String-Based Problems

1. Reverse a string :

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

2. Check if a string is a palindrome :

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

3. Convert a string to uppercase :

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

4. Convert a string to lowercase :

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

5. Count the number of vowels in a string :

```python
def count_vowels(s):
    return sum(1 for char in s if char.lower() in 'aeiou')
```

6. Count the number of consonants in a string :

```python
def count_consonants(s):
    return sum(1 for char in s if char.isalpha() and char.lower() not in 'aeiou')
```

7. Remove all whitespaces from a string :

```python
def remove_whitespaces(s):
    return s.replace(" ", "")
```

8. Find the length of a string without using `len()` :

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

9. Check if a string contains a specific word :

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

10. Replace a word in a string with another word :

```python
def replace_word(s, old, new):
    return s.replace(old, new)
```

11. Count the occurrences of a word in a string :

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

12. Find the first occurrence of a word in a string :

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

13. Find the last occurrence of a word in a string :

```python
```

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

14. Split a string into a list of words :

```python
def split_string(s):
    return s.split()
```

15. Join a list of words into a string :

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

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

```python
def convert_to_underscore(s):
    return s.replace(" ", "_")
```

```

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

```python
def starts_with(s, prefix):
    return s.startswith(prefix)
```

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

```python
def ends_with(s, suffix):
    return s.endswith(suffix)
```

19. Convert a string to title case :

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

20. Find the longest word in a string :

```python
def longest_word(s):
    words = s.split()
    return max(words, key=len)
```

21. Find the shortest word in a string :

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

22. Reverse the order of words in a string :

```python
def reverse_word_order(s):
    return ' '.join(s.split()[::-1])
```

23. Check if a string is alphanumeric :

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

24. Extract all digits from a string :

```python
def extract_digits(s):
    return ''.join([char for char in s if char.isdigit()])
```

25. Extract all alphabets from a string :

```python
def extract_alphabets(s):
    return ''.join([char for char in s if char.isalpha()])
```

List-Based Problems

1. Create a list with integers from 1 to 10 :

```python
list_1_to_10 = list(range(1, 11))
```

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

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

3. Append an element to the end of a list :

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

4. Insert an element at a specific index in a list :

```python
def insert_element(lst, index, element):
    lst.insert(index, element)
```

5. Remove an element from a list by its value :

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

6. Remove an element from a list by its index :

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

7. Check if an element exists in a list :

```python
```

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

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

```python
def first_index(lst, element):
    return lst.index(element)
```

9. Count the occurrences of an element in a list :

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

10. Reverse the order of elements in a list :

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

```

### Tuple-Based Problems

1. Create a tuple with integers from 1 to 5 :

```python
my_tuple = tuple(range(1, 6))
```

2. Access the third element of a tuple :

```python
def third_element(tup):
    return tup[2]  # Indexing starts from 0
```

3. Find the length of a tuple without using `len()` :

```python
def tuple_length(tup):
    count = 0
    for _ in tup:
```
```

```
            count += 1
        return count
```

4. Count the occurrences of an element in a tuple :

```python
def count_occurrences_in_tuple(tup, element):
    return tup.count(element)
```

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

```python
def first_index_in_tuple(tup, element):
    return tup.index(element)
```

Set-Based Problems

1. Create a set with integers from 1 to 5 :

```python
```

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

2. Add an element to a set :

```python
def add_element_to_set(my_set, element):
    my_set.add(element)
```

3. Remove an element from a set :

```python
def remove_element_from_set(my_set, element):
    my_set.remove(element)
```

4. Find the union of two sets :

```python
def union_of_sets(set1, set2):
    return set1.union(set2)
```

5. Find the intersection of two sets :

```python
def intersection_of_sets(set1, set2):
    return set1.intersection(set2)
```

These code snippets cover several of the problems related to strings, lists, tuples, and sets. You can try them individually to see how they work for different inputs! If you need further explanations or additional examples for other problems, feel free to ask.