Q1. What is the purpose of the try statement?

Sol. The try statement catches and recovers from exceptions—it specifies a block of code to run, and one or more handlers for exceptions that may be raised during the block’s execution

Q2. What are the two most popular try statement variations?

Sol. The two common variations on the try statement are try/except/else (for catching exceptions) and try/finally (for specifying cleanup actions that must occur whether an exception is raised or not). In Python 2.4, these were separate statements that could be combined by syntactic nesting; in 2.5 and later, except andfinally blocks may be mixed in the same statement, so the two statement forms

are merged. In the merged form, the finally is still run on the way out of the try, regardless of what exceptions may have been raised or handled.

Q3. What is the purpose of the raise statement?

Sol. The raise statement raises (triggers) an exception. Python raises built-in exceptions on errors internally, but your scripts can trigger built-in or user-defined exceptions with raise, too.

Q4. What does the assert statement do, and what other statement is it like?

Sol. The assert statement raises an AssertionError exception if a condition is false. It works like a conditional raise statement wrapped up in an if statement.

Q5. What is the purpose of the with/as argument, and what other statement is it like?

Sol. The with/as statement is designed to automate startup and termination activitiesthat must occur around a block of code. It is roughly like a try/finally statement in that its exit actions run whether an exception occurred or not, but it allows a richer object-based protocol for specifying entry *and* exit actions