Reverse a string:

```
def reverse_string(s):
    return s[::-1]
```

Check if a string is a palindrome:

```
def is_palindrome(s):
    return s == s[::-1]
```

Convert a string to uppercase:

```
def to_uppercase(s):
    return s.upper()
```

Convert a string to lowercase:

```
def to_lowercase(s):
    return s.lower()
```

Count the number of vowels in a string:

```
def count_vowels(s):
    vowels = 'aeiouAEIOU'
    return sum(1 for char in s if char in vowels)
```

Count the number of consonants in a string:

```
def count_consonants(s):
    vowels = 'aeiouAEIOU'
    return sum(1 for char in s if char.isalpha() and char not in vowels)
```

Remove all whitespaces from a string:

```
def remove_whitespaces(s):
    return s.replace(' ', '')
```

Find the length of a string without using the len() function:

```
def string_length(s):
    count = 0
    for char in s:
        count += 1
    return count
```

Check if a string contains a specific word:

```
def contains_word(s, word):
    return word in s
```

Replace a word in a string with another word:

```
def replace_word(s, old_word, new_word):
    return s.replace(old_word, new_word)
```

Count the occurrences of a word in a string:

```
def count_word_occurrences(s, word):
    return s.split().count(word)
```

Find the first occurrence of a word in a string:

```
def first_occurrence(s, word):
    return s.find(word)
```

Find the last occurrence of a word in a string:

```
def last_occurrence(s, word):
    return s.rfind(word)
```

Split a string into a list of words:

```
def split_into_words(s):
    return s.split()
```

Join a list of words into a string:

```
def join_words(words):
    return ' '.join(words)
```

Convert a string where words are separated by spaces to one where words are separated by underscores:

```
def spaces_to_underscores(s):
    return s.replace(' ', '_')
```

Check if a string starts with a specific word or phrase:

```
def starts_with(s, word):
    return s.startswith(word)
```

Check if a string ends with a specific word or phrase:

```
def ends_with(s, word):
    return s.endswith(word)
```

Convert a string to title case:

```
def to_title_case(s):
    return s.title()
```

Find the longest word in a string:

```
def longest_word(s):
```

```
    words = s.split()
    return max(words, key=len)
```

Find the shortest word in a string:

```
def shortest_word(s):
    words = s.split()
    return min(words, key=len)
```

Reverse the order of words in a string:

```
def reverse_words(s):
    words = s.split()
    return ' '.join(reversed(words))
```

Check if a string is alphanumeric:

```
def is_alphanumeric(s):
    return s.isalnum()
```

Extract all digits from a string:

```
def extract_digits(s):
    return ''.join(filter(str.isdigit, s))
```

Extract all alphabets from a string:

```
def extract_alphabets(s):
    return ''.join(filter(str.isalpha, s))
```

Count the number of uppercase letters in a string:

```
def count_uppercase(s):
    return sum(1 for char in s if char.isupper())
```

Count the number of lowercase letters in a string:

```
def count_lowercase(s):
    return sum(1 for char in s if char.islower())
```

Swap the case of each character in a string:

```
def swap_case(s):
    return s.swapcase()
```

Remove a specific word from a string:

```
def remove_word(s, word):
    return s.replace(word, '')
```

Check if a string is a valid email address:

```
import re

def is_valid_email(s):
    pattern = r'^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$'
    return re.match(pattern, s) is not None
```

Extract the username from an email address string:

```
def extract_username(email):
    return email.split('@')[0]
```

Extract the domain name from an email address string:

```
def extract_domain(email):
    return email.split('@')[1]
```

Replace multiple spaces in a string with a single space:

```
def replace_multiple_spaces(s):
    return ' '.join(s.split())
```

Check if a string is a valid URL:

```
import re

def is_valid_url(s):
    pattern = r'^(http|https)://[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}(/.*)?$'
    return re.match(pattern, s) is not None
```

Extract the protocol (http or https) from a URL string:

```
def extract_protocol(url):
    return url.split('://')[0]
```

Find the frequency of each character in a string:

```
from collections import Counter

def char_frequency(s):
    return dict(Counter(s))
```

