Question 1: Write a function to print your name and age.

In [5]:
```
def info(name, age):
    try:
        print(f"Name: {name}, Age: {age}")
    except Exception as e:
        print(f"Error: {e}")
info("Alex", 30)
```

```
Name: Alex, Age: 30
```

Question 2: Write a function that accepts two numbers and returns their sum.

In [6]:
```
def sum(num1, num2):
    try:
        return num1 + num2
    except TypeError:
        return "Error: Input must be numbers."
print(sum(5, 10))
```

```
15
```

Question 3: Write a function that takes a number and prints whether it is positive, negative, or zero.

In [3]:
```
def q3_check_sign(num):
    try:
        if num > 0:
            print("Positive")
        elif num < 0:
            print("Negative")
        else:
            print("Zero")
    except TypeError:
        print("Error: Input must be a number.")
q3_check_sign(10)
```

```
Positive
```

Question 4: Write a function to calculate the square of a given number.

In [4]:
```
def q4_square(num):
    try:
        return num ** 2
    except TypeError:
        return "Error: Input must be a number."
print(q4_square(9))
```

```
81
```

Question 5: Write a function to print all even numbers between 1 and 50.

In [5]:
```python
def q5_even_numbers():
    print(*(i for i in range(2, 51, 2)))
q5_even_numbers()
```

```
2 4 6 8 10 12 14 16 18 20 22 24 26 28 30 32 34 36 38 40 42 44 46 48 50
```

Question 6: Write a function that takes a number and prints its multiplication table.

In [6]:
```python
def q6_mult_table(num):
    try:
        for i in range(1, 11):
            print(f"{num} * {i} = {num * i}")
    except TypeError:
        print("Error: Input must be a number.")
q6_mult_table(7)
```

```
7 * 1 = 7
7 * 2 = 14
7 * 3 = 21
7 * 4 = 28
7 * 5 = 35
7 * 6 = 42
7 * 7 = 49
7 * 8 = 56
7 * 9 = 63
7 * 10 = 70
```

Question 7: Write a function to check whether a number is divisible by 5 or not.

In [7]:
```python
def q7_divisible_by_5(num):
    try:
        if num % 5 == 0:
            print(f"{num} is divisible by 5.")
        else:
            print(f"{num} is not divisible by 5.")
    except TypeError:
        print("Error: Input must be an integer.")
    except ZeroDivisionError:
        print("Error: Cannot use 0 as input for this type of check.")
q7_divisible_by_5(15)
```

```
15 is divisible by 5.
```

Question 8: Write a function that takes two strings and concatenates them.

In [8]:
```python
def q8_concatenate(s1, s2):
    try:
        return s1 + s2
    except TypeError:
        return "Error: Input must be strings."
print(q8_concatenate("Hello, ", "World!"))
```

```
Hello, World!
```

Question 9: Write a function to count how many times a word appears in a given sentence.

In [9]:
```
def q9_word_count(sent, word):
    try:
        return sent.lower().split().count(word.lower())
    except AttributeError:
        return "Error: Inputs must be strings."
print(q9_word_count("The fox is quick, and the fox is brown.", "fox"))
```

```
2
```

Question 10: Write a function to return the length of a string without using the built-in len() function.

In [10]:
```
def q10_string_length(s):
    try:
        count = 0
        for _ in s:
            count += 1
        return count
    except TypeError:
        return "Error: Input must be a string."
print(q10_string_length("Python"))
```

```
6
```

Question 11: Write a function to count the number of vowels in a given string.

In [11]:
```
def q11_vowel_count(s):
    try:
        vowels = "aeiouAEIOU"
        return sum(1 for char in s if char in vowels)
    except TypeError:
        return "Error: Input must be a string."
print(q11_vowel_count("Programming"))
```

```
3
```

Question 12: Write a function to check whether a string starts with a vowel.

In [12]:
```
def q12_starts_with_vowel(s):
    try:
        vowels = "aeiouAEIOU"
        if not s: return False
        return s[0] in vowels
    except TypeError:
        return "Error: Input must be a string."
print(q12_starts_with_vowel("Apple"))
```

```
True
```

Question 13: Write a function to remove all spaces from a given string.

In [13]:
```
def q13_remove_spaces(s):
    try:
        return s.replace(" ", "")
    except AttributeError:
        return "Error: Input must be a string."
print(q13_remove_spaces("H e l l o W o r l d"))
```

```
HelloWorld
```

Question 14: Write a function to replace all vowels in a string with *.

