Q1. Describe three applications for exception processing.

1. Error handling: Exception processing is commonly used in error handling to catch and handle errors that may occur during the execution of a program. This ensures that the program does not crash and provides useful information about the error to the user.
2. Input validation: Exception processing can be used to validate user input and handle invalid input. For example, if a user enters a non-numeric value where a number is expected, an exception can be raised and handled to inform the user of the error.
3. Resource management: Exception processing can be used to ensure proper resource management. For example, if a file is opened but an exception is raised before it is closed, the exception can be handled to ensure the file is closed properly before the program terminates.

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

If an exception is not handled in some way, it will propagate up the call stack until it reaches the top-level of the program. If the exception reaches the top-level without being handled, the program will terminate abruptly and any resources that were not cleaned up will not be released.

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

The options for recovering from an exception in a script depend on the programming language being used and the type of exception that occurred. Generally, the options include:

1. Ignoring the exception and continuing execution of the script.
2. Handling the exception by executing code that can recover from the error.
3. Raising a new exception to indicate that the error has been handled.
4. Exiting the script.

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

1. Explicitly raising an exception: In most programming languages, you can explicitly raise an exception by calling a function or method that raises an exception. For example, in Python, the `raise` statement can be used to raise an exception.
2. Letting the language/runtime raise an exception: Exceptions can also be raised by the programming language or runtime when certain conditions are met. For example, in Python, a `ZeroDivisionError` will be raised if you try to divide by zero.

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

1. Finally block: Most programming languages provide a `finally` block that is executed regardless of whether or not an exception occurred. This can be used to ensure that resources are properly cleaned up before the program terminates.
2. Exit handlers: Some programming languages provide the ability to register exit handlers that will be executed when the program terminates, regardless of whether

or not an exception occurred. These handlers can be used to perform cleanup or logging operations before the program terminates.