

11 - Exceptions

Problem Description:

# Out of Range Numbers

Write a Python script that asks the user to enter a number within a specified range (e.g., 1 to 100). Handle 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:

num = int(input()) if 1 <= num <= 100:

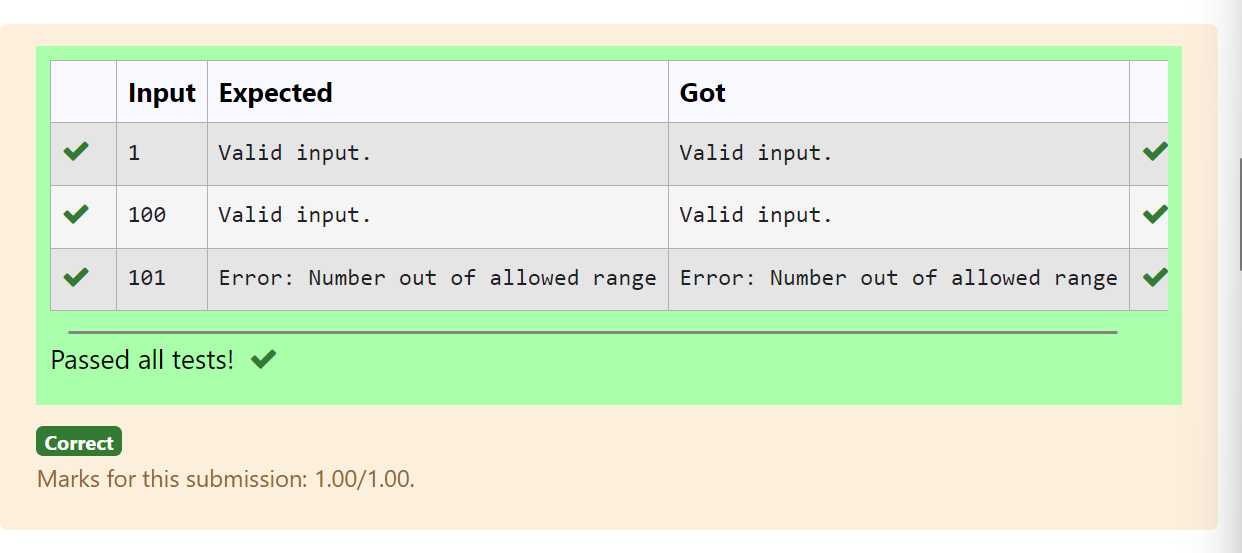
print("Valid input.")

else:

print("Error: Number out of allowed range") except ValueError:

print("Error: invalid literal for int()")







# Divide by Zero

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:

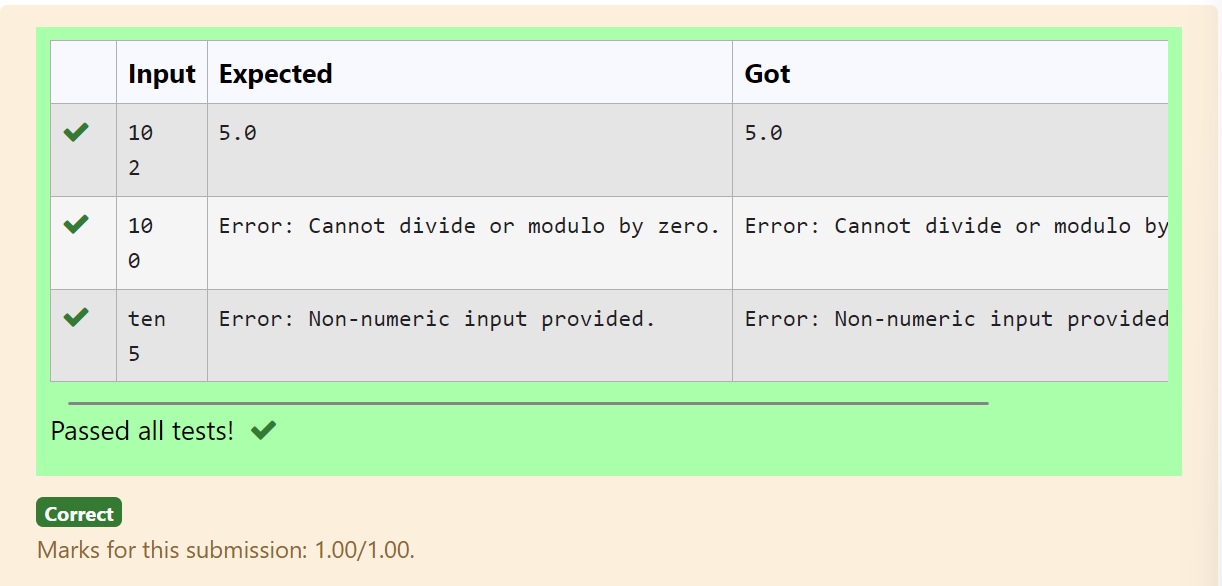
a=int(input()) b=int(input()) print(a/b)

except ValueError:

print("Error: Non-numeric input provided.") except ZeroDivisionError:

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





Problem Description:

# Valid Age

Write a Python script that asks the user to enter a number within a specified range (e.g., 1 to 100). Handle 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:

num = int(input())