In [14]:
```
def q14_replace_vowels(s):
    try:
        vowels = "aeiouAEIOU"
        return "".join('*' if char in vowels else char for char in s)
    except TypeError:
        return "Error: Input must be a string."
print(q14_replace_vowels("Programming"))
```

```
Pr*gr*mm*ng
```

Question 15: Write a function to count capital and small letters in a string.

In [15]:
```
def q15_count_cases(s):
    try:
        cap_c = sum(1 for char in s if 'A' <= char <= 'Z')
        small_c = sum(1 for char in s if 'a' <= char <= 'z')
        print(f"Capital: {cap_c}, Small: {small_c}")
    except TypeError:
        print("Error: Input must be a string.")
q15_count_cases("PyThOn PrOgRaMmInG")
```

```
Capital: 9, Small: 8
```

Question 16: Write a function to reverse a string passed by the user.

In [16]:
```
def q16_reverse_string(s):
    try:
        return s[::-1]
    except TypeError:
        return "Error: Input must be a string."
print(q16_reverse_string("ReverseMe"))
```

```
eMesreveR
```

Question 17: Write a function to check if a string is a palindrome or not.

In [17]:
```
def q17_is_palindrome(s):
    try:
        clean_s = "".join(filter(str.isalnum, s)).lower()
        return clean_s == clean_s[::-1]
    except AttributeError:
        return "Error: Input must be a string."
print(q17_is_palindrome("Madam, I'm Adam"))
```

```
True
```

Question 18: Write a function to count the number of words in a given sentence.

In [18]:
```
def q18_word_count_sent(sent):
    try:
        return len(sent.split())
    except AttributeError:
        return "Error: Input must be a string."
print(q18_word_count_sent("This is a test sentence."))
```

```
5
```

Question 19: Write a function to find the most repeated character in a string.

In [1]:
```
from collections import Counter
def q19_most_repeated_char(s):
    try:
        s = s.lower().replace(" ", "")
        if not s: return "String is empty."
        counts = Counter(s)
        return counts.most_common(1)[0][0]
    except TypeError:
        return "Error: Input must be a string."
print(q19_most_repeated_char("Programming is fun"))
```

```
r
```

Question 20: Write a function to convert all lowercase letters to uppercase without using upper().

In [2]:
```
def q20_to_uppercase(s):
    try:
        def to_upper_char(char):
            if 'a' <= char <= 'z':
                return chr(ord(char) - 32)
            return char
        new_s = "".join(to_upper_char(char) for char in s)
        return new_s
    except TypeError:
        return "Error: Input must be a string."
print(q20_to_uppercase("hello World 123"))
```

```
HELLO WORLD 123
```

Question 21: Write a function to find the greatest element in a list.

In [21]:
```
def q21_greatest_element(lst):
    try:
        if not lst: return "List is empty."
        greatest = lst[0]
        for item in lst[1:]:
            if item > greatest:
                greatest = item
        return greatest
    except TypeError:
        return "Error: List elements must be comparable."
print(q21_greatest_element([10, 5, 20, 8]))
```

```
20
```

Question 22: Write a function to find the smallest element in a list.

In [22]:
```
def q22_smallest_element(lst):
    try:
        if not lst: return "List is empty."
        smallest = lst[0]
        for item in lst[1:]:
            if item < smallest:
                smallest = item
        return smallest
    except TypeError:
        return "Error: List elements must be comparable."
print(q22_smallest_element([10, 5, 20, 8]))
```

```
5
```

Question 23: Write a function to create a list by taking user inputs.

In [23]:
```
def q23_create_list(simulated_inputs=[10, "hello", 20]):
    """Simulates list creation by user input for notebook execution."""
    lst = []
    for item in simulated_inputs:
        lst.append(item)
    return lst
print(q23_create_list())
```

```
[10, 'hello', 20]
```

Question 24: Write a function to remove duplicates from a list.

In [24]:
```
def q24_remove_duplicates(lst):
    try:
        return list(dict.fromkeys(lst))
    except TypeError:
        seen = set()
        new_lst = []
        for item in lst:
            try:
                if item not in seen:
                    seen.add(item)
                    new_lst.append(item)
            except TypeError:
                if item not in new_lst: new_lst.append(item)
        return new_lst
print(q24_remove_duplicates([1, 2, 2, 3, 1, 4]))
```

```
[1, 2, 3, 4]
```

Question 25: Write a function to count even and odd numbers in a list.

