Q1. Describe three applications for exception processing.

1. Preventing program crashes:

Exception processing can help to prevent your program from crashing when an error occurs. This is because it allows you to handle the error and take corrective action, rather than letting the error cause your program to crash.

2. Improving program reliability:

Exception processing can help to improve the reliability of your program. This is because it allows you to handle errors and continue running, even if an error occurs. This can help to ensure that your program is always available to your users.

3. Making program more user-friendly:

Exception processing can help to make your program more user-friendly. This is because it allows you to handle errors and provide meaningful error messages to your users. This can help your users to understand what went wrong and how to fix the problem.

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

Errors and exceptions can lead to program failure or unexpected behaviour.

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

When a Python code throws an exception, it has two options: handle the exception immediately create log file to store an error for troubleshoot or stop and quit.

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

There are two ways to trigger exceptions in Python:

1. Using the raise statement: The raise statement is used to raise an exception explicitly.

For example:

raise ValueError("This is an example of raising an exception")

1. Using the try/except block: The try/except block is used to handle exceptions that may occur during the execution of a code block.

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

There are two methods for specifying actions to be executed at termination time, regardless of whether or not an exception exists in Python:

**The try/except/finally block:**

The try/except/finally block is a construct in Python that allows you to handle exceptions that occur during the execution of your code. The try block is the block of code that you want to try to execute. The except block is the block of code that you want to execute if an exception occurs in the try block. The finally block is the block of code that you want to execute regardless of whether or not an exception occurs in the try block.

**The raise statement:**

The raise statement is a statement in Python that allows you to raise an exception. An exception is an object that represents an error that has occurred during the execution of your code. The raise statement can be used to raise an exception manually, or it can be used to raise an exception that is caused by an error in your code.