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

***Ans.* Exception Description**

ArithmeticError Raised when an error occurs in numeric calculations

AssertionError Raised when an assert statement fails

AttributeError Raised when attribute reference or assignment fails

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

***Ans.***When an exception occurred, if you don't handle it, the program terminates abruptly and the code past the line that caused the exception will not get executed.

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

***Ans.***You can also provide a generic except clause, which handles any exception. After the except clause(s), you can include an else-clause. The code in the else-block executes if the code in the try: block does not raise an exception. The else-block is a good place for code that does not need the try: block's protection.

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

***Ans.***To avoid such a scenario, there are two methods to handle Python exceptions: Try – This method catches the exceptions raised by the program. 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 code within the try clause will be executed statement by statement. If no exception occurs during the execution, the execution will reach the break statement and the while loop will be left.