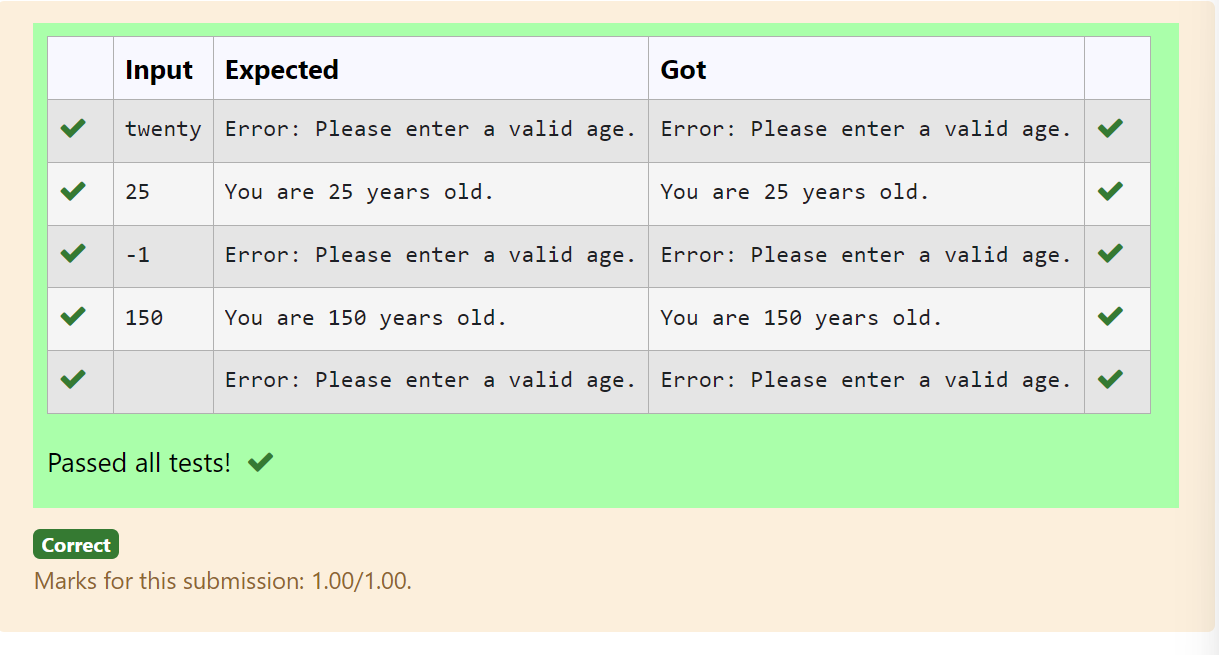
if 1 <= num <= 100: print("Valid input.")

else:

print("Error: Number out of allowed range") except ValueError:

print("Error: invalid literal for int()")







# Safe Square Root

Problem Description:

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**

try:

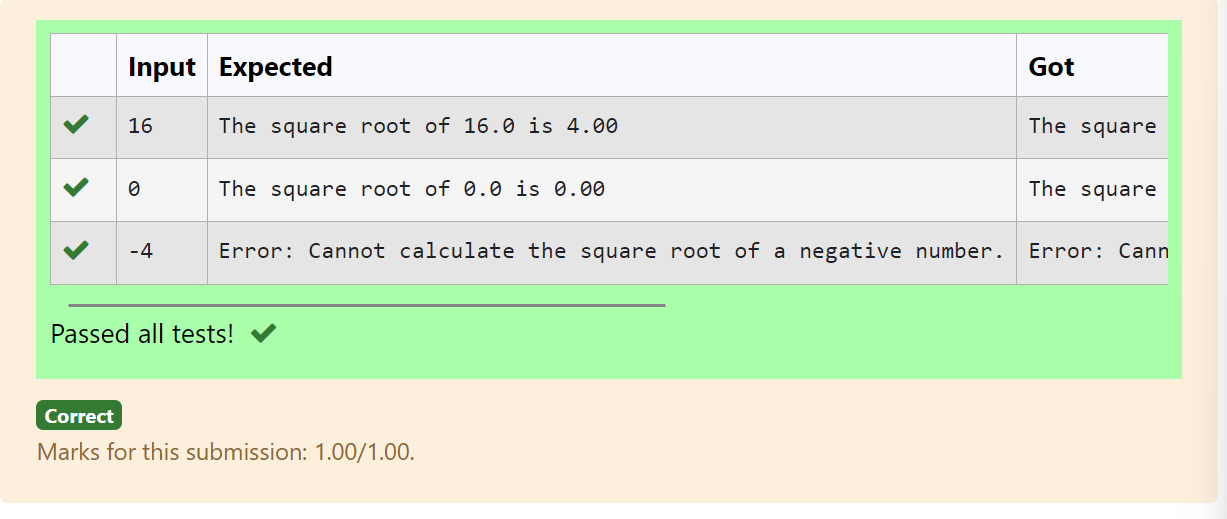
a=int(input()) if a>=0:

print("The square root of %.1f is %.2f"%(float(a),float(a\*\*0.5))) else:

print("Error: Cannot calculate the square root of a negative number.") except:

print("Error could not convert string to float")





Problem Description:

# Valid Integer

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** |
| --- | --- |
| 25 | You are 25 years old. |
| rec | Error: Please enter a valid age. |
| -5 | Error: Please enter a valid age. |

**PROGRAM**

try:

n=int(input()) if n>=1:

print("You are",n,"years old.") else:

print("Error: Please enter a valid age.") except:

print("Error: Please enter a valid age.")