Remove all punctuation from a string:

```
import string

def remove_punctuation(s):
    return s.translate(str.maketrans('', '', string.punctuation))
```

Check if a string contains only digits:

```
def is_only_digits(s):
    return s.isdigit()
```

Check if a string contains only alphabets:

```
def is_only_alphabets(s):
    return s.isalpha()
```

Convert a string to a list of characters:

```
def string_to_char_list(s):
    return list(s)
```

Check if two strings are anagrams:

```
def are_anagrams(s1, s2):
    return sorted(s1) == sorted(s2)
```

Encode a string using a Caesar cipher:

```
def caesar_cipher_encode(s, shift):
    def shift_char(c):
        if c.isalpha():
            start = ord('A') if c.isupper() else ord('a')
            return chr((ord(c) - start + shift) % 26 + start)
        return c
    return ''.join(shift_char(c) for c in s)
```

Decode a Caesar cipher encoded string:

```
def caesar_cipher_decode(s, shift):
    return caesar_cipher_encode(s, -shift)
```

Find the most frequent word in a string:

```
from collections import Counter

def most_frequent_word(s):
    words = s.split()
    return Counter(words).most_common(1)[0][0]
```

Find all unique words in a string:

```
def unique_words(s):
    words = s.split()
    return list(set(words))
```

Count the number of syllables in a string:

```
import re

def count_syllables(s):
    words = s.split()
    syllable_count = 0
    for word in words:
        syllable_count += len(re.findall(r'[aeiouyAEIOUY]+', word))
```

```
    return syllable_count
```

Check if a string contains any special characters:

```
import re

def contains_special_characters(s):
    return bool(re.search(r'[^a-zA-Z0-9]', s))
```

Remove the nth word from a string:

```
def remove_nth_word(s, n):
    words = s.split()
    if 0 <= n < len(words):
        del words[n]
    return ' '.join(words)
```

Insert a word at the nth position in a string:

```
def insert_word(s, word, n):
    words = s.split()
    words.insert(n, word)
    return ' '.join(words)
```

Convert a CSV string to a list of lists:

```
def csv_to_list_of_lists(csv_string):
    return [line.split(',') for line in csv_string.split('\n')]
```

Great! Let's continue with the list-based practice problems:

Create a list with integers from 1 to 10:

```
my_list = list(range(1, 11))
```

Find the length of a list without using the `len()` function:

```
def list_length(lst):
    count = 0
    for _ in lst:
        count += 1
    return count
```

Append an element to the end of a list:

```
def append_element(lst, element):
    lst.append(element)
    return lst
```

Insert an element at a specific index in a list:

```
def insert_element(lst, index, element):
```

```
    lst.insert(index, element)
    return lst
```

Remove an element from a list by its value:

```
def remove_element_by_value(lst, value):
    lst.remove(value)
    return lst
```

Remove an element from a list by its index:

```
def remove_element_by_index(lst, index):
    del lst[index]
    return lst
```

Check if an element exists in a list:

```
def element_exists(lst, element):
    return element in lst
```

Find the index of the first occurrence of an element in a list:

```
def first_occurrence_index(lst, element):
    return lst.index(element)
```

Count the occurrences of an element in a list:

```
def count_occurrences(lst, element):
    return lst.count(element)
```

Reverse the order of elements in a list:

```
def reverse_list(lst):
    return lst[::-1]
```

Sort a list in ascending order:

```
def sort_ascending(lst):
    return sorted(lst)
```

Sort a list in descending order:

```
def sort_descending(lst):
    return sorted(lst, reverse=True)
```

Create a list of even numbers from 1 to 20:

```
even_numbers = [x for x in range(1, 21) if x % 2 == 0]
```

Create a list of odd numbers from 1 to 20:

```
odd_numbers = [x for x in range(1, 21) if x % 2 != 0]
```

Find the sum of all elements in a list:

```
def sum_of_elements(lst):
    return sum(lst)
```

Find the maximum value in a list:

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

Find the minimum value in a list:

```
def min_value(lst):
    return min(lst)
```

Create a list of squares of numbers from 1 to 10:

```
squares = [x**2 for x in range(1, 11)]
```

