Exception Handling Questions (1–8) with Answers

---

1. Write a program to take two numbers from the user and divide them. Handle `ZeroDivisionError`.

```
try:
    a = int(input("Enter first number: "))
    b = int(input("Enter second number: "))
    result = a / b
    print("Result:", result)
except ZeroDivisionError:
    print("Cannot divide by zero.")
```

---

2. Take a number input and convert it to integer. Handle `ValueError`.

```
try:
    num = int(input("Enter a number: "))
    print("You entered:", num)
except ValueError:
    print("Invalid input. Please enter an integer.")
```

---

3. Take a list of numbers. Ask the user for an index and print the value at that index. Handle `IndexError`.

```
numbers = [10, 20, 30]
try:
    index = int(input("Enter index (0-2): "))
    print("Value at index:", numbers[index])
except IndexError:
    print("Invalid index.")
```

---

4. Create a program that takes two numbers, performs division, and uses multiple `except` blocks to handle both `ValueError` and `ZeroDivisionError`.

```
try:
    a = int(input("Enter numerator: "))
    b = int(input("Enter denominator: "))
    print("Result:", a / b)
except ValueError:
    print("Please enter valid integers.")
except ZeroDivisionError:
    print("Cannot divide by zero.")
```

---

5. Write a program to perform safe integer conversion from input using `try-except` and print “Invalid input” if it fails.

```
try:
    value = int(input("Enter an integer: "))
    print("You entered:", value)
except ValueError:
    print("Invalid input")
```

---

6. Use `try-except-finally` to demonstrate a risky division and always print “Done” at the end.

```
try:
    a = int(input("Enter numerator: "))
    b = int(input("Enter denominator: "))
    print("Result:", a / b)
except ZeroDivisionError:
    print("Division by zero not allowed.")
finally:
    print("Done")
```

---

7. Demonstrate nested `try-except` blocks for taking user input and dividing two numbers.

```
try:
    a = int(input("Enter first number: "))
    b = int(input("Enter second number: "))
    try:
        print("Result:", a / b)
    except ZeroDivisionError:
        print("Cannot divide by zero.")
except ValueError:
    print("Invalid input. Please enter integers only.")
```

---

8. Take input from user for an index and access that element in a list. If any error occurs, handle it using a generic `except:` block.

```
my_list = [1, 2, 3, 4, 5]
try:
    index = int(input("Enter an index: "))
    print("Element:", my_list[index])
except:
    print("Something went wrong.")
```

List and Tuple Questions (9–25) with Answers

---

9. Create a list of 5 student names. Print each name using a loop.

```
students = ["Alice", "Bob", "Charlie", "David", "Eve"]
for name in students:
    print(name)
```

---

10. Write a program that appends 5 numbers entered by the user into a list and prints it.

```
nums = []
for _ in range(5):
    n = int(input("Enter a number: "))
    nums.append(n)
print("List:", nums)
```

---

11. Write a function that takes a list of numbers and returns the maximum value.

```
def get_max(lst):
    return max(lst)

print(get_max([10, 45, 2, 99]))
```

---

12. Sort a list of integers and print the result in both ascending and descending order.

```
nums = [5, 3, 8, 1, 9]
print("Ascending:", sorted(nums))
print("Descending:", sorted(nums, reverse=True))
```

---

13. Write a program to remove duplicates from a list.

```
nums = [1, 2, 2, 3, 4, 4, 5]
unique = []
for n in nums:
    if n not in unique:
        unique.append(n)
print("Without duplicates:", unique)
```

---

14. Create a tuple of 5 cities and print them using a loop.

```
cities = ("Kathmandu", "Pokhara", "Biratnagar", "Lalitpur", "Butwal")
for city in cities:
```

```
    print(city)
```

---

15. Ask the user to input 3 numbers and store them in a tuple. Then print the tuple.

```
numbers = []
for _ in range(3):
    numbers.append(int(input("Enter a number: ")))
tuple_data = tuple(numbers)
print("Tuple:", tuple_data)
```

---

16. Write a function that takes a tuple of numbers and returns their sum.

```
def sum_tuple(t):
    return sum(t)

print(sum_tuple((1, 2, 3, 4)))
```

---

17. Given a list of numbers, write a function that returns a list containing only the even numbers.

```
def get_even(lst):
    return [n for n in lst if n % 2 == 0]

print(get_even([1, 2, 3, 4, 5, 6]))
```

---

18. Create a list of words and count how many start with the letter ‘a’.

```
words = ["apple", "banana", "apricot", "cherry", "avocado"]
count = 0
for word in words:
    if word.startswith('a'):
        count += 1
print("Count:", count)
```

---

19. Replace the second item in a list with a new value and print the updated list.

```
lst = [10, 20, 30, 40]
lst[1] = 99
print("Updated list:", lst)
```

---

20. Demonstrate slicing on a list to get the first 3 and last 2 elements.

```
lst = [1, 2, 3, 4, 5, 6]
print("First 3:", lst[:3])
print("Last 2:", lst[-2:])
```

---

21. Write a program that joins two tuples and prints the combined result.

```
t1 = (1, 2, 3)
t2 = (4, 5)
combined = t1 + t2
print("Combined tuple:", combined)
```

---

22. Ask the user for a sentence and count how many words it contains using a list.

```
sentence = input("Enter a sentence: ")
words = sentence.split()
print("Word count:", len(words))
```

---

23. Create a program that stores names in a list and allows the user to search for a name using the `in` operator.

```
names = ["Alice", "Bob", "Charlie"]
search = input("Enter name to search: ")
if search in names:
```

```
    print("Found")
else:
    print("Not found")
```

---

24. Take a list of numbers and reverse it using both slicing and the `reverse()` method.

```
nums = [1, 2, 3, 4, 5]
print("Reversed (slicing):", nums[::-1])
nums.reverse()
print("Reversed (method):", nums)
```

---

25. Given a list of numbers, create a new list with their squares.

```
nums = [1, 2, 3, 4]
squares = []
for n in nums:
    squares.append(n ** 2)
print("Squares:", squares)
```