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

Ans: The try and except block in python are used to handle errors.

Try = we write the code here that might be produce an error.

Except = we write the code here that should run if an error occurs in try block.

e.g.

try:

x = 10/0

except ZeroDivisionError:

print(“We can’t divide by zero.”)

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

Ans:

try:

# write the code that might occur an error

except ExceptionType:

# Code that runs if error occurs in try block.

**3.** **What happens if an exception occurs inside a try block and there is no matching**

**except block?**

Ans:

If an exception occurs inside a try block and there in no matching except block, the program will terminate and display a traceback error. The exception will not be handled. And also the rest of the code after try and except will not be executed.

**4. What is the difference between using a bare except block and specifying a specific**

**exception type?**

Ans:

The difference between bare except block and specific exception is:

**Bare except block**: It catches all exceptions regardless of their type, This is generally not recommended because it can catches unexpected exceptions and make debugging difficult.

try:

# Code that might raise an exception

pass

except:

# Code that runs for any exception

pass

**Specific exception block**: It catches only the specified type of exception and this is recommended because it makes code clearer and make debugging easy.

try:

# Code that might raise an exception

pass

except ZeroDivisionError:

# Code that runs for ZeroDivisionError

pass

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

Ans:

Yes, we can have nested try-except block in python:

e.g.

try:

#outer try block

x = int(input(“Enter a number: ”))

try:

# inner try block

result = 10 / x

except ZeroDivisionError:

print(“You can’t divide by zero.”)

except ValueError:

print(“Please provide valid input!”)

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

Ans:

Yes, you can use multiple exception blocks to handle exceptions, here is an example:

try:

x = int(input(“Enter a number: ”))

result = 10 / x

print(f“Result: {result}”

except ZeroDivisionError:

print(“You can’t divide by zero.”)

except ValueError:

print(“Invalid input! Please provide valid input..”)

7. **Write the reason due to which following errors are raised:**

1. EOFError: No data to read at the end of the file
2. FloatingPointError: issue with floating point arithmetic
3. IndexError: Trying to access an index outside the range
4. MemoryError: running out of available memory
5. OverflowError: Result are two large for data type
6. TabError: Inconsistent use of tabs and spaces
7. ValueError: Inappropriate value for an operation

8. **Write code for the following given scenario and add try-exception block to it.**

1. Program to divide two numbers

try:

num1 = float(input(“Enter the first digit: ”))

num2 = float(input(“Enter the second digit: ”))

result = num1 / num2

print(result)

except ZeroDivisionError:

print(“You can’t divide by zero.”)

except ValueError:

print(“Invalid input! Please provide valid input..”)

1. Program to convert a string to an integer

try:

num\_str = input(“Enter a number as a string: ”))

num\_int = int(num\_str)

print(f"The integer value of '{num\_str}' is: {num\_int}")

except ValueError:

print(“Invalid input! Please enter a valid integer string.”)

1. Program to access an element in a list

try:

my\_list = [10,20,30,40,50]

index = int(input(“Please enter the index to access: ”))

element = my\_list[index]

print(f"The element at index {index} is: {element}”)

except IndexError:

print(“Index out of range.”)

1. Program to handle a specific exception

try:

my\_dict = {“name”: “Alice”, “Age”: 22}

key = input(“Enter the key to access in the dictionary: ”)

value = my\_dict[key]

print(f"The value for key '{key}' is: {value}")

except KeyError:

print(f"Error: Key '{key}' not found.")

1. Program to handle any exception

try:

num1 = int(input("Enter the first number: "))

num2 = int(input("Enter the second number: "))

result = num1 / num2

print(f"The result of {num1} divided by {num2} is: {result}")

except Exception as e:

print(f"An error occurred: {e}")