Q1. Describe three applications for exception processing.

A1. Exception processing can be used for error handling, program control flow, and program debugging. With exception handling, we can gracefully handle unexpected errors and provide a fallback for continuing program execution. We can also use exceptions to change the normal flow of control in our program. Finally, exceptions can be used for debugging purposes, as they can provide detailed information about the location and nature of errors that occur during program execution.

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

A2. If an exception is not handled, it will propagate up the call stack until it is either caught by a handler or the program terminates. If an unhandled exception reaches the top level of the program, it will cause the program to terminate with an error message.

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

A3. When an exception occurs, there are several ways to recover from it. One option is to catch the exception and handle it, either by fixing the error that caused the exception or by taking some other action to recover from the error. Another option is to raise a different exception to indicate that an error has occurred. Finally, it is sometimes possible to simply ignore the exception and continue program execution, although this should be done with caution.

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

A4. Two common ways to trigger exceptions in a script are to use the "raise" statement and to call functions that can raise exceptions. The "raise" statement is used to raise a specific exception manually. For example, we might raise an exception if a function is called with invalid arguments. Functions that can raise exceptions include built-in functions like "open" (which can raise a "FileNotFoundError" exception if the specified file cannot be found) and user-defined functions that contain "raise" statements.

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

A5. Two methods for specifying actions to be executed at termination time are the "finally" clause and the "atexit" module. The "finally" clause is used to specify code that should be executed regardless of whether or not an exception was raised. This is useful for cleaning up resources or releasing locks that were acquired during program execution. The "atexit" module provides a way to register functions that should be called when the program terminates, either normally or due to an unhandled exception. This is useful for performing tasks such as closing open files or saving program state.