Create a list of random numbers:

```
import random

random_numbers = [random.randint(1, 100) for _ in range(10)]
```

Remove duplicates from a list:

```
def remove_duplicates(lst):
    return list(set(lst))
```

Find the common elements between two lists:

```
def common_elements(lst1, lst2):
    return list(set(lst1) & set(lst2))
```

Find the difference between two lists:

```
def difference_between_lists(lst1, lst2):
    return list(set(lst1) - set(lst2))
```

Merge two lists:

```
def merge_lists(lst1, lst2):
    return lst1 + lst2
```

Multiply all elements in a list by 2:

```
def multiply_by_two(lst):
    return [x * 2 for x in lst]
```

Filter out all even numbers from a list:

```
def filter_even_numbers(lst):
    return [x for x in lst if x % 2 != 0]
```

Convert a list of strings to a list of integers:

```
def strings_to_integers(lst):
    return [int(x) for x in lst]
```

Convert a list of integers to a list of strings:

```
def integers_to_strings(lst):
    return [str(x) for x in lst]
```

Flatten a nested list:

```
def flatten_list(nested_lst):
    return [item for sublist in nested_lst for item in sublist]
```

Create a list of the first 10 Fibonacci numbers:

```
def fibonacci_numbers(n):
    fibs = [0, 1]
    for i in range(2, n):
        fibs.append(fibs[-1] + fibs[-2])
    return fibs[:n]
```

Check if a list is sorted:

```
def is_sorted(lst):
    return lst == sorted(lst)
```

Rotate a list to the left by n positions:

```
def rotate_left(lst, n):
    return lst[n:] + lst[:n]
```

Rotate a list to the right by n positions:

```
def rotate_right(lst, n):
    return lst[-n:] + lst[:-n]
```

Create a list of prime numbers up to 50:

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

prime_numbers = [x for x in range(2, 51) if is_prime(x)]
```

Split a list into chunks of size n:

```
def split_into_chunks(lst, n):
    return [lst[i:i + n] for i in range(0, len(lst), n)]
```

Find the second largest number in a list:

```
def second_largest(lst):
    unique_lst = list(set(lst))
    unique_lst.sort()
    return unique_lst[-2]
```

Replace every element in a list with its square:

```
def replace_with_squares(lst):
    return [x**2 for x in lst]
```

Convert a list to a dictionary where list elements become keys and their indices become values:

```
def list_to_dict(lst):
    return {element: index for index, element in enumerate(lst)}
```

Shuffle the elements of a list randomly:

```
import random

def shuffle_list(lst):
    random.shuffle(lst)
    return lst
```

Create a list of the first 10 factorial numbers:

```
def factorial(n):
    if n == 0:
        return 1
    else:
        return n * factorial(n-1)

factorial_numbers = [factorial(x) for x in range(10)]
```

Check if two lists have at least one element in common:

```
def have_common_elements(lst1, lst2):
    return bool(set(lst1) & set(lst2))
```

Remove all elements from a list:

```
def clear_list(lst):
    lst.clear()
    return lst
```

Replace negative numbers in a list with 0:

```
def replace_negatives(lst):
    return [0 if x < 0 else x for x in lst]
```

Convert a string into a list of words:

```
def string_to_word_list(s):
    return s.split()
```

Convert a list of words into a string:

```
def word_list_to_string(words):
    return ' '.join(words)
```

Create a list of the first n powers of 2:

```
def powers_of_two(n):
    return [2**x for x in range(n)]
```

Find the longest string in a list of strings:

```
def longest_string(lst):
    return max(lst, key=len)
```

Find the shortest string in a list of strings:

```
def shortest_string(lst):
    return min(lst, key=len)
```

Create a list of the first n triangular numbers:

```
def triangular_numbers(n):
    return [(x * (x + 1)) // 2 for x in range(1, n + 1)]
```

Check if a list contains another list as a subsequence:

```
def contains_subsequence(lst, sub_lst):
    sub_len = len(sub_lst)
    for i in range(len(lst) - sub_len + 1):
        if lst[i:i + sub_len] == sub_lst:
            return True
    return False
```

Swap two elements in a list by their indices:

