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

A: Try block: in try block we suspect that this line of code might have caused an error and we try to handle it.

Exception block: In except block, we handle exceptions which occurs in try block. For example, number divided by zero gives ‘ZeroDivisionError’. As we are aware that this error will occur when user put 0 in denominator and add this exception to handle it in exception block

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

A: try:

Code………….

except:

Code……………

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

A: If there are no matching exceptions in the code then the program will terminate and gives corresponding error. For example,

try:

x = '10'

y = 7

z = x+y

print(z)

except ZeroDivisionError:

print('number cannot be divided by zero')

In above code, there are no matching exceptions then the program will execute and will throw ValueError.

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

exception type?

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

A: Yes, there can be nested try expect block. For example,

try:

num1 = int(input("Enter a number: "))

num2 = int(input("Enter a number: "))

try:

result = num1 / num2

print("Result:", result)

except ZeroDivisionError:

print("Error: Cannot divide by zero.")

except ValueError:

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

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

A: Yes, there can be multiple exceptions in a code, for example:

try:

a = int(input('enter first number: '))

b = int(input('enter second num: '))

x = a/b

print('result: ',x)

except ZeroDivisionError:

print('number cant be divided by zero')

except ValueError:

print('you have entered value other than integer')

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

a. EOFError: Raised when the input() function hits an end-of-file condition (EOF  
b. FloatingPointError : Raised when a floating point operation fails.

c. IndexError: when we can for an index value which is out of the list then this error occurs.  
d. MemoryError: raised when operation is out of memory  
e. OverflowError: Raised when the result of an arithmetic operation is too large to be expressed.  
f. TabError : Raised when indentation contains mixed tabs and spaces.

g. ValueError : Raised when user input incorrect data type. For example: try:

a = int(input('enter first number: '))

b = int(input('enter second num: '))

x = a/b

print('result: ',x)

except ValueError:

print('you have entered value other than integer')

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

1. Program to divide two numbers

A: try:

a = int(input('enter first number: '))

b = int(input('enter second num: '))

x = a/b

print('result: ',x)

except ZeroDivisionError:

print('number cant be divided by zero')

1. Program to convert a string to an integer

A: num1 = input('enter a number: ')

num2 = int(num1)

type(num1)

Output: enter a number: 3

Out[14]:

str

type(num2)

output:

int

1. Program to access an element in a list

A: try:

l = [1,2,3,'r']

print(l[4])

except IndexError:

print('Enter Correct index number')

Output: Enter Correct index number

1. Program to handle a specific exception

A: try:

dic = {'name':'navtej','class':'job','educ':'btech'}

print(dic['city'])

except KeyError:

print('The key you have entered is not available in the dictionary mentioned')

Output: The key you have entered is not available in the dictionary mentioned

1. Program to handle any exception

A: try:

num1 = int(input("Enter the number 1: "))

num2 = int(input("Enter the number 2: "))

result = num1 / num2

print("Result:", result)

except Exception as e:

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

Output: Enter the number 1: 1

Enter the number 2: s

An exception occurred: invalid literal for int() with base 10: 's'