Q1. Describe three applications for exception processing.

Answer:

Exception processing is important to find the errors and exceptions during execution of an application or program so that they can be communicated to the user application cleanly. It find exceptions that causes the runtime error while runtime errors halt the program execution when exception occurs. Following are the application of exception processing:

1) Checking appropriate use of input in an application

2) Checking for arithmetic exceptions in mathematical executions

3) Checking file I/O exceptions during file handling

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

Answer:

Exception can occur at any line of code if their requirement does not meet and it will throw an error immediately. Perhaps, if we don't do something to handle it then the program will terminates abruptly. Thus the further code is not executed.

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

Answer:

Python language provides following important features that uses to handle any unexpected error in our program, such as: try ...except ...else ...finally statement. When an exception occurs within the “try” block then, “except” block is executed and when there is no exceptions after “try” and before “finally” then, the optional “else” block is executed and the “finally” block always executed at the end, regardless of whether exceptions occurred in the try block or not.

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

Answer:

🡪 raise: this keyword triggers an exception if condition provided to it turns out to be true.

🡪 assert: this keyword generally test if a condition in a code returns true and if it is false then the program will raise an assertion error.

|  |
| --- |
| # Example of raise  x = 7  if x > 5:  raise Exception('The provided number should not exceed 5’)  # Example of assert  x = 10  assert x > 0, 'Only +ve numbers are allowed' |

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

Answer:

🡪 else: when there is no exceptions after “try” and before “finally” then, the optional “else” block is executed

🡪 finally: always executed at the end, regardless of whether exceptions occurred in the try block or not.