Q1. Three applications for exception processing are:

- Error handling: Exceptions allow you to handle errors and unexpected events that may occur during the execution of your program. You can use exceptions to gracefully handle errors and provide appropriate error messages to users.

- Program flow control: Exceptions can be used to change the flow of a program based on certain conditions. For example, if a file cannot be opened, you can raise an exception and execute an alternative code path.

- Resource management: Exceptions can be used to manage resources such as files and network connections. For example, if a file cannot be closed, you can catch the exception and ensure that the file is closed before the program terminates.

Q2. If you don't handle an exception, it will propagate up the call stack until it is caught by an exception handler or until it reaches the top-level of the program, where it will cause the program to terminate and display an error message.

Q3. Some options for recovering from an exception in your script are:

- Retry the operation that raised the exception: You can use a loop to retry the operation that raised the exception until it succeeds.

- Log the error and continue execution: You can catch the exception, log an error message, and continue executing the rest of the code.

- Raise a different exception: You can catch the exception and raise a different exception to indicate that something went wrong.

Q4. Two methods for triggering exceptions in your script are:

- raise statement: You can use the raise statement to explicitly raise an exception. You can raise built-in exceptions or create your own custom exceptions.

- assert statement: You can use the assert statement to raise an AssertionError if a condition is not true. This is useful for debugging and testing your code.

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

- finally block: You can use the finally block to specify code that must be executed regardless of whether or not an exception was raised. This is useful for releasing resources such as files and network connections.

- atexit module: You can use the atexit module to register functions that will be called when the program exits normally. This is useful for performing cleanup tasks such as removing temporary files.