11- EXCEPTION HANDLING

Ex. No. : 11.1 **Date: 25/05/24**

Register No.: 230701348 **Name: N Subramanian**

EXCEPTION HANDLING

To find whether a digit lies in the specified range(1-100). Handling exceptions for invalid inputs and out-of-range numbers .

Input Format:

User inputs a number.

Output Format:

Confirm the input or print an error message if it's invalid or out of range.

For example:

Input	Result
1	Valid input.
101	Error: Number out of allowed range
rec	Error: invalid literal for int()

Program:

```
try:

   a=input()     if(int(a)>0
and int(a)<101):
       print("Valid
input.")     else:         print("Error: Number out
of allowed range") except:
   print("Error: invalid literal for int()")
```

Ex. No. : 11.2 **Date: 25/05/24**

Register No.: 230701348 **Name : N Subramanian**

EXCEPTION HANDLING

Write a Python program that performs division and modulo operations on two numbers provided by the user. Handle division by zero and non-numeric inputs.

Input Format:

Two lines of input, each containing a number.

Output Format:

Print the result of division and modulo operation, or an error message if an exception occurs.

For example:

Input	Result
10 2	Division result: 5.0 Modulo result: 0
7 3	Division result: 2.3333333333333335 Modulo result: 1
8 0	Error: Cannot divide or modulo by zero.

Program:

```
try:

    a=input()     b=input()
  c=int(a)/int(b)     d=int(a
)%int(b) except
ZeroDivisionError:
    print("Error: Cannot divide or modulo by zero.")
except:
    print("Error: Non-numeric input provided.")
else:
    print("Division
result:",c)     print("Modulo result:",d)
```


Ex. No. : 11.3 **Date: 25/05/24**

Register No.: 230701348 **Name: N Subramanian**

EXCEPTION HANDLING

Write a Python program that asks the user for their age and prints a message based on the age. Ensure that the program handles cases where the input is not a valid integer.

Input Format: A single line input representing the user's age.

Output Format: Print a message based on the age or an error if the input is invalid.

For example:

Input	Result
twenty	Error: Please enter a valid age.
25	You are 25 years old.
-1	Error: Please enter a valid age.

Program:
```
try:
   a=input()
  if int(a)>=0:
       print("You are",a,"years old.")
   else:        print("Error: Please enter a
valid age.") except:
   print("Error: Please enter a valid age.")
```

Ex. No. : 11.4 **Date: 25/05/24**

Register No.: 230701348 **Name: N Subramanian**

EXCEPTION HANDLING

Develop a Python program that safely calculates the square root of a number provided by the user. Handle exceptions for negative inputs and non-numeric inputs.

Input Format:

User inputs a number.

Output Format:

Print the square root of the number or an error message if an exception occurs.

For example:

Input	Result
16	The square root of 16.0 is 4.00
-4	Error: Cannot calculate the square root of a negative number.
rec	Error: could not convert string to float

Program:

```
import math
try:
   n=input()
   n=float(n)     if n < 0:         print("Error: Cannot calculate the square
root of a negative number.")     else:
       r= math.sqrt(n)
       print("The square root of {} is {:.2f}".format(n, r))


except ValueError:     print("Error: could not
convert string to float")
```

Ex. No. : 11.5 **Date: 25/05/24**

Register No.: 230701348 **Name: N Subramanian**

EXCEPTION HANDLING

Develop a Python program that safely performs division between two numbers provided by the user. Handle exceptions like division by zero and non-numeric inputs.

Input Format: Two lines of input, each containing a number.

Output Format: Print the result of the division or an error message if an exception occurs.

For example:

Input	Result
10 2	5.0
10 0	Error: Cannot divide or modulo by zero.
ten 5	Error: Non-numeric input provided.

Program: try:
a=input() b=inp
ut() c=float(a)/floa
t(b) except
ZeroDivisionError:
print("Error: Cannot divide or modulo by zero.")
except:
print("Error: Non-numeric input
provided.") else: print(c)