```
def swap_elements(lst, index1, index2):
    lst[index1], lst[index2] = lst[index2], lst[index1]
    return lst
```

Create a tuple with integers from 1 to 5:

```
my_tuple = (1, 2, 3, 4, 5)
```

Access the third element of a tuple:

```
def access_third_element(tpl):
    return tpl[2]
```

Find the length of a tuple without using the len() function:

```
def tuple_length(tpl):
    count = 0
    for _ in tpl:
        count += 1
    return count
```

Count the occurrences of an element in a tuple:

```
def count_occurrences(tpl, element):
    return tpl.count(element)
```

Find the index of the first occurrence of an element in a tuple:

```
def first_occurrence_index(tpl, element):
    return tpl.index(element)
```

Check if an element exists in a tuple:

```
def element_exists(tpl, element):
    return element in tpl
```

Convert a tuple to a list:

```
def tuple_to_list(tpl):
    return list(tpl)
```

Convert a list to a tuple:

```
def list_to_tuple(lst):
    return tuple(lst)
```

Unpack the elements of a tuple into variables:

```
def unpack_tuple(tpl):
    a, b, c, d, e = tpl
    return a, b, c, d, e
```

Create a tuple of even numbers from 1 to 10:

```
even_tuple = tuple(x for x in range(1, 11) if x % 2 == 0)
```

Create a tuple of odd numbers from 1 to 10:

```
odd_tuple = tuple(x for x in range(1, 11) if x % 2 != 0)
```

Concatenate two tuples:

```
def concatenate_tuples(tpl1, tpl2):
    return tpl1 + tpl2
```

Repeat a tuple three times:

```
def repeat_tuple(tpl):
    return tpl * 3
```

Check if a tuple is empty:

```
def is_empty(tpl):
    return len(tpl) == 0
```

Create a nested tuple:

```
nested_tuple = ((1, 2), (3, 4), (5, 6))
```

Access the first element of a nested tuple:

```
def access_first_element(nested_tpl):
    return nested_tpl[0][0]
```

Create a tuple with a single element:

```
single_element_tuple = (1,)
```

Compare two tuples:

```
def compare_tuples(tpl1, tpl2):
    return tpl1 == tpl2
```

Delete a tuple:

```
def delete_tuple():
    tpl = (1, 2, 3)
    del tpl
    return "Tuple deleted"
```

Slice a tuple:

```
def slice_tuple(tpl, start, end):
    return tpl[start:end]
```

Find the maximum value in a tuple:

```
def max_value(tpl):
    return max(tpl)
```

Find the minimum value in a tuple:

```
def min_value(tpl):
    return min(tpl)
```

Convert a string to a tuple of characters:

```
def string_to_tuple(s):
    return tuple(s)
```

Convert a tuple of characters to a string:

```
def tuple_to_string(tpl):
    return ''.join(tpl)
```

Create a tuple from multiple data types:

```
mixed_tuple = (1, "hello", 3.14, True)
```

Check if two tuples are identical:

```
def are_identical(tpl1, tpl2):
    return tpl1 == tpl2
```

Sort the elements of a tuple:

```
def sort_tuple(tpl):
    return tuple(sorted(tpl))
```

Convert a tuple of integers to a tuple of strings:

```
def int_to_str_tuple(tpl):
    return tuple(str(x) for x in tpl)
```

Convert a tuple of strings to a tuple of integers:

```
def str_to_int_tuple(tpl):
    return tuple(int(x) for x in tpl)
```

Merge two tuples:

```
def merge_tuples(tpl1, tpl2):
    return tpl1 + tpl2
```

Flatten a nested tuple:

```
def flatten_tuple(nested_tpl):
    return tuple(item for sublist in nested_tpl for item in sublist)
```

Create a tuple of the first 5 prime numbers:

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

prime_tuple = tuple(x for x in range(2, 12) if is_prime(x))[:5]
```

Check if a tuple is a palindrome:

```
def is_palindrome(tpl):
    return tpl == tpl[::-1]
```

