Q1. What is the purpose of the try statement?

The purpose of the try statement in Python is to define a block of code where exceptions may occur. It is used in conjunction with the except and optionally the finally clauses to handle and manage exceptions.

try:

# Code that may raise exceptions

except ExceptionType1:

# Code to handle exceptions of type ExceptionType1

except ExceptionType2:

# Code to handle exceptions of type ExceptionType2

finally:

# Code that will be executed regardless of exceptions

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

The two most popular variations of the try statement in Python are:

1. try-except statement: This variation allows you to catch and handle specific exceptions that may occur within the try block. You can specify one or more except clauses after the try block, each corresponding to a different exception type. If an exception of the specified type occurs, the corresponding except block is executed to handle the exception. Here's an example:

try:

# Code that may raise exceptions

except ExceptionType1:

# Code to handle exceptions of type ExceptionType1

except ExceptionType2:

# Code to handle exceptions of type ExceptionType2

1. try-finally statement: This variation allows you to define a finally block that will be executed regardless of whether an exception occurred or not. The code within the finally block is typically used to perform cleanup actions or finalize operations that should always be executed, such as closing files or releasing resources. Here's an example:

try:

# Code that may raise exceptions

finally:

# Code that will be executed regardless of exceptions

Q3. What is the purpose of the raise statement?

The raise statement in Python is used to explicitly raise an exception. It allows you to generate and raise exceptions programmatically, instead of relying on exceptions that occur naturally during the execution of your code.

Eg:

def divide(a, b):

if b == 0:

raise ValueError("Cannot divide by zero")

return a / b

try:

result = divide(10, 0)

except ValueError as e:

print("Error:", str(e))

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

The assert statement in Python is used for debugging and testing purposes. It allows you to check if a given condition is true and raises an AssertionError exception if the condition is false. It is primarily used to assert that certain conditions or assumptions in your code are true at specific points.

The syntax of the assert statement is as follows:

assert condition, message

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

The purpose of the with/as statement is to provide a convenient and concise way to manage resources by automatically handling the setup and teardown operations. It ensures that the necessary setup code is executed before entering the block of code and that the cleanup code is executed when exiting the block, even if an exception occurs.

The syntax of the with/as statement is as follows:

with open('file.txt', 'r') as file:

contents = file.read()

# Perform operations with the file contents

# After the block, the file is automatically closed