In [25]:
```
def q25_count_even_odd(lst):
    try:
        int_lst = [int(x) for x in lst]
        even_c = sum(1 for x in int_lst if x % 2 == 0)
        odd_c = sum(1 for x in int_lst if x % 2 != 0)
        print(f"Even: {even_c}, Odd: {odd_c}")
    except ValueError:
        print("Error: List elements must be convertible to integers.")
    except TypeError:
        print("Error: Input must be a list.")
q25_count_even_odd([1, 2, 3, 4, 5, 6])
```

```
Even: 3, Odd: 3
```

Question 26: Write a function to find the sum of all elements in a list.

In [26]:
```
def q26_list_sum(lst):
    try:
        total = 0
        for x in lst:
            total += x
        return total
    except TypeError:
        return "Error: List elements must be numbers."
print(q26_list_sum([1, 2, 3, 4]))
```

```
10
```

Question 27: Write a function to sort a list without using the built-in sort() function. (Using Bubble Sort)

In [27]:
```
def q27_custom_sort(lst):
    try:
        n = len(lst)
        arr = lst[:]
        for i in range(n):
            for j in range(0, n-i-1):
                if arr[j] > arr[j+1]:
                    arr[j], arr[j+1] = arr[j+1], arr[j] # Swap
        return arr
    except TypeError:
        return "Error: List elements must be comparable."
print(q27_custom_sort([5, 1, 4, 2, 8]))
```

```
[1, 2, 4, 5, 8]
```

Question 28: Write a function to find the smallest string from a list of strings.

In [28]:
```
def q28_smallest_string(lst):
    try:
        if not lst: return "List is empty."
        return min(lst, key=len)
    except TypeError:
        return "Error: Input must be a list of strings."
print(q28_smallest_string(["apple", "banana", "kiwi", "orange"]))
```

```
kiwi
```

Question 29: Write a function to find the factorial of a given number.

In [29]:
```
def q29_factorial(num):
    try:
        num = int(num)
        if num < 0: return "Factorial is not defined for negative numbers."
        if num == 0: return 1
        fact = 1
        for i in range(1, num + 1):
            fact *= i
        return fact
    except ValueError:
        return "Error: Input must be an integer."
print(q29_factorial(5))
```

```
120
```

Question 30: Write a function to check if a number is prime or not.

In [30]:
```
def q30_is_prime(num):
    try:
        num = int(num)
        if num < 2: return False
        for i in range(2, int(num**0.5) + 1):
            if num % i == 0:
                return False
        return True
    except ValueError:
        return "Error: Input must be an integer."
print(q30_is_prime(11))
```

```
True
```

Question 31: Write a function to find all prime numbers between 1 and 100.

In [31]:
```
def is_prime_local(num):
    if num < 2: return False
    for i in range(2, int(num**0.5) + 1):
        if num % i == 0:
            return False
    return True

def q31_primes_1_to_100():
    return [i for i in range(1, 101) if is_prime_local(i)]
print(q31_primes_1_to_100())
```

```
[2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71,
73, 79, 83, 89, 97]
```

Question 32: Write a function to find the sum of digits of a number.

In [32]:
```
def q32_sum_digits(num):
    try:
        s_num = str(abs(int(num)))
        return sum(int(digit) for digit in s_num)
    except ValueError:
        return "Error: Input must be a number."
print(q32_sum_digits(12345))
```

```
15
```

Question 33: Write a function to calculate the Fibonacci series up to n terms.

In [33]:
```
def q33_fibonacci(n):
    try:
        n = int(n)
        if n <= 0: return []
        a, b = 0, 1
        series = []
        for _ in range(n):
            series.append(a)
            a, b = b, a + b
        return series
    except ValueError:
        return "Error: Input must be a non-negative integer."
print(q33_fibonacci(8))
```

```
[0, 1, 1, 2, 3, 5, 8, 13]
```

Question 34: Write a function to check if a number is an Armstrong number.

In [34]:
```
def q34_is_armstrong(num):
    try:
        num_int = int(num)
        if num_int < 0: return False
        s_num = str(num_int)
        n = len(s_num)
        arm_sum = sum(int(digit) ** n for digit in s_num)
        return arm_sum == num_int
    except ValueError:
        return "Error: Input must be an integer."
print(q34_is_armstrong(153))
```

```
True
```

Question 35: Write a function to create a text file and write "Hello, Python" in it.