Create a tuple of squares of numbers from 1 to 5:

```
squares_tuple = tuple(x**2 for x in range(1, 6))
```

Filter out all even numbers from a tuple:

```
def filter_even_numbers(tpl):
    return tuple(x for x in tpl if x % 2 != 0)
```

Multiply all elements in a tuple by 2:

```
def multiply_by_two(tpl):
    return tuple(x * 2 for x in tpl)
```

Create a tuple of random numbers:

```
import random

random_tuple = tuple(random.randint(1, 100) for _ in range(10))
```

Check if a tuple is sorted:

```
def is_sorted(tpl):
    return tpl == tuple(sorted(tpl))
```

Rotate a tuple to the left by n positions:

```
def rotate_left(tpl, n):
    return tpl[n:] + tpl[:n]
```

Rotate a tuple to the right by n positions:

```
def rotate_right(tpl, n):
    return tpl[-n:] + tpl[:-n]
```

Create a tuple of the first 5 Fibonacci numbers:

```
def fibonacci_numbers(n):
    fibs = [0, 1]
    for i in range(2, n):
        fibs.append(fibs[-1] + fibs[-2])
    return tuple(fibs[:n])

fibonacci_tuple = fibonacci_numbers(5)
```

Create a tuple from user input:

```
def tuple_from_input():
    user_input = input("Enter elements separated by commas: ")
    return tuple(user_input.split(','))
```

Swap two elements in a tuple:

```
def swap_elements(tpl, index1, index2):
    lst = list(tpl)
    lst[index1], lst[index2] = lst[index2], lst[index1]
    return tuple(lst)
```

Reverse the elements of a tuple:

```
def reverse_tuple(tpl):
    return tpl[::-1]
```

Create a tuple of the first n powers of 2:

```
def powers_of_two(n):
    return tuple(2**x for x in range(n))
```

Find the longest string in a tuple of strings:

```
def longest_string(tpl):
    return max(tpl, key=len)
```

Find the shortest string in a tuple of strings:

```
def shortest_string(tpl):
    return min(tpl, key=len)
```

Create a tuple of the first n triangular numbers:

```
def triangular_numbers(n):
    return tuple((x * (x + 1)) // 2 for x in range(1, n + 1))
```

Check if a tuple contains another tuple as a subsequence:

```
def contains_subsequence(tpl, sub_tpl):
    sub_len = len(sub_tpl)
    for i in range(len(tpl) - sub_len + 1):
        if tpl[i:i + sub_len] == sub_tpl:
            return True
```

```
    return False
```

Create a tuple of alternating 1s and 0s of length n:

```
def alternating_ones_zeros(n):
    return tuple(1 if i % 2 == 0 else 0 for i in range(n))
```

Create a set with integers from 1 to 5:

```
my_set = {1, 2, 3, 4, 5}
```

Add an element to a set:

```
def add_element(s, element):
    s.add(element)
    return s
```

Remove an element from a set:

```
def remove_element(s, element):
    s.remove(element)
    return s
```

Check if an element exists in a set:

```
def element_exists(s, element):
    return element in s
```

Find the length of a set without using the len() function:

```
def set_length(s):
    count = 0
    for _ in s:
        count += 1
    return count
```

Clear all elements from a set:

```
def clear_set(s):
    s.clear()
    return s
```

Create a set of even numbers from 1 to 10:

```
even_set = {x for x in range(1, 11) if x % 2 == 0}
```

Create a set of odd numbers from 1 to 10:

```
odd_set = {x for x in range(1, 11) if x % 2 != 0}
```

Find the union of two sets:

```
def union_sets(s1, s2):
    return s1 | s2
```

Find the intersection of two sets:

```
def intersection_sets(s1, s2):
    return s1 & s2
```

Find the difference between two sets:

```
def difference_sets(s1, s2):
    return s1 - s2
```

Check if a set is a subset of another set:

```
def is_subset(s1, s2):
    return s1 <= s2
```

Check if a set is a superset of another set:

```
def is_superset(s1, s2):
    return s1 >= s2
```

