**Assignment - 6**

Q1. Describe three applications for exception processing.

Ans; Three applications for exception processing in Python are:

* Error Handling: Exception processing allows developers to handle errors and unexpected situations gracefully, preventing program crashes and providing informative error messages to users.
* Resource Management: Exception processing can be used to manage resources efficiently, ensuring that resources are properly released even in the event of exceptions. For example, file handles, network connections, and database connections can be safely closed in exception handlers.
* Control Flow: Exception processing can be used to alter the flow of control in a program. By raising and catching exceptions strategically, developers can implement complex logic and error recovery mechanisms.

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

Ans; If an exception occurs in Python and is not handled by a try-except block or other exception handling mechanism, the program will terminate abruptly, and an error message will be displayed to the user. This can disrupt the normal execution flow of the program and potentially lead to data loss or other undesirable consequences.

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

Ans; There are several options for recovering from an exception in a Python script:

* Handling the Exception: Use a try-except block to catch and handle the exception gracefully, allowing the program to recover from the error condition and continue executing.
* Logging: Log details about the exception, including error messages and stack traces, to a log file or console for later analysis and troubleshooting.
* Graceful Degradation: Implement fallback mechanisms or alternative workflows to gracefully degrade functionality in the event of exceptions, ensuring that the program remains usable even under adverse conditions.

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

Ans; Two methods for triggering exceptions in a Python script are:

* raise Statement: Use the raise statement to raise a specific exception explicitly at a particular point in the code. Developers can raise built-in exceptions or create custom exceptions to signal error conditions or exceptional circumstances.
* assert Statement: Use the assert statement to trigger an AssertionError if a given condition evaluates to False. Assertions are typically used for debugging and testing purposes to enforce assumptions about the state of the program.

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

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

* finally Block: Use a finally block in conjunction with a try-except block to specify cleanup code that should be executed regardless of whether an exception occurs. The code in the finally block will always be executed, even if an exception is raised or caught.
* atexit Module: Use the atexit module to register functions that should be called when the Python interpreter exits. Functions registered with atexit will be executed when the script exits normally, either by reaching the end of the script or by calling the sys.exit() function.