Q1. Describe three applications for exception processing.

Ans: -
Error Handling: Exception handling in Python is a mechanism for gracefully responding to runtime errors or exceptional cases in a program. It allows the program to continue with the next instructions or to take necessary corrective measures to handle the situation.
Debugging: Exceptions can provide detailed insights into what went wrong during the execution of a program. The traceback information can help to identify the section of the code where the error occurred.
Control Flow: In some cases, exceptions are used for controlling the flow of execution in a program. They can be used to break out of multiple nested loops or to skip certain sections of the code when a specific condition is met.

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

Ans: - If an exception is not handled (i.e., caught within a try/except block), it propagates up to the top level of execution, causing the program to terminate and print a traceback to the stderr.

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

Ans: -
Catch and Handle the Exception: You can use a try/except block to catch and handle the exception. The try block contains the code that might raise an exception, and the except block contains the code that handles the exception.
Retry the Operation: In some cases, you might want to retry the operation that caused the exception. This is common in network operations where temporary issues can cause exceptions.
Log the Error and Continue: If the operation that caused the exception is not critical to the overall function of your script, you might choose to log the error and continue with the rest of the script.

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

Ans: -
The raise Statement: You can use the raise statement to throw an exception in Python. The argument to raise indicates the exception to be raised and must be an instance of the BaseException class or a class derived from it.
Assertion: You can also use the assert statement to raise an exception. The assert statement throws an AssertionError exception if a specified condition is not True.

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

Ans: -

The finally Clause: In Python, the finally clause in a try/except block will always be executed before leaving the try statement, whether an exception has occurred or not.

The atexit Module: The atexit module defines a single function to register cleanup functions. Functions thus registered are automatically executed upon normal interpreter termination.