Q1. Describe three applications for exception processing.

Ans: Exception processing is an important concept in software development that involves handling unexpected or abnormal situations that can occur during the execution of a program. Here are three common applications for exception processing:

1. Error handling

2. Resource management

3. Workflow management

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

Ans: If an exception occurs in a program and is not properly handled, the program will typically terminate abruptly, producing an error message or crash report. This can be disruptive and frustrating for users, and can also cause data loss or corruption if the program was in the middle of performing a critical operation. Additionally, if the program is being used in a production environment, unhandled exceptions can potentially cause downtime and financial loss for the organization.

In some cases, unhandled exceptions may also leave the program in an inconsistent state, where resources such as files, database connections, or memory have not been properly released. This can lead to memory leaks, file corruption, or other issues that can cause the program to behave unpredictably or even crash in subsequent runs.

Properly handling exceptions is an important part of writing robust and reliable software. By anticipating potential errors and implementing exception handling routines, developers can ensure that their programs are able to gracefully handle unexpected situations and continue to operate correctly even in the face of errors or other problems.

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

Ans: When an exception occurs in a script, there are several options for recovering from it. Here are some common techniques:

a. Catching and handling the exception: The most common way to recover from an exception is to catch it and handle it in the code. This involves using a try-except block to catch the exception and then taking appropriate action, such as logging the error, displaying an error message to the user, or retrying the operation that caused the exception.

b. Raising a different exception: In some cases, it may be appropriate to raise a different exception in response to the original exception. For example, if a file cannot be opened due to a permissions issue, the script could raise a new exception indicating that the user does not have sufficient privileges to access the file.

c. Graceful degradation: In some cases, it may be possible to recover from an exception by gracefully degrading the functionality of the script. This involves detecting the exception and then disabling or reducing the functionality that is affected by the exception. For example, if a script that downloads data from the internet encounters a network error, it could continue to operate by using cached data or reducing the number of downloads it attempts.

d. Retry: Another option for recovering from an exception is to retry the operation that caused the exception. This can be useful for transient errors, such as network timeouts or database connectivity issues. The script can be configured to attempt the operation again after a certain amount of time or after a user-specified number of attempts.

e. Abort the script: In some cases, it may be appropriate to abort the script if an exception occurs. This can be useful if the script is unable to operate correctly without the functionality that caused the exception, or if the exception indicates a critical error that cannot be recovered from.

The choice of recovery technique will depend on the nature of the exception and the requirements of the script.

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

Ans: In Python, there are several ways to trigger exceptions in a script. Here are two common methods:

a. Using the raise statement: The raise statement is used to explicitly raise an exception in a script. This can be useful when the script encounters an error that cannot be handled, or when the script needs to signal an error condition to another part of the program. To raise an exception, simply use the raise keyword followed by the type of exception you want to raise. For example, the following code raises a ValueError exception:

if x < 0:

raise ValueError("x cannot be negative")

b. Calling functions or methods that raise exceptions: Many built-in Python functions and methods can raise exceptions if they encounter errors or unexpected conditions. For example, the open() function can raise a FileNotFoundError exception if the specified file does not exist. Similarly, the append() method of a list object can raise a TypeError exception if an object of the wrong type is appended to the list. To trigger an exception in this way, simply call the function or method that can raise an exception and pass in any required arguments. For example:

my\_list = [1, 2, 3]

my\_list.append("four") # Raises a TypeError exception

By understanding the different types of exceptions that can be raised in Python and how they can be triggered, developers can design more robust and reliable scripts that are able to handle errors and unexpected conditions more effectively.

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

Ans: In Python, there are two common methods for specifying actions to be executed at termination time, regardless of whether or not an exception exists:

1. Using the try-finally block: The try-finally block is used to ensure that a set of statements is executed before the script terminates, regardless of whether or not an exception occurs. The code inside the try block is executed first, and if an exception occurs, the code inside the finally block is still executed before the script terminates.

2. Using the atexit module: The atexit module provides a way to register functions to be called when the Python interpreter is about to exit. These functions are called in reverse order of registration, and they are executed regardless of whether or not an exception occurs. To register a function with the atexit module, simply use the register() function, passing in the function to be called.