# Python Exception Handling — Questions & Answers (Set 7)

## Q1. What is the purpose of the try statement?

The try statement is used to catch and handle exceptions that occur in a block of code.  
It prevents the program from crashing and allows graceful recovery or cleanup.

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

1) try/except — catch and handle exceptions.  
 Example:  
 try:  
 x = 1 / 0  
 except ZeroDivisionError:  
 x = None  
  
2) try/finally — ensure cleanup code always runs.  
 Example:  
 try:  
 f = open('data.txt')  
 finally:  
 f.close()  
  
Combinations like try/except/finally or try/except/else/finally are also common.

## Q3. What is the purpose of the raise statement?

raise is used to manually trigger an exception.  
It signals error conditions intentionally.  
Example:  
def divide(a, b):  
 if b == 0:  
 raise ZeroDivisionError('Cannot divide by zero')  
 return a / b

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

assert tests a condition, and if it is False, raises AssertionError.  
Used for debugging and sanity checks.  
It is like raise but automatic.  
Example:  
x = -1  
assert x >= 0, 'x must be non-negative'  
# Equivalent to:  
# if not (x >= 0): raise AssertionError('x must be non-negative')

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

with/as is used for context management, ensuring resources are acquired and released.  
It is like try/finally but cleaner.  
Example:  
with open('data.txt') as f:  
 content = f.read()  
# File automatically closed (like using try/finally)