Q1. Describe three applications for exception processing.

Exception processing is used for error handling and control flow in Python applications. Three applications include: graceful program termination (closing files or resources), user-friendly error messages (catching exceptions and displaying helpful information), and program robustness (handling unexpected input or external conditions).

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

If an exception is not handled, it will propagate up the call stack until it reaches the global scope, potentially causing the program to terminate abruptly. An unhandled exception will display an error message and a traceback, revealing the line of code where the exception occurred.

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

Options for recovering from an exception include using a try and except block to catch and handle exceptions, allowing the program to continue execution. Alternatively, you can log the exception details and gracefully exit the program, or prompt the user for corrected input.

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

Two methods for triggering exceptions are: raising an exception explicitly using the raise statement with an exception type, and using built-in functions like assert to raise an exception if a certain condition is not met.

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

Two methods for specifying actions to be executed at termination time are: using a try and finally block, where the finally block contains code that will be executed regardless of whether an exception occurred or not; and utilizing the atexit module to register functions that will be called when the program exits, regardless of exceptions.