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

***Ans:*** Raise and Assert are the two latest user-defined exception constraints in Python 3.X.

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

***Ans***: because they inherit from in-built Exception class.

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

***Ans***:

1. We can use user defined exceptions to add the context of the exception

2. logging.error() can also be used to add the information about the exception

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

***Ans:***

1.Using the raise statement with a string argument:

When raising an exception with the raise statement, you can pass a string argument that represents the error message.

2. Defining a custom exception class with a \_\_str\_\_ method:

You can also define a custom exception class and override its \_\_str\_\_ method to provide a custom error message.

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

***Ans:***

1.Limited information: A string-based exception provides only limited information about the error, making it harder to debug and fix issues.

2.No way to catch specific exceptions: Since all string-based exceptions are essentially the same type of object (a string), it is impossible to catch specific exceptions or distinguish between different types of errors.

3.Poor readability: When reading code that raises a string-based exception, it can be unclear what type of exception is being raised or what the error message means.