programming.