Q1. Describe three applications for exception processing.

Exception processing is a crucial aspect of programming that allows for the handling of exceptional situations and errors that may occur during the execution of a program. Here are three applications for exception processing

Error Handling:

Input Validation

Resource Management

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

If an exception occurs in a program and is not properly handled, it will propagate up the call stack until it reaches the top level of the program or an exception handler is encountered. If the exception is not caught and handled, the program will terminate abruptly, and an error message or traceback will be displayed

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

When an exception occurs in a script, you have several options for recovering from it and continuing the execution of your program. Here are some common approaches.

Exception Handling with Try-Except.

Retry Mechanism: In some cases, it might be appropriate to retry the operation that raised the exception.

Graceful Termination: Depending on the severity of the exception and the nature of the error, it may be appropriate to gracefully terminate the program

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

In Python, there are multiple ways to trigger exceptions in your script. Here are two common methods

Raise an Exception Manually: You can manually raise an exception at any point in your script using the raise statement. By specifying the type of exception to raise, you can explicitly trigger an exception and control the flow of your program.

Let Built-in Operations Raise Exceptions: Certain built-in operations or functions in Python automatically raise exceptions when specific conditions are not met.

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

by using two methods.

Finally Block: The finally block is a construct that allows you to define a set of statements that will be executed regardless of whether an exception occurs or not. It is typically used in conjunction with a try-except block to ensure certain actions are performed, such as cleanup operations or resource release, regardless of the outcome.

Context Managers: Context managers provide a convenient way to specify actions that need to be executed at the beginning and end of a block of code, regardless of whether an exception occurs or not. The with statement is used to define a context manager. It ensures that the necessary setup and cleanup operations are performed.