1. What is the role of try and exception block?

Ans: The try-except block in Python is used for exception handling. It allows you to handle and recover from exceptional conditions or errors that may occur during the execution of your code.

1. What is the syntax for a basic try-except block?

Ans: try:

# Code block where an exception might occur

except ExceptionType:

# Code block to handle the exception

1. What happens if an exception occurs inside a try block and there is no matching except block?

Ans: If no suitable except block is found, the program will terminate and an error message will be displayed.

1. What is the difference between using a bare except block and specifying a specific exception type?

Ans: A bare except block, without specifying any exception type, will catch and handle any exception that occurs inside the try block. It acts as a catch-all for any type of exception. While it can be convenient to catch all exceptions, it can also make it difficult to diagnose and handle specific errors because the code inside the except block will execute regardless of the exception type. It can lead to unintentionally catching and handling exceptions that should be handled differently.

On the other hand, Specifying a specific exception type in an except block allows you to catch and handle only that particular type of exception. It provides more control over error handling because you can write specific code to handle different types of exceptions differently.

1. Can you have nested try-except blocks in Python? If yes, then give an example.

Ans: Yes, it is possible to have nested try-except blocks in Python

try:

# Outer try block

# Code that may raise exceptions

try:

# Inner try block

# Code that may raise exceptions specific to this block

except ValueError:

# Exception handling specific to ValueError in the inner try block

except:

# Exception handling for other exceptions in the inner try block

except:

# Exception handling for any other exceptions in the outer try block

1. Can we use multiple exception blocks, if yes then give an example.

Ans: try:

# Outer try block

# Code that may raise exceptions

try:

# Inner try block

# Code that may raise exceptions specific to this block

except ValueError:

# Exception handling specific to ValueError in the inner try block

except:

# Exception handling for other exceptions in the inner try block

except:

# Exception handling for any other exceptions in the outer try block

1. Write the reason due to which following errors are raised:
   1. EOFError: It typically occurs when the user or the file input ends abruptly before providing the expected input.
   2. FloatingPointError: This error is raised when a floating-point operation fails to produce a valid result. It can occur due to various reasons, such as division by zero, taking the square root of a negative number etc
   3. IndexError: This error is raised when you try to access a sequence, such as a list, tuple, or string, using an index that is outside the valid range of indices
   4. MemoryError: It occurs when your program tries to allocate more memory than the system can provide or when the available memory is heavily fragmented.
   5. OverflowError: It occurs when you try to perform an arithmetic calculation that results in a value that is too large to be stored within the given numeric type.
   6. TabError: This error is raised when there is an indentation-related issue in code
   7. ValueError: This error is raised when a function receives an argument of the correct type but an invalid value.
2. Write code for the following given scenario and add try-exception block to it.
   1. Program to divide two numbers

Ans:

try:

dividend = int(input("Enter the dividend: "))

divisor = int(input("Enter the divisor: "))

result = dividend / divisor

print("Result:", result)

except ZeroDivisionError:

print("Error: Division by zero is not allowed.")

except ValueError:

print("Error: Invalid input. Please enter integers only.")

* 1. Program to convert a string to an integer

Ans:

num\_str = input("Enter a number: ")

num = int(num\_str)

print("Number:", num)

except ValueError:

print("Error: Invalid input. Please enter a valid integer.")

* 1. Program to access an element in a list

Ans:

try:

my\_list = [1, 2, 3, 4, 5]

index = int(input("Enter the index: "))

element = my\_list[index]

print("Element at index", index, ":", element)

except IndexError:

print("Error: Index is out of range.")

except ValueError:

print("Error: Invalid input. Please enter a valid integer.")

* 1. Program to handle a specific exception

Ans: the below program will handle only value errors and index errors

try:

my\_list = [1, 2, 3, 4, 5]

index = int(input("Enter the index: "))

element = my\_list[index]

print("Element at index", index, ":", element)

except IndexError:

print("Error: Index is out of range.")

except ValueError:

print("Error: Invalid input. Please enter a valid integer.")

* 1. Program to handle any exception

Ans:

try:

# Code that may raise an exception

except Exception as e:

print("An error occurred:", str(e))