Q1. What is the purpose of the try statement?

Ans: Whenever a certain piece of code is not handled correctly, it is about to raise cetain errors or some Runtime Exceptions. To handle those exceptions or errors, we need to use exception handling concept.

With the help of try statement, the code which is about to raise some error or exceptions, we can place that code inside try block.

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

Ans:

**Try/Except/Else**:

while True:

try:

num = int(input("Enter an int: "))

except Exception as e:

print(e)

else:

print("Thank you for the integer!")

break

**Explanation**: If user enters a valid input i.e., which is type of int, then else block will execute otherwise exception is handled and loop will run again.

**Try/Except/Finally:**

count = 0

while True:

try:

num = int(input("Enter an int: "))

break

except Exception as e:

print(e)

finally:

count += 1

print("Attempt #:",count)

**Explanation**: In this usecase, regardless of exception is handled or not finally block will execute always.

Q3. What is the purpose of the raise statement?

Ans: Python raise Keyword is used to raise exceptions or errors. The raise keyword raises an error and stops the control flow of the program. It is used to bring up the current exception in an exception handler so that it can be handled further up the call stack.

**Code:**

def fun():

raise Exception("Hello")

try:

fun()

except Exception as e:

print(e)

**Advantages of the raise keyword:**

a. It helps us raise exceptions when we may run into situations where execution can’t proceed.

b. It helps us reraise an exception that is caught.

c. Raise allows us to throw one exception at any time.

d. It is useful when we want to work with input validations.

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

Ans:

a. The assert keyword is used when debugging code.

b. The assert keyword lets you test if a condition in your code returns True, if not, the program will raise an AssertionError.

We can write a message to be written if the code returns False, check the example below:

x = "hello"

#if condition returns True, then nothing happens:

assert x == "hello, welcome to assertion statement"

#if condition returns False, AssertionError is raised:

assert x == "Welcome to Assertion Error block"

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

Ans: In Python, with statement is used in exception handling to make the code cleaner and much more readable. It simplifies the management of common resources like file streams.

Without with statement:

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

file.write('hello world !')

file.close()

Using with statement:

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

file.write('hello world !')

When using with statement, there is no need to call file.close(). The with statement itself ensures proper acquisition and release of resources. An exception during the file.write() call in the first implementation can prevent the file from closing properly which may introduce several bugs in the code, i.e. many changes in files do not go into effect until the file is properly closed. Also, 'as' will act as alias(file descriptor) to with statement