Create a set from a list:

```
def set_from_list(lst):
    return set(lst)
```

Convert a set to a list:

```
def set_to_list(s):
    return list(s)
```

Remove a random element from a set:

```
def remove_random_element(s):
    s.pop()
    return s
```

Pop an element from a set:

```
def pop_element(s):
    return s.pop()
```

Check if two sets have no elements in common:

```
def no_common_elements(s1, s2):
    return s1.isdisjoint(s2)
```

Find the symmetric difference between two sets:

```
def symmetric_difference(s1, s2):
```

```
    return s1 ^ s2
```

Update a set with elements from another set:

```
def update_set(s1, s2):
    s1.update(s2)
    return s1
```

Create a set of the first 5 prime numbers:

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

prime_set = {x for x in range(2, 12) if is_prime(x)}
```

Check if two sets are identical:

```
def are_identical(s1, s2):
    return s1 == s2
```

Create a frozen set:

```
frozen_set = frozenset([1, 2, 3, 4, 5])
```

Check if a set is disjoint with another set:

```
def is_disjoint(s1, s2):
    return s1.isdisjoint(s2)
```

Create a set of squares of numbers from 1 to 5:

```
squares_set = {x**2 for x in range(1, 6)}
```

Filter out all even numbers from a set:

```
def filter_even_numbers(s):
    return {x for x in s if x % 2 != 0}
```

Multiply all elements in a set by 2:

```
def multiply_by_two(s):
    return {x * 2 for x in s}
```

Create a set of random numbers:

```
import random
```

```
random_set = {random.randint(1, 100) for _ in range(10)}
```

Check if a set is empty:

```
def is_empty(s):
    return len(s) == 0
```

Create a nested set (hint: use frozenset):

```
nested_set = {frozenset({1, 2}), frozenset({3, 4})}
```

Remove an element from a set using the discard method:

```
def discard_element(s, element):
    s.discard(element)
    return s
```

Compare two sets:

```
def compare_sets(s1, s2):
    return s1 == s2
```

Create a set from a string:

```
def set_from_string(s):
    return set(s)
```

Convert a set of strings to a set of integers:

```
def strings_to_integers(s):
    return {int(x) for x in s}
```

Convert a set of integers to a set of strings:

```
def integers_to_strings(s):
    return {str(x) for x in s}
```

Create a set from a tuple:

```
def set_from_tuple(tpl):
    return set(tpl)
```

Convert a set to a tuple:

```
def set_to_tuple(s):
    return tuple(s)
```

Find the maximum value in a set:

```
def max_value(s):
    return max(s)
```

Find the minimum value in a set:

```
def min_value(s):
    return min(s)
```

Create a set from user input:

```
def set_from_input():
    user_input = input("Enter elements separated by commas: ")
    return set(user_input.split(','))
```

Check if the intersection of two sets is empty:

```
def is_intersection_empty(s1, s2):
    return s1.isdisjoint(s2)
```

Create a set of the first 5 Fibonacci numbers:

```
def fibonacci_numbers(n):
    fibs = [0, 1]
    for i in range(2, n):
        fibs.append(fibs[-1] + fibs[-2])
    return set(fibs[:n])

fibonacci_set = fibonacci_numbers(5)
```

Remove duplicates from a list using sets:

```
def remove_duplicates(lst):
    return list(set(lst))
```

Check if two sets have the same elements, regardless of their count:

```
def same_elements(s1, s2):
    return s1 == s2
```

Create a set of the first n powers of 2:

```
def powers_of_two(n):
    return {2**x for x in range(n)}
```

Find the common elements between a set and a list:

```
def common_elements(s, lst):
    return s.intersection(lst)
```

Create a set of the first n triangular numbers:

```
def triangular_numbers(n):
    return {((x * (x + 1)) // 2) for x in range(1, n + 1)}
```

Check if a set contains another set as a subset:

```
def contains_subset(s1, s2):
    return s2.issubset(s1)
```

Create a set of alternating 1s and 0s of length n:

```
def alternating_ones_zeros(n):
    return {1 if i % 2 == 0 else 0 for i in range(n)}
```

Merge multiple sets into one:

```
def merge_sets(*sets):
    result = set()
    for s in sets:
        result.update(s)
    return result
```