# Input: A comma-separated string of numbers

input\_str = input("Enter numbers separated by commas: ")

# Convert the input string to a set of integers

numbers = set(map(int, input\_str.split(',')))

# Display the original set

print("Original Set:", numbers)

# 1. Length of set

print("Length of set:", len(numbers))

# 2. Maximum value

print("Maximum value:", max(numbers) if numbers else "Set is empty")

# 3. Minimum value

print("Minimum value:", min(numbers) if numbers else "Set is empty")

# 4. Sum of elements

print("Sum of elements:", sum(numbers))

# 5. Sorted list of elements

print("Sorted list of elements:", sorted(numbers))

# 6. All elements are true

print("All elements are true:", all(numbers))

# 7. Any element is true

print("Any element is true:", any(numbers))

# 8. List from set

print("List from set:", list(numbers))

# 9. Filtering: Elements greater than a threshold (e.g., 2)

threshold = 2

filtered = set(filter(lambda x: x > threshold, numbers))

print(f"Elements greater than {threshold}:", filtered)

# 10. Mapping: Double the elements

doubled = set(map(lambda x: x \* 2, numbers))

print("Doubled elements:", doubled)

# 11. Intersection with another set

another\_set = {3, 4, 5, 6}

intersection = numbers.intersection(another\_set)

print("Intersection with {3, 4, 5, 6}:", intersection)

# 12. Union with another set

union = numbers.union(another\_set)

print("Union with {3, 4, 5, 6}:", union)

# 13. Difference with another set

difference = numbers.difference(another\_set)

print("Difference with {3, 4, 5, 6}:", difference)

# 14. Symmetric difference with another set

symmetric\_difference = numbers.symmetric\_difference(another\_set)

print("Symmetric difference with {3, 4, 5, 6}:", symmetric\_difference)

# 15. Subset check

subset\_check = {1, 2, 3}.issubset(numbers)

print("Is {1, 2, 3} a subset of the original set?", subset\_check)

# 16. Superset check

superset\_check = numbers.issuperset({1, 2})

print("Is the original set a superset of {1, 2}?", superset\_check)

# 17. Count occurrences (from the input)

occurrences = {num: input\_str.count(str(num)) for num in numbers}

print("Occurrences of each number in the original input:", occurrences)

# 18. Convert to a frozenset

frozen\_set = frozenset(numbers)

print("Frozenset of the original set:", frozen\_set)

# 19. Set comprehension to square elements

squared\_set = {x\*\*2 for x in numbers}

print("Squared elements in a new set:", squared\_set)

# 20. Cartesian product using itertools

from itertools import product

cartesian\_product = set(product(numbers, repeat=2))

print("Cartesian product of the set with itself:", cartesian\_product)

# 21. Check if a specific element is in the set

element\_check = 2 in numbers

print("Is 2 in the original set?", element\_check)

# 22. Remove an element (if exists)

numbers\_copy = numbers.copy()

numbers\_copy.discard(2) # Use discard to avoid KeyError

print("Set after removing 2 (if it existed):", numbers\_copy)

# 23. Add a new element

numbers\_copy.add(7)

print("Set after adding 7:", numbers\_copy)

# 24. Clear the set

cleared\_set = numbers.copy()

cleared\_set.clear()

print("Set after clearing:", cleared\_set)

# 25. Convert set to a string representation

set\_str = str(numbers)

print("String representation of the set:", set\_str)

# 26. Create a new set from a string

string\_input = "hello"

char\_set = set(string\_input)

print("Set from string 'hello':", char\_set)

# 27. Check the length of the character set

print("Length of character set:", len(char\_set))

# 28. Generate a set from a range

range\_set = set(range(5)) # Numbers from 0 to 4

print("Set from range(5):", range\_set)

# 29. Generate a set with a condition

condition\_set = {x for x in range(10) if x % 2 == 0}

print("Set of even numbers from 0 to 9:", condition\_set)

# 30. Convert a list with duplicates to a set

duplicate\_list = [1, 2, 2, 3, 4, 4, 5]

unique\_set = set(duplicate\_list)

print("Set from a list with duplicates:", unique\_set)

# 31. Check if two sets are disjoint

disjoint\_check = numbers.isdisjoint({8, 9})

print("Are the original set and {8, 9} disjoint?", disjoint\_check)

# 32. Find the largest element of a set with a custom key

largest\_even = max(numbers, key=lambda x: x if x % 2 == 0 else float('-inf'))

print("Largest even number in the set:", largest\_even)

# 33. Use any() with a condition

any\_even = any(x % 2 == 0 for x in numbers)

print("Any even number in the original set?", any\_even)

# 34. Use all() with a condition

all\_positive = all(x > 0 for x in numbers)

print("All numbers in the original set are positive?", all\_positive)

# 35. Get the first element (if set is not empty)

first\_element = next(iter(numbers), None)

print("First element in the set:", first\_element)

# 36. List comprehension to create a set

list\_comprehension\_set = {x for x in range(10) if x > 5}

print("Set created using list comprehension (x > 5):", list\_comprehension\_set)

# 37. Combine two sets

combined\_set = numbers | another\_set

print("Combined set using | operator:", combined\_set)

# 38. Update the set with another set

numbers.update({8, 9})

print("Set after update with {8, 9}:", numbers)

# 39. Pop an element from the set

popped\_element = numbers.pop() if numbers else None

print("Popped element from the set:", popped\_element)

print("Set after popping an element:", numbers)

# 40. Checking for specific elements in the set using a loop

print("Checking for specific elements:")

for i in range(5):

exists = i in numbers

print(f"Does {i} exist in the set? {exists}")

# 41. Create a new set from a conditional operation

conditional\_set = {x for x in range(1, 20) if x % 3 == 0}

print("Set of numbers from 1 to 20 that are multiples of 3:", conditional\_set)

# 42. Create a set of prime numbers (basic example)

def is\_prime(n):

if n <= 1:

return False

for i in range(2, int(n\*\*0.5) + 1):

if n % i == 0:

return False

return True

prime\_set = {x for x in range(1, 30) if is\_prime(x)}

print("Set of prime numbers from 1 to 30:", prime\_set)

# 43. Using reduce to sum the set elements (importing from functools)

from functools import reduce

set\_sum = reduce(lambda x, y: x + y, numbers)

print("Sum using reduce:", set\_sum)

# 44. Create a set of tuples

tuple\_set = {(x, x\*\*2) for x in numbers}

print("Set of tuples (x, x^2):", tuple\_set)

# 45. Use zip to create a set of pairs from two lists

list1 = [1, 2, 3]

list2 = ['a', 'b', 'c']

zipped\_set = set(zip(list1, list2))

print("Zipped set from two lists:", zipped\_set)

# 46. Create a set from an existing set using copy()

copied\_set = numbers.copy()

print("Copied set:", copied\_set)

# 47. Calculate the symmetric difference with a new set

new\_set = {2, 3, 7}

sym\_diff = numbers.symmetric\_difference(new\_set)

print("Symmetric difference with {2, 3, 7}:", sym\_diff)

# 48. Count the number of elements greater than a specific value

count\_greater\_than\_2 = sum(1 for x in numbers if x > 2)

print("Count of elements greater than 2:", count\_greater\_than\_2)

# 49. Create a set from a generator expression

gen\_set = {x \* 2 for x in (x for x in range(10) if x % 2 == 0)}

print("Set from generator expression (doubled evens):", gen\_set)

# 50