In [1]:
```
import re
import random
import string

def is_password_strong(password, username=None):
    # Define password policy requirements
    # At least 8 characters
    if len(password) < 8:
        return False, "Password must be at least 8 characters long."

    # Contains at least one uppercase letter
    if not any(char.isupper() for char in password):
        return False, "Password must contain at least one uppercase letter

    # Contains at least one lowercase letter
    if not any(char.islower() for char in password):
        return False, "Password must contain at least one lowercase letter

    # Contains at least one digit
    if not any(char.isdigit() for char in password):
        return False, "Password must contain at least one digit."

    # Contains at least one special character
    special_characters = "!@#$%^&*()-_=+[{]}|;:,<.>/?"
    if not any(char in special_characters for char in password):
        return False, "Password must contain at least one special character

    # Check for consecutive characters (e.g., "123", "abc")
    if re.search(r'(.)\1\1', password):
        return False, "Password cannot contain consecutive characters (e.g

    # Check for sequences (e.g., "123", "abc")
    for i in range(len(password)-2):
        if ord(password[i]) == ord(password[i+1]) - 1 == ord(password[i+2]
            return False, "Password cannot contain sequential characters (e

    # Check if password contains username (case insensitive)
    if username and username.lower() in password.lower():
        return False, "Password cannot contain your username."

    # Check against a list of commonly used passwords (can be customized)
    common_passwords = ["password", "123456", "qwerty", "abc123"]
    if password.lower() in common_passwords:
        return False, "Password is too commonly used."

    # Password meets all requirements
    return True, "Password meets the strong password policy."


def generate_strong_password(length=12):
    # Generate a random strong password
    uppercase_letters = string.ascii_uppercase
    lowercase_letters = string.ascii_lowercase
    digits = string.digits
    special_characters = "!@#$%^&*()-_=+[{]}|;:,<.>/?"

    # Ensure at least one of each type of character
    password_characters = []
    password_characters.append(random.choice(uppercase_letters))
    password_characters.append(random.choice(lowercase_letters))
    password_characters.append(random.choice(digits))
```

```
    password_characters.append(random.choice(special_characters))

    # Fill the rest of the password with random characters
    remaining_length = length - 4  # subtract 4 because we already have 4 c
    password_characters.extend(random.choices(string.ascii_letters + string

    # Shuffle the password characters to make it random
    random.shuffle(password_characters)

    # Join the characters to form the password
    generated_password = ''.join(password_characters)

    return generated_password


# Example usage:
password = "StrongP@ssw0rd"
username = "example_user"

# Validate password
is_strong, message = is_password_strong(password, username)
if is_strong:
    print("Password meets the strong password policy.")
else:
    print(f"Password does not meet the strong password policy: {message}")

# Generate a strong password
generated_password = generate_strong_password()
print(f"Generated strong password: {generated_password}")
```

```
Password meets the strong password policy.
Generated strong password: rV4u/Lf[K>S*
```

In [ ]: