Q1. What is the purpose of the try statement?

**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?

**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).**

Q3. What is the purpose of the raise statement?

**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?

**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, and can**

**be disabled with a –O switch**

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

**The with/as statement is designed to automate startup and termination activities**

**that 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, and may reduce code size. Still, it’s not quite as general, as it applies only to objects that support**

**its protocol; try handles many more use cases**.