In [ ]:
```
Q1. What is an Exception in Python? Write the difference between Exceptions and Syntax errors.
```

```
An exception in Python is an event that occurs during the execution of a program, disrupting the
normal flow of the program's instructions. Exceptions can be caused by various factors, such as
invalid input, division by zero, or trying to access an undefined variable
```

```
Difference between Exceptions and Syntax errors:

Syntax errors: These errors occur when the Python interpreter cannot interpret the code due to
incorrect syntax. They are detected during the parsing phase before the code is executed.

Exceptions: These errors occur during the execution of a program. They are not detected by the
Python interpreter during the parsing phase but during runtime when an unexpected situation
arises
```

In [ ]:
```
Q.2- What happens when an exception is not handled? Explain with an example.
```

```
When an exception is not handled, it propagates up the call stack until it encounters an
appropriate exception handler. If there is no handler, the program terminates, and an error
message is displayed.
```

In [ ]:
```
# Division by zero exception not handled
result = 10 / 0
print("This line will not be executed.")
```

In [ ]:
```
Q3. Which Python statements are used to raise and handle exceptions? Explain with an example.
```

In [2]:
```
try:
    # Code that may cause an exception
    result = 10 / 0
except ZeroDivisionError:
    # Handling the ZeroDivisionError
    print("Cannot divide by zero!")
```

```
Cannot divide by zero!
```

In [ ]:
```
Q4. Explain with an example: try, except, else, and finally.
```

In [3]:
```
try:
    # Code that may cause an exception
    result = 10 / 2
except ZeroDivisionError:
    # Handling the ZeroDivisionError
    print("Cannot divide by zero!")
else:
    # Executed if no exception is raised
    print("Division successful!")
finally:
    # Always executed, whether an exception occurred or not
    print("This block always runs.")
```

```
Division successful!
This block always runs.
```

In [ ]:
```
Q5. What are Custom Exceptions in Python? Why do we need Custom Exceptions? Explain with an example
```

Custom exceptions in Python are user-defined exceptions created by deriving from the Exception class. They allow programmers to create their own exception types based on specific application requirements.

In [4]:
```
class CustomError(Exception):
    def __init__(self, message):
        super().__init__(message)

try:
    # Code that may raise a custom exception
    raise CustomError("This is a custom exception.")
except CustomError as ce:
    # Handling the custom exception
    print(f"Custom Exception: {ce}")
```

```
Custom Exception: This is a custom exception.
```

In [ ]:
```
Q6. Create a custom exception class. Use this class to handle an exception.
```

In [5]:
```
class CustomError(Exception):
    def __init__(self, message):
        super().__init__(message)

try:
    # Code that may raise a custom exception
    raise CustomError("This is a custom exception.")
except CustomError as ce:
    # Handling the custom exception
    print(f"Custom Exception: {ce}")
```

```
Custom Exception: This is a custom exception.
```

In [ ]:
```

```