Counting Character Occurrences in Words

```
words = ["apple", "banana", "cherry"]
char_count = {}
for word in words:
    for char in word:
        if char in char_count:
            char_count[char] += 1
        else:
            char_count[char] = 1
```

What is `char_count`?

Mapping Word Lengths

```
words = ["apple", "banana", "kiwi"]
word_lengths = {}
for word in words:
    word_lengths[word] = len(word)
```

What is `word_lengths`?

Grouping Words by First Character

```
words = ["ant", "bat", "apple", "banana", "cat"]
grouped_words = {}
for word in words:
    first_letter = word[0]
    if first_letter in grouped_words:
        grouped_words[first_letter].append(word)
    else:
        grouped_words[first_letter] = [word]
```

What is `grouped_words`?

Finding Vowels in Each Word

```
words = ["grape", "melon", "kiwi"]
vowels = "aeiou"
word_vowels = {}
for word in words:
    word_vowels[word] = []
    for char in word:
        if char in vowels and char not in word_vowels[word]:
            word_vowels[word].append(char)
```

What is `word_vowels`?

Reverse Indexing Words by Length

```
words = ["cat", "hippopotamus", "bat"]
length_dict = {}
for word in words:
    length = len(word)
    if length not in length_dict:
        length_dict[length] = []
    length_dict[length].append(word)
```

What is `length_dict`?

Creating Index Mapping of Words

```
words = ["alpha", "beta", "gamma"]
index_map = {}
for i, word in enumerate(words):
    index_map[word] = i
```

What is `index_map`?

Letter Frequency Across Words

```
words = ["hello", "world"]
letter_frequency = {}
for word in words:
    for char in word:
        if char in letter_frequency:
            letter_frequency[char] += 1
        else:
            letter_frequency[char] = 1
```

What is `letter_frequency`?

Tracking Indexes of Characters in Words

```
sentence = "programming"
index_dict = {}
for i, char in enumerate(sentence):
    if char in index_dict:
        index_dict[char].append(i)
    else:
        index_dict[char] = [i]
```

What is `index_dict`?

Grouping Strings by Length

```
words = ["tiny", "small", "big", "giant", "huge"]
length_groups = {}
for word in words:
    length = len(word)
    if length not in length_groups:
        length_groups[length] = []
    length_groups[length].append(word)
```

What is `length_groups`?

Creating a Dictionary of Words and Their Reversed

```
words = ["", "java", "swift"]
reversed_words = {}
for word in words:
    reversed_words[word] = word[::-1]
```

What is `reversed_words`?

Counting Words by Starting Character

```
words = ["apple", "apricot", "banana", "blueberry"]
start_char_count = {}
for word in words:
    first_char = word[0]
    if first_char in start_char_count:
        start_char_count[first_char] += 1
    else:
        start_char_count[first_char] = 1
```

What is `start_char_count`?

Mapping Words to Their Lengths, Excluding Specific Words

```
words = ["", "java", "html", "css"]
length_map = {}
exclude_words = ["html", "css"]
for word in words:
    if word not in exclude_words:
        length_map[word] = len(word)
```

What is `length_map`?

Building Indexes of Words with Certain Characters

```
words = ["apple", "banana", "pear", "peach"]
indexed_words = {}
for i, word in enumerate(words):
    if 'a' in word:
        indexed_words[word] = i
```

What is `indexed_words`?

Finding Common Characters in Words

```
words = ["program", "progress", "profound"]
common_chars = {}
for word in words:
    for char in word:
        if char in common_chars:
            common_chars[char] += 1
        else:
            common_chars[char] = 1
```

What is `common_chars`?

Building a Dictionary of Word Frequencies

```
words = ["apple", "banana", "apple", "cherry", "banana", "banana"]
freq_dict = {}
for word in words:
    if word in freq_dict:
        freq_dict[word] += 1
    else:
        freq_dict[word] = 1
```

What is `freq_dict`?

Mapping Strings by Length and Reversing Them

```
words = ["data", "science", ""]
reverse_map = {}
for word in words:
    if len(word) > 4:
        reverse_map[word] = word[::-1]
```

What is `reverse_map`?

Separating Words by Vowel Presence

```
words = ["cat", "dog", "eel", "bird"]
contains_vowel = {}
for word in words:
    if any(v in word for v in "aeiou"):
        contains_vowel[word] = True
    else:
        contains_vowel[word] = False
```

What is `contains_vowel`?

Creating a Dictionary with Unique Characters in Each Word

```
words = ["apple", "banana", "cherry"]
unique_chars = {}
for word in words:
    chars = set(word)
    unique_chars[word] = list(chars)
```

What is `unique_chars`?

Creating a Dictionary of Words and Index Ranges

```
words = ["apple", "banana", "cherry"]
index_ranges = {}
index = 0
for word in words:
```

```
    start = index
    index += len(word)
    index_ranges[word] = (start, index - 1)
```

What is `index_ranges`?

Checking Words for Duplicate Letters

```
words = ["hello", "world", ""]
duplicates = {}
for word in words:
    letter_set = set()
    has_duplicate = False
    for char in word:
        if char in letter_set:
            has_duplicate = True
            break
        letter_set.add(char)
    duplicates[word] = has_duplicate
```

What is `duplicates`?