Q1. Describe three applications for exception processing.

* Exception processing is used to handle exceptional situations in a program. Three applications include:
* Error Handling: Exceptions are used to gracefully handle errors and prevent program crashes. For example, you can catch file-not-found errors when reading files.
* Validation: Exceptions can be used to validate input or conditions and provide appropriate feedback. For instance, you can raise exceptions when user inputs are invalid.
* Resource Cleanup: Exceptions can help ensure that resources like files or network connections are properly closed and released, even if an error occurs during processing.

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

If an exception is not explicitly handled using try-except blocks, the program will terminate abruptly and display an error message. This can lead to incomplete operations and a poor user experience. Exception handling allows you to manage errors more gracefully and continue execution or provide informative feedback.

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

When recovering from an exception, you have several options:

* Logging: Log the error details for later analysis.
* Retry Mechanism: Retry the failed operation after a delay to see if the issue resolves.
* Fallback Values: Provide default or fallback values if data retrieval fails.
* User Feedback: Notify users about the error and guide them on what to do next.
* Alternate Paths: Redirect execution to an alternative path if a critical operation fails.

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

Using raise Statement: You can manually raise exceptions using the raise statement followed by the exception class and an optional message. For example:

raise ValueError("This is a custom exception.")