**Q1. Describe three applications for exception processing.**

**Answer:-**

Exception processing has three key applications: error handling, input validation, and resource management. It enables graceful error handling, validates input, and manages resources effectively.

Exception processing is used for error handling, input validation, and resource management. It ensures graceful error handling, validates input, and manages resources effectively.

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

**Answer:-**

Without proper exception handling, unhandled exceptions can disrupt the normal flow of your program and lead to unexpected behavior or crashes. It's important to handle exceptions appropriately to ensure your program handles errors gracefully, provides feedback to users, and continues execution or terminates in a controlled manner when necessary.

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

**Answer:-**

When encountering an exception in a script, you have several options for recovering from it:

1. Exception Handling with try-except: You can use a try-except block to catch specific exceptions and handle them gracefully. By enclosing the code that may raise an exception within a try block, you can catch the exception in an except block and perform appropriate error handling or recovery actions. This allows you to continue execution even if an exception occurs.
2. Exception Propagation: If you don't catch an exception locally, it will propagate up the call stack to higher-level code. This can provide an opportunity to handle the exception at a higher level or allow it to propagate further. Ultimately, if the exception is not caught, it may result in the termination of the script and an error message.
3. Logging and Reporting: In addition to handling exceptions, you can log information about the exception to a log file or other logging mechanism. Logging helps in debugging and troubleshooting issues by capturing details about the exception, such as the error message, stack trace, and context. This information can be useful for identifying the cause of the exception and improving the script's robustness.
4. Graceful Termination: In certain cases, it may be appropriate to gracefully terminate the script if an exception occurs that cannot be handled or recovered from. This ensures that the script doesn't continue executing in an inconsistent or erroneous state.

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

**Answer:-**

1. Using the raise Statement: You can use the raise statement to explicitly raise an exception at a specific point in your code. By specifying the type of exception to raise, you can trigger it and interrupt the normal flow of execution. For example:

# Raise a ValueError exception

raise ValueError("Invalid value provided")

1. Invoking Built-in Exceptions: Python provides a range of built-in exception classes that you can directly raise to indicate specific error conditions. These exceptions are defined in the Python standard library and cover a wide range of common errors. For example:

# Raise a ZeroDivisionError exception

x = 10

y = 0

if y == 0:

raise ZeroDivisionError("Cannot divide by zero")

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

**Answer:-**

Using a finally Block: The finally block is used in conjunction with a try-except block to define a set of statements that will be executed regardless of whether an exception occurs or not. The code inside the finally block will be executed after the try block completes, even if an exception is raised. This is useful for performing cleanup operations or releasing resources that need to be executed regardless of the exception outcome. For example:

**try:**

**# Code that may raise an exception**

**except Exception:**

**# Exception handling**

**finally:**

**# Code that will be executed regardless of exception**

**# Cleanup operations or resource release**