Q1. Write and test a function that takes a string as a parameter and returns a sorted list of all the unique letters used in the string. So, if the string is cheese, the list returned should be ['c', 'e', 'h', 's'].

Ans:

```
def unique_sorted_letters(input_string):
    unique_letters = set(input_string.lower())
    unique_letters = {char for char in unique_letters if char.isalpha()}
    return sorted(unique_letters)

test_string = "cheese"
result = unique_sorted_letters(test_string)
print(result)
```

Output:

['c', 'e', 'h', 's']

Q2. Write and test three functions that each take two words (strings) as parameters and return sorted lists (as defined above) representing respectively:

Letters that appear in at least one of the two words.

Letters that appear in both words.

Letters that appear in either word, but not in both.'''

Ans:

```
def unique_sorted_letters(word):
    return sorted(set(char.lower() for char in word if char.isalpha()))

def letters_in_at_least_one(word1, word2):
```

```
    letters1 = unique_sorted_letters(word1)
    letters2 = unique_sorted_letters(word2)
    return sorted(set(letters1) | set(letters2))

def letters_in_both(word1, word2):
    letters1 = unique_sorted_letters(word1)
    letters2 = unique_sorted_letters(word2)
    return sorted(set(letters1) & set(letters2))

def letters_in_either_not_both(word1, word2):
    letters1 = unique_sorted_letters(word1)
    letters2 = unique_sorted_letters(word2)
    return sorted(set(letters1) ^ set(letters2))

word1 = "hello"
word2 = "world"

print("Letters in at least one:", letters_in_at_least_one(word1, word2))
print("Letters in both:", letters_in_both(word1, word2))
print("Letters in either not both:", letters_in_either_not_both(word1, word2))
```

Output:

Letters in at least one: ['d', 'e', 'h', 'l', 'o', 'r', 'w']

Letters in both: ['l', 'o']

Letters in either not both: ['d', 'e', 'h', 'r', 'w']

Q3. Write a program that manages a list of countries and their capital cities. It should prompt the user to enter the name of a country. If the program

already "knows" the name of the capital city, it should display it. Otherwise it should ask the user to enter it. This should carry on until the user terminates the program (how this happens is up to you).

Ans:

```
def main():
    countries = {}

    while True:
        country = input("Enter the name of a country (or type 'exit' to quit): ").strip()
        if country.lower() == 'exit':
            print("Exiting the program. Goodbye!")
            break

        if country in countries:
            print(f"The capital of {country} is {countries[country]}.")
        else:
            capital = input(f"I don't know the capital of {country}. Please enter it: ").strip()
            countries[country] = capital
            print(f"Thank you! I've added {country} with its capital {capital}.")

if __name__ == "__main__":
    main()
```

Output:

Enter the name of a country (or type 'exit' to quit): Nepal

I don't know the capital of Nepal. Please enter it: Kathmandu

Thank you! I've added Nepal with its capital Kathmandu.

Enter the name of a country (or type 'exit' to quit): exit

Exiting the program. Goodbye!

Q4. One approach to analysing some encrypted data where a substitution is suspected is frequency analysis. A count of the different symbols in the message can be used to identify the language used, and sometimes some of the letters. In English, the most common letter is "e", and so the symbol representing "e" should appear most in the encrypted text. Write a program that processes a string representing a message and reports the six most common letters, along with the number of times they appear. Case should not matter, so "E" and "e" are considered the same. Hint: There are many ways to do this. It is obviously a dictionary, but we will want zero counts, so some initialisation is needed. Also, sorting dictionaries is tricky, so best to ignore that initially, and then check the usual resources for the runes.

Ans:

```
def frequency_analysis(message):
    letter_counts = {}

    for char in message:

        if char.isalpha():
            char = char.lower()
            if char in letter_counts:
                letter_counts[char] += 1
            else:
                letter_counts[char] = 1

    sorted_counts = sorted(letter_counts.items(), key=lambda item: item[1], reverse=True)
    most_common = sorted_counts[:6]
```

```
    print("The six most common letters are:")
    for letter, count in most_common:
        print(f"Letter: '{letter}', Count: {count}")

if __name__ == "__main__":
    message = input("Enter a message for frequency analysis: ")
    frequency_analysis(message)
```

Output:

Enter a message for frequency analysis: Hello World

The six most common letters are:

Letter: 'l', Count: 3

Letter: 'o', Count: 2

Letter: 'h', Count: 1

Letter: 'e', Count: 1

Letter: 'w', Count: 1

Letter: 'r', Count: 1