Q1. Describe three applications for exception processing.

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

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

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

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

Answer:

Q1. Exception processing is an important feature of Python that allows programs to handle and recover from errors that occur during execution. Some common applications of exception processing in Python include:

* Error handling: Exceptions can be used to catch and handle errors that occur during program execution, allowing the program to recover and continue running instead of crashing.
* Resource cleanup: Exceptions can be used to ensure that resources (such as files, sockets, or database connections) are properly cleaned up and released, even if an error occurs.
* Control flow: Exceptions can be used to change the flow of a program, allowing it to handle unexpected situations and take alternative paths.

Q2. If an exception is raised in your Python script and you don't handle it, the program will terminate and print a traceback that shows where the exception occurred. This can be useful for debugging and troubleshooting, but it also means that any code after the point of the exception will not be executed.

Q3. When an exception is raised in your Python script, you have several options for recovering from it, including:

* Catching and handling the exception using a try-except block: This allows you to specify a block of code that will be executed if the specified exception is raised, and can be used to handle the exception and continue running the program.
* Raising a new exception: This allows you to propagate the exception to a higher level of the program, where it can be handled or logged in a more appropriate way.
* Exiting the program: In some cases, it may be appropriate to exit the program if an unrecoverable error occurs.

Q4. There are several ways to trigger exceptions in your Python script, including:

* Using the raise statement: This allows you to raise a specific exception, with an optional message and traceback, at any point in your code.
* Calling a function or method that raises an exception: Many built-in functions and methods in Python can raise exceptions if they encounter an error, such as trying to open a file that doesn't exist.

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

* Using a finally block: This allows you to specify a block of code that will be executed whether or not an exception is raised. This can be useful for releasing resources or cleaning up after a block of code.
* Using the atexit module: This module provides a way to register functions that will be called when the Python interpreter exits, regardless of whether or not an exception occurred. This can be useful for performing final cleanup or logging.