Q1. What is the purpose of the try statement?

If you have some suspicious code that may raise an exception, you can defend your program by placing the suspicious code in a try: block.

A single try statement can have multiple except statements. This is useful when the try block contains statements that may throw different types of exceptions.

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

Try- except

Try – except- except

Try – except -else

Try -except-else-finally

Q3. What is the purpose of the raise statement?

You can raise exceptions in several ways by using the raise statement. The general syntax for the raise statement is as follows.

raise [Exception [, args [, traceback]]]

Here, Exception is the type of exception (for example, NameError) and argument is a value for the exception argument. The argument is optional; if not supplied, the exception argument is None. The final argument, traceback, is also optional (and rarely used in practice), and if present, is the traceback object used for the exception.

Raise – . **raise** statement helps in triggering exception as per programming logic. If the expression is evaluates true then the error is raises.The raise keyword is used to raise an exception.here our condition i.e expression is true -1<0. Hence this raises exception.

x = -1

if x < 0:

raise Exception("Sorry, no numbers below zero")

o/p -Traceback (most recent call last):

File "./prog.py", line 3, in <module>

Exception: Sorry, no numbers below zero

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

**assert** statement takes a boolean condition output of which is True, Further Program Executes. if output of assert statement is False, it raises an Assertion Error.

Example :

**assert**(2 **==** 4), f'this is not a valid statement'

**---------------------------------------------------------------------------**

**AssertionError** Traceback (most recent call last)

**<ipython-input-2-b7228a17c25c>** in <module>**()**

**----> 1 assert(2** **==** **4),** **f'this is not a valid statement'**

**AssertionError**: this is not a valid statement

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

The with statement is a replacement for commonly used try/finally error-handling statements. A common example of using the with statement is opening a file. To open and write to a file in Python, you can use the with statement as follows:

# without using with statement

file **=** open('file\_path', 'w')

**try**:

    file.write('hello world')

**finally**:

    file.close()

# using with statement

with open('file\_path or name', 'w') as file:

    file.write('hello world !')

When we use a **with** statement for file reading, there is no need for programmer to explicitly takecare of activities like resource deallocation and file closing by using file.close() method. **with** statement itself ensures proper acquisition and release of resources. this avoids triggering of exceptions if file closing is unknowingly forgotten in the code execution.