Error and Exception Handling:

1. What is the correct order of execution in a try-except-else-finally block? a) try, except, else, finally b) try, else, except, finally c) try, finally, except, else d) try, except, finally, else
2. Which of the following is not a built-in exception in Python? a) ValueError b) TypeError c) IndexError d) LoopError
3. What will be the output of the following code?

python

Copy

try:

x = 1 / 0

except ZeroDivisionError:

print("A")

except ArithmeticError:

print("B")

a) A b) B c) AB d) No output

1. Which statement is used to raise an exception in Python? a) throw b) raise c) except d) error
2. What is the purpose of the finally clause in a try-except block? a) To define the main code to be executed b) To handle exceptions that weren't caught by except clauses c) To execute code regardless of whether an exception occurred d) To raise a final exception if none were caught
3. What will be the output of the following code?

python

Copy

def func():

try:

return 1

finally:

return 2

print(func())

a) 1 b) 2 c) None d) Exception

1. Which of the following is true about custom exceptions in Python? a) They must inherit from the BaseException class b) They can't have custom attributes c) They are defined using the exception keyword d) They can inherit from any existing exception class
2. What does the else clause do in a try-except block? a) It's executed if no exception occurs in the try block b) It's always executed, regardless of exceptions c) It's executed if a specific exception occurs d) It's used to define additional exception handlers
3. How can you catch multiple exceptions in a single except clause? a) except (Exception1, Exception2): b) except Exception1 or Exception2: c) except [Exception1, Exception2]: d) except {Exception1, Exception2}:
4. What will be the output of the following code?

python

Copy

try:

assert False, "Error occurred"

except AssertionError as e:

print(str(e))

a) AssertionError b) Error occurred c) False d) No output

Error and Exception Handling:

1. Answer: a) try, except, else, finally Explanation: This is the correct order of execution for a try-except-else-finally block.
2. Answer: d) LoopError Explanation: LoopError is not a built-in exception in Python.
3. Answer: a) A Explanation: ZeroDivisionError is caught first, so "A" is printed. The more general ArithmeticError is not reached.
4. Answer: b) raise Explanation: The raise statement is used to raise exceptions in Python.
5. Answer: c) To execute code regardless of whether an exception occurred Explanation: The finally clause is always executed, whether an exception occurred or not.