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

**Ans:**

real world examples:

1. you provide a web form for users to fill in and submit.but incase there are a lot of conditions to be handled and the conditions keeps changing periodically,you use exception handling to simplify the process
2. database connectivity uses exception handling(why???) this is because the reason for database connectivity failure cannot be predicted and handled as it can be caused by many variables such as power failure, unreachable server,failure at client front/back end and so on.
3. internet communication
4. arithmetic exceptions such as division by zero and so on.
5. operating systems use exception handling to resolve deadlocks,recover from crash and so forth

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

**Ans:**

An exception object is created when a Python script raises an exception. If the script explicitly doesn't handle the exception, **the program will be forced to terminate abruptly**.

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

**Ans:**

A direct logic is followed to catch exceptions in Python. When an exception occurs, the Python interpreter stops the current process. It is handled by passing through the calling process. If not, the program will crash.

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

**Ans:**

An unhandled exception displays an error message and the program suddenly crashes. To avoid such a scenario, there are two methods to handle Python exceptions:

1. **Try**– This method catches the exceptions raised by the program
2. **Raise**– Triggers an exception manually using custom exceptions

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

**Ans:**

The statements in the finally block will always be executed, whether an exception is thrown or not.