In [35]:
```
def q35_create_file(file_name="q35_file.txt", content="Hello, Python"):
    try:
        with open(file_name, 'w') as f:
            f.write(content)
        return f"File '{file_name}' created and written successfully."
    except IOError as e:
        return f"File I/O Error: {e}"
print(q35_create_file())
```

```
File 'q35_file.txt' created and written successfully.
```

Question 36: Write a function to read content from a text file and display it.

In [36]:
```
def q36_read_file(file_name="q35_file.txt"):
    try:
        with open(file_name, 'r') as f:
            content = f.read()
        return content
    except FileNotFoundError:
        return f"Error: File '{file_name}' not found."
    except IOError as e:
        return f"File I/O Error: {e}"
print(q36_read_file())
```

```
Hello, Python
```

Question 37: Write a function to count the number of lines, words, and characters in a text file.

In [37]:
```
def q37_count_file_content(file_name="q35_file.txt"):
    try:
        with open(file_name, 'r') as f:
            content = f.read()

        lines = content.count('\n') + (1 if content else 0)
        words = len(content.split())
        chars = len(content)

        return f"Lines: {lines}, Words: {words}, Characters: {chars}"
    except FileNotFoundError:
        return f"Error: File '{file_name}' not found."
    except IOError as e:
        return f"File I/O Error: {e}"
print(q37_count_file_content())
```

```
Lines: 1, Words: 2, Characters: 13
```

Question 38: Write a function to append user input to an existing file.

In [38]:
```
def q38_append_to_file(file_name="q35_file.txt", new_data="\nAppended line."
    """Simulates appending data for notebook execution."""
    try:
        with open(file_name, 'a') as f:
            f.write(new_data)
        return f"Data appended to '{file_name}' successfully."
    except FileNotFoundError:
        return f"Error: File '{file_name}' not found."
    except IOError as e:
        return f"File I/O Error: {e}"
print(q38_append_to_file())
```

```
Data appended to 'q35_file.txt' successfully.
```

Question 39: Write a function that generates a random number between 1 and 100 and lets the user guess it.

In [39]:
```
import random
def q39_guess_number(target=None, guess=50):
    """Simulates one guess for notebook testability."""
    try:
        if target is None:
            target = random.randint(1, 100)
        guess = int(guess)
        if guess == target:
            return f"Guess {guess}: Correct! Target was {target}."
        elif guess < target:
            return f"Guess {guess}: Too low. Target was {target}."
        else:
            return f"Guess {guess}: Too high. Target was {target}."
    except ValueError:
        return "Error: Guess must be an integer."
    except Exception as e:
        return f"An unexpected error occurred: {e}"
print(q39_guess_number(target=75, guess=70))
```

```
Guess 70: Too low. Target was 75.
```

Question 40: Write a function to simulate rolling two dice and display their total.

In [40]:

```
import random
def q40_roll_dice():
    try:
        die1 = random.randint(1, 6)
        die2 = random.randint(1, 6)
        total = die1 + die2
        return f"Roll: {die1} + {die2} = {total}"
    except Exception as e:
        return f"An error occurred: {e}"
print(q40_roll_dice())
```

```
Roll: 2 + 5 = 7
```

Unnumbered Question: Write a function using recursion to calculate the sum of natural numbers up to n.

In [41]:

```
def q40a_recursive_sum(n):
    try:
        n = int(n)
        if n < 0: return "Input must be a non-negative integer."
        if n == 0:
            return 0
        else:
            # Recursive step
            return n + q40a_recursive_sum(n - 1)
    except (ValueError, RecursionError) as e:
        return f"Error: {e}"
print(q40a_recursive_sum(10))
```

```
55
```

Question 41: Write a function to check if two strings are anagrams of each other.

In [42]:

```
def q41_are_anagrams(s1, s2):
    try:
        s1_clean = sorted(s1.lower().replace(" ", ""))
        s2_clean = sorted(s2.lower().replace(" ", ""))
        return s1_clean == s2_clean
    except AttributeError:
        return "Error: Input must be strings."
print(q41_are_anagrams("listen", "silent"))
```

```
True
```

Question 42: Write a function that takes a sentence and returns the words in alphabetical order.

