Q1. Describe three applications for exception processing.

Exception processing is applied for error handling, validating user input, and managing resources. It allows efficient handling of errors, guiding users for valid input, and ensuring proper cleanup of resources like files or resources to make program responsible.

Q2. What happens if you don't do something extra to treat an exception?

The program will terminate abruptly and display an error message or traceback. This can result in an undesirable user experience and prevent proper execution of subsequent code.

Q3. What are your options for recovering from an exception in your script?

Catch the exception using try-except block and execute alternative code to handle the error. It allows us to continue the script's execution without termination. Capture and log the exception details to a log file

Q4. Describe two methods for triggering exceptions in your script.

Raise an Exception: Using raise statement to create custom exceptions with specific error messages, allow us to indicate exceptional conditions or errors that need special handling in our code.

Built in exceptions: python exception class provides pre-defined exceptions for common errors like invalid operations or accessing elements beyond list bounds. These exceptions are automatically triggered when such situations occur.

Q5. Identify two methods for specifying actions to be executed at termination time, regardless of whether or not an exception exists.

Using a finally block which is placed after the try and except blocks and is executed regardless of whether an exception is raised or not. It is commonly used to perform cleanup operations, such as closing files or releasing resources, ensuring they are executed even if an exception occurs.