**Q1. Describe three applications for exception processing.**

1. **Input Validation**:
   * Handle invalid user input gracefully.
   * Example:

try:

age = int(input("Enter your age: "))

except ValueError:

print("Invalid input. Please enter a number.")

1. **File Handling**:
   * Manage file-related errors (e.g., missing files, permission issues).
   * Example:

try:

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

content = file.read()

except FileNotFoundError:

print("File not found.")

1. **Network Operations**:
   * Handle network-related errors (e.g., connection issues, timeouts).
   * Example:

import requests

try:

response = requests.get("https://example.com")

except requests.ConnectionError:

print("Connection error.")

**Q2. What happens if you don't do something extra to treat an exception?**

* **Unhandled Exception**:
  + The program terminates abruptly.
  + A traceback is printed, showing the error and its location.
  + Example:

print(1 / 0) # ZeroDivisionError: division by zero

**Q3. What are your options for recovering from an exception in your script?**

1. **Try-Except Block**:
   * Catch and handle the exception.
   * Example:

try:

result = 1 / 0

except ZeroDivisionError:

result = "Undefined"

1. **Try-Except-Else Block**:
   * Execute code if no exception occurs.
   * Example:

try:

result = 1 / 1

except ZeroDivisionError:

result = "Undefined"

else:

print("No exception occurred.")

1. **Try-Except-Finally Block**:
   * Execute cleanup code regardless of exceptions.
   * Example:

try:

result = 1 / 0

except ZeroDivisionError:

result = "Undefined"

finally:

print("Cleanup code executed.")

**Q4. Describe two methods for triggering exceptions in your script.**

1. **Raise Statement**:
   * Manually raise an exception.
   * Example:

if condition:

raise ValueError("Invalid condition")

1. **Assert Statement**:
   * Raise an AssertionError if a condition is false.
   * Example:

assert condition, "Condition is false"

**Q5. Identify two methods for specifying actions to be executed at termination time, regardless of whether or not an exception exists.**

1. **Finally Block**:
   * Executes code after a try block, regardless of exceptions.
   * Example:

try:

result = 1 / 0

except ZeroDivisionError:

result = "Undefined"

finally:

print("Cleanup code executed.")

1. **Context Managers (with Statement)**:
   * Ensures cleanup code is executed when exiting a block.
   * Example:

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

content = file.read()

# File is automatically closed here