In [43]:
```
def q42_sort_words(sent):
    try:
        words = sent.split()
        words.sort()
        return " ".join(words)
    except AttributeError:
        return "Error: Input must be a string."
print(q42_sort_words("quick brown fox jumps over the lazy dog"))
```

```
brown dog fox jumps lazy over quick the
```

Question 43: Write a function that uses list comprehension to find all even squares between 1 and 50.

In [44]:
```
def q43_even_squares():
    return [x*x for x in range(1, 8) if (x*x) % 2 == 0]
print(q43_even_squares())
```

```
[4, 16, 36]
```

Question 44: Write a function that takes a list of numbers and returns a new list containing only unique elements.

In [45]:
```
def q44_unique_elements(lst):
    try:
        return list(set(lst))
    except TypeError:
        return "Error: List elements must be hashable."
print(q44_unique_elements([1, 2, 2, 3, 1, 4]))
```

```
[1, 2, 3, 4]
```

Question 45: Write a function using lambda to find the cube of a number.

In [46]:
```
def q45_lambda_cube(num):
    try:
        cube = lambda x: x ** 3
        return cube(num)
    except TypeError:
        return "Error: Input to lambda must be a number."
print(q45_lambda_cube(5))
```

```
125
```

Question 46: Write a function that shuffles the elements of a list randomly.

In [47]:
```
import random
def q46_shuffle_list(lst):
    try:
        new_lst = lst[:]
        random.shuffle(new_lst)
        return new_lst
    except TypeError:
        return "Error: Input must be a list."
test_list = [1, 2, 3, 4, 5]
print(f"Original: {test_list}, Shuffled: {q46_shuffle_list(test_list)}")
```

```
Original: [1, 2, 3, 4, 5], Shuffled: [3, 5, 4, 1, 2]
```

Question 47: Write a function that generates a random password of 8 characters using letters and digits.

In [48]:
```
import string
import random
def q47_generate_password(length=8):
    try:
        chars = string.ascii_letters + string.digits
        password = "".join(random.choice(chars) for _ in range(length))
        return password
    except Exception as e:
        return f"An error occurred: {e}"
print(q47_generate_password())
```

```
bQXJ5ePr
```

Question 48: Write a function to count how many numbers in a list are greater than the average of the list.

In [49]:
```
def q48_count_greater_than_avg(lst):
    try:
        if not lst: return 0
        avg = sum(lst) / len(lst)
        count = sum(1 for x in lst if x > avg)
        return count
    except TypeError:
        return "Error: List elements must be numbers."
    except ZeroDivisionError:
        return 0
print(q48_count_greater_than_avg([10, 20, 30, 40, 50]))
```

```
2
```

Question 49: Write a function to display a menu-based simple calculator using while True loop and functions for each operation.

In [ ]:
```
def add(x, y):
    return x + y

def subtract(x, y):
    return x - y

def multiply(x, y):
    return x * y

def divide(x, y):
    if y == 0:
        return "Error: Division by zero"
    return x / y

def calculator():
    while True:
        print("\nSimple Calculator Menu")
        print("1. Add")
        print("2. Subtract")
        print("3. Multiply")
        print("4. Divide")
        print("5. Exit")

        choice = input("Enter choice (1-5): ")

        if choice == '5':
            print("Exiting calculator.")
            break

        if choice in ('1', '2', '3', '4'):
            try:
                num1 = float(input("Enter first number: "))
                num2 = float(input("Enter second number: "))
            except ValueError:
                print("Invalid input. Please enter numeric values.")
                continue

            if choice == '1':
                print("Result:", add(num1, num2))
            elif choice == '2':
                print("Result:", subtract(num1, num2))
            elif choice == '3':
                print("Result:", multiply(num1, num2))
            elif choice == '4':
                print("Result:", divide(num1, num2))
        else:
            print("Invalid choice. Please select a number between 1 and 5.")

# Call the calculator function to start
calculator()
```

```
Simple Calculator Menu
1. Add
2. Subtract
3. Multiply
4. Divide
5. Exit
Result: 9.0

Simple Calculator Menu
1. Add
2. Subtract
3. Multiply
4. Divide
5. Exit
```

In [ ]:
```
### Question 50: Write a function to return greatest substring in any senten
```

In [3]:
```python
def q50_longest_word(sent):
    try:
        words = sent.split()
        if not words: return "Sentence is empty."
        longest_word = max(words, key=len)
        return longest_word
    except AttributeError:
        return "Error: Input must be a string."
print(q50_longest_word("The quick brown fox jumps over the lazy dog."))
```

```
quick
```